A Unified Adversarial Learning Framework for Semi-supervised Multi-target Domain Adaptation

Xinle Wu[1,2], Lei Wang[2](✉), Shuo Wang[1], Xiaofeng Meng[1], Linfeng Li[2,3], Haitao Huang[4], Xiaohong Zhang[5], and Jun Yan[2]

[1] School of Information, Renmin University of China, Beijing, China
{xinle.wu,shuowang,xfmeng}@ruc.edu.cn
[2] Yidu Cloud (Beijing) Technology Co., Ltd., Beijing, China
{lei.wang01,Linfeng.Li,jun.yan}@yiducloud.cn
[3] Institute of Information Science, Beijing Jiaotong University, Beijing, China
[4] The second Department of Neurology, Liaoning People's Hospital, Shenyang, China
[5] The Fourth Affiliated Hospital, China Medical University, Taichung, Taiwan, R.O.C.

Abstract. Machine learning algorithms have been criticized as difficult to apply to new tasks or datasets without sufficient annotations. Domain adaptation is expected to tackle this problem by establishing knowledge transfer from a labeled source domain to an unlabeled or sparsely labeled target domain. Most existing domain adaptation models focus on the single-source-single-target scenario. However, the pairwise domain adaptation approaches may lead to suboptimal performance when there are multiple target domains available, because the information from other related target domains is not being utilized. In this work, we propose a unified semi-supervised multi-target domain adaptation framework to implement knowledge transfer among multiple domains (a single source domain and multiple target domains). Specifically, we aim to learn an embedded space and minimize the marginal probability distribution differences among all domains in the space. Meanwhile, we introduce Prototypical Networks to perform classification, and extend it to semi-supervised settings. On this basis, we further align the conditional probability distributions among the domains by generating pseudo-labels for the unlabeled target data and training the model with bootstrapping method. Extensive sentiment analysis experiments show that our approach significantly outperforms several state-of-the-art methods.

Keywords: Domain adaptation · Adversarial learning · Semi-supervised · Prototypical networks · Self-training · Sentiment analysis

1 Introduction

Supervised learning algorithms have achieved great success in many fields with the availability of large quantities of labeled data. However, it is costly and time-consuming to annotate such large-scale training data for new tasks or datasets.

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12112, pp. 419–434, 2020.
https://doi.org/10.1007/978-3-030-59410-7_29

A naive idea is directly applying the model trained on a labeled source domain to the related and sparsely labeled target domain. Unfortunately, the model usually fails to perform well in the target domain due to domain shifts [24]. Domain adaptation (DA) is proposed to address this problem by transferring knowledge from a labeled source domain to a sparsely labeled target domain.

Existing DA methods can be divided into: supervised DA (SDA) [14,20,26], semi-supervised DA (SSDA) [11,21,23,31], and unsupervised DA (UDA) [4,5,9,15]. SDA methods assume that there are some labeled data in the target domain, and perform DA algorithms only use the labeled data. Conversely, UDA methods do not need any target data labels, but they require large amounts of unlabeled target data to align the distributions between domains. Considering that it is cheap to annotate a small number of samples and a few labeled data often leads to significant performance improvements, we focus on SSDA which exploits both labeled and unlabeled data in target domains.

Typical DA methods are designed to embed the data from the source and target domains into a common embedding space, and align the marginal probability distributions between the two domains. There are two approaches to achieve this, adversarial training [4,10,20,27] and directly minimizing the distance between the two distributions [15,18,28,35]. Both of the methods can generate domain-invariant feature representations for input data, and the representations from the source domain are used to train a classifier, which is then generalized to the target domain. However, only aligning the marginal distributions is not sufficient to ensure the success of DA [4,5,12,33], because the conditional probability distributions between the source and target domains may be different.

Most DA algorithms focus on the single-source-single-target setting. However, in many practical applications, there are multiple sparsely labeled target domains. For example, in the sentiment analysis task of product reviews, we can take the reviews of Books, DVDs, Electronics and Kitchen appliances as different domains. If we only have access to sufficient labeled data of Book reviews (source domain), and hope to transfer knowledge to the other domains, then each of the other domains can be seen as a target domain. In this case, pairwise adaptation approaches may be suboptimal, especially when there are shared features between the source and multiple target domains or the source and the target domain are associated through another target domain [9]. This is due to that these methods fail to leverage the knowledge from other relevant target domains. In addition, considering the distribution differences among multiple target domains, simply merging multiple target domains into a single one may not be the optimal solution.

To address these problems, we propose semi-supervised multi-target domain adaptation networks (MTDAN). Specifically, we use a shared encoder to extract the common features shared by all domains, and a private encoder to extract the domain-specific features of each domain. For feature representations generated by the two encoders, we train a domain discriminator to distinguish which domain they come from. To ensure that the shared representation is domain-invariant, the shared encoder is encouraged to generate the representation cannot be correctly distinguished by the domain discriminator. Given that there are only a few labeled data in each target domain, we introduce Prototypical Networks to perform classification, which is more superior than deep classifiers in few-shot scenarios [25]. We further leverage unlabeled data to refine prototypes, and extend Prototypical Networks to semi-supervised scenarios. Moreover, we utilize the self-training algorithm to exploit unlabeled target data, and we show that it can also align the class-conditional probability distributions among multiple domains.

Contributions. Our contributions are: a) We propose a unified adversarial learning framework for semi-supervised multi-target DA. b) We show that the prototype-based classifier can achieve better performance than the deep classifier when target domains have only a few labeled data and large amounts of unlabeled data. c) We show that the self-training algorithm can effectively align the class-conditional probability distributions among multiple domains. d) Our method outperforms several state-of-the-art DA approaches on sentiment analysis dataset.

2 Related Work

Domain Adaptation. Numerous domain adaptation approaches have been proposed to solve domain shift [29]. Most of them seek to learn a shared embedded space, in which the representations of source domain and target domain cannot be distinguished [27]. Based on that, the classifier trained with labeled source data can be generalized to the target domain. There are two typical ways to learn cross-domain representations: directly minimizing the distance between two distributions [15,17,18] and adversarial learning [6,7,26,27].

For the first method, several distance metrics have been proposed to measure the distance between source and target distributions. One common distance metric is the Maximum Mean Discrepancy (MMD) [2], which computes the norm of the difference between two domain means in the reproducing Kernel Hilbert Space (RKHS). Specifically, the DDC method [28] used both MMD and regular classification loss on the source to learn representations that are discriminative and domain invariant. The Deep Adaptation Network (DAN) [15] applied MMD to the last full connected layers to match higher order statistics of the two distributions. Most recently, [18] proposed to reduce domain shift in joint distributions of the network activation of multiple task-specific layers. Besides, Zellinger et al. proposed Center Moment Discrepancy (CMD) [32] to diminish the domain shift by aligning the central moment of each order across domains.

The other method is to optimize the source and target mappings using adversarial training. The idea is to train a domain discriminator to distinguish whether input features come from the source or target, whereas the feature encoder is trained to deceive the domain discriminator by generating representations that cannot be distinguished. [6] proposed the gradient reversal algorithm (ReverseGrad), which directly maximizes the loss of the domain discriminator by reversing its gradients. DRCN in [8] takes a similar approach in addition to learning to reconstruct target domain images. [3] enforced these adversarial losses in a shared feature space, while learned a private feature space for each domain to avoid the contamination of shared representations.

[4,27,33] argued that only aligning the marginal probability distributions between the source and target is not enough to guarantee successful domain adaptation. [16] proposed to align the marginal distributions and conditional distributions between the source and target simultaneously. [20] extended the domain discriminator to predict the domain and category of the embedded representation at the same time to align the joint probability distributions of input and output, and achieved a leading effect in the supervised domain adaptive scene. [5] proposed to align the class-conditional probability distributions between the source and target.

Recently, Zhao et al. [34] introduced an adversarial framework called MDAN, which is used for multi-source-single-target domain adaption. They utilized a multi-class domain discriminator to align the distributions between multiple source and a target domain. [9] proposed an information theoretic approach to solve unsupervised multi-target domain adaptation problem, which maximizes the mutual information between the domain labels and domain-specific features, while minimizes the mutual information between the the domain labels and the domain-invariant features. Unlike their approach, we base our method on self-training rather than entropy regularization. Moreover, we introduce prototypical networks to perform classification, which is more effective than deep classifiers in SSDA scenarios.

Semi-supervised Learning. Recently, some works treat domain adaptation as a semi-supervised learning task. [11] proposed a Domain Adaptive Semi-supervised learning framework (DAS) to jointly perform feature adaptation and semi-supervised learning. [21] applied a co-training framework for semi-supervised domain adaptation, in which the shared classifier and the private classifier boost each other to achieve better performance. [22] re-evaluated classic general-purpose bootstrapping approaches under domain shift, and proved that the classic bootstrapping algorithms make strong baselines on domain adaptation tasks.

3 Preliminaries

In this section, we introduce the notations and definitions related to single-source-multi-target DA.

Notations. We use $\mathcal{D}$ to denote a domain, which consists of an m-dimensional feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathrm{x})$, i.e., $\mathcal{D} = \{\mathcal{X}, P(\mathrm{x})\}$, where $\mathrm{x} \in \mathcal{X}$. We use $\mathcal{T}$ to denote a task which consists of a C-cardinality label set $\mathcal{Y}$ and a conditional probability distribution $P(y|\mathrm{x})$, i.e., $\mathcal{T} = \{\mathcal{Y}, P(y|\mathrm{x})\}$, where $y \in \mathcal{Y}$.

Problem Formulation (Single-Source-Multi-target Domain Adaptation). Let $\mathcal{D}_s = \{(\mathrm{x}_l^s, y_l^s)\}_{l=1}^{n_s}$ be a labeled source domain where n_s is the

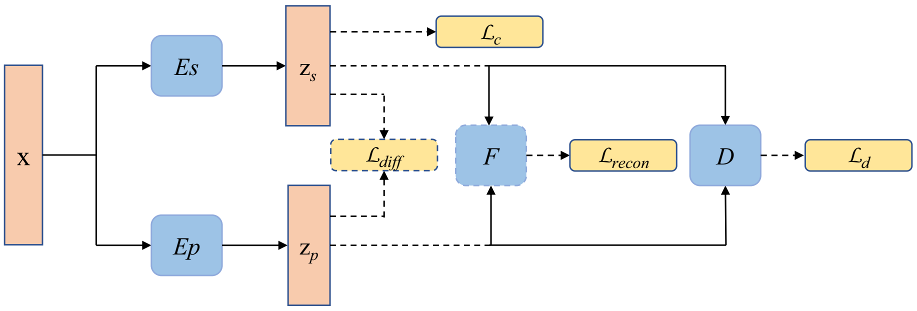


Fig. 1. The network structure of the proposed framework. The shared encoder E_s captures the common features shared among domains, while the private encoder E_p captures the domain-specific features. The shared decoder F reconstruct the input samples by using both the shared and private representations. The domain classifier D learns to distinguish which domain the input representations come from. The orthogonality constraint loss $\mathcal{L}_{diff}$ encourages E_s and E_p to encode different aspects of the inputs. The prototype-based classifier is computed on-the-fly, and the classification loss $\mathcal{L}_c$ is only used to optimize E_s.

number of labeled samples and let $\mathcal{D}_t = \{\mathcal{D}_{t_i}\}_{i=1}^{K}$ be multiple sparsely labeled target domains where $\mathcal{D}_{t_i} = \{(\mathrm{x}_l^{t_i}, y_l^{t_i})\}_{l=1}^{n_{l_i}} \bigcup \{\mathrm{x}_u^{t_i}\}_{u=1}^{n_{u_i}}$, K is the number of target domains, and $n_{l_i}(n_{l_i} \ll n_s)$ and $n_{u_i}(n_{u_i} \gg n_{l_i})$ refer to the number of labeled and unlabeled samples of i-th target domain respectively. We assume that all domains share the same feature space $\mathcal{X}$ and label space $\mathcal{Y}$, but the marginal probability distributions and the conditional probability distributions of source domain and multiple target domains are different from each other. The goal is to learn a classifier using the labeled source data and a few labeled target data, that generalizes well to the target domain.

4 Methodology

In this section, we describe each component and the corresponding loss function of the proposed framework in detail.

4.1 Proposed Approach

Our model consists of four components as shown in Fig. 1. A shared encoder E_s is trained to learn cross-domain representations, a private encoder E_p is trained to learn domain-specific representations, a shared decoder F is trained to reconstruct the input sample, and a discriminator D is trained to distinguish which domain the input sample comes from. Task classification is performed by calculating the distance from the domain-invariant representations to prototype representations of each label class.

Domain-Invariant and Domain-Specific Representations. We seek to extract domain-invariant (shared) and domain-specific (private) representations for each input $\mathbf{x}$ simultaneously. In our model, the shared encoder E_s and the private encoder E_p learn to generate the above two representations respectively:

$$\begin{aligned} \mathbf{z}_s &= E_s(\mathbf{x}, \boldsymbol{\theta}_s) \\ \mathbf{z}_p &= E_p(\mathbf{x}, \boldsymbol{\theta}_p) \end{aligned} \tag{1}$$

Here, $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_p$ refer to the parameters of E_s and E_p respectively, $\mathbf{z}_s$ and $\mathbf{z}_p$ refer to the shared and private representations of the input $\mathbf{x}$ respectively. Note that E_s and E_p can be MLP, CNN or LSTM encoders, depending on different tasks and datasets.

Reconstruction. In order to avoid information loss during the encoding, we reconstruct input samples with both shared and private representations. We use $\hat{\mathbf{x}}$ to denote the reconstruction of the input $\mathbf{x}$, which is generated by decoder F:

$$\hat{\mathbf{x}} = F(\mathbf{z}_s + \mathbf{z}_p, \boldsymbol{\theta}_f), \tag{2}$$

where $\boldsymbol{\theta}_f$ are the parameters of F. We use mean square error to define the reconstruction loss $\mathcal{L}_{Recon}$, which is applied to all domains:

$$\mathcal{L}_{Recon} = \frac{\lambda_r}{N} \sum_{i=1}^{N} \frac{1}{C} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2, \tag{3}$$

where C is the dimension of the input $\mathbf{x}$, N is the total number of samples in all domains, $\mathbf{x}_i$ refers to the i-th sample, λ_r is the hyper-parameter controlling the weight of the loss function, and $\|\cdot\|_2^2$ is the squared L_2-norm.

Orthogonality Constraints. To minimize the redundancy between shared and private representations, we introduce orthogonality constraints to encourage the shared and private encoders to encode different aspects of inputs. Specifically, we use $\mathbf{H}_s$ to denote a matrix, each row of which corresponds to the shared representation of each input $\mathbf{x}$. Similarly, let $\mathbf{H}_p$ be a matrix, each row of which corresponds to the private representation of each input $\mathbf{x}$. The corresponding loss function is:

$$\mathcal{L}_{Diff} = \lambda_{diff} \|\mathbf{H}_s^\top \mathbf{H}_p\|_F^2, \tag{4}$$

where λ_{diff} is the scale factor, $\|\cdot\|_F^2$ is the squared Frobenius norm.

Adversarial Training. The goal of adversarial training is to regularize the learning of the shared encoder E_s, so as to minimize the distance of distributions among source and multiple target domains. After that, we can apply the source classification model directly to the target representations. Therefore, we first train a domain discriminator D with the domain labels of the shared and private representations (since we know which domain each sample comes from, it is obvious that we can generate a domain label for each sample). The discriminator D is a multi-class classifier designed to distinguish which domain the

Algorithm 1. MTDAN Algorithm

Input: labeled source domain examples L_s, labeled multi-target domain examples $L_t = \{L_{t_i}\}_{i=1}^{K}$, unlabeled multi-target domain examples $U_t = \{U_{t_i}\}_{i=1}^{K}$

Hyper-parameters: coefficients for different losses: $\lambda_r, \lambda_d, \lambda_c, \lambda_{diff}$, mini-batch size b, learning rate η

```
 1: initialize $\boldsymbol{\theta}_s, \boldsymbol{\theta}_p, \boldsymbol{\theta}_f, \boldsymbol{\theta}_d$
 2: repeat
 3:     repeat
 4:         Sample a mini-batch from $\{L_s, L_t\}$
 5:         Train $F$ by minimizing $\mathcal{L}_{Recon}$
 6:         Train $D$ by minimizing $\mathcal{L}_D$
 7:         Train $E_p$ by minimizing $\mathcal{L}_P$
 8:         Train $E_s$ by minimizing $\mathcal{L}_S$
 9:     until Convergence
10:     Apply Eq.(9) to label $U_t$
11:     Select the most confident p positive and n negative predicted examples $U_t^l$
        from $U_t$
12:     Remove $U_t^l$ from $U_t$
13:     Add examples $U_t^l$ and their corresponding labels to $L_t$
14: until obtain best performance on the developing dataset
```

input representation comes from. Thus, D is optimized according to a standard supervised loss, defined below:

$$\mathcal{L}_D = \mathcal{L}_{D_p} + \mathcal{L}_{D_s}, \tag{5}$$

$$\mathcal{L}_{D_p} = -\frac{\lambda_d}{N} \sum_{i=1}^{N} \mathbf{d}_i^{\top} \log D(E_p(\mathbf{x}_i, \boldsymbol{\theta}_p), \boldsymbol{\theta}_d), \tag{6}$$

$$\mathcal{L}_{D_s} = -\frac{\lambda_d}{N} \sum_{i=1}^{N} \mathbf{d}_i^{\top} \log D(E_s(\mathbf{x}_i, \boldsymbol{\theta}_s), \boldsymbol{\theta}_d), \tag{7}$$

where $\mathbf{d}_i$ is the one-hot encoding of the i-th sample's domain label, $\boldsymbol{\theta}_d$ is the parameter of D, and λ_d is the scale factor.

Second, we train the shared encoder E_s to fool the discriminator D by generating cross-domain representations. We guarantee this by adding $-\mathcal{L}_{D_s}$ to the loss function of the shared encoder E_s. On the other hand, we hope the private encoder only extracts domain-specific features. Thus, we add $\mathcal{L}_{D_p}$ to the loss function of E_p to generate representations that can be distinguished by D.

Prototypical Networks for Task Classification. The simplest way to classify the target samples is to train a deep classifier, however, it may only achieve suboptimal performance as we can see in Table 1. The reason is that there are only a few labeled samples in each target domain, which is not enough to fine-tune a deep classifier with many parameters, so that the classifier is easy to over fit the source labeled data. Although we could generate pseudo-labeled data for target domains, the correctness of the pseudo-labels can not be guaranteed due to the poor performance of the deep classifier.

To efficiently utilize the labeled samples in target domains, we refer to the idea of prototypical networks [25]. Prototypical networks assume that there is a prototype in the latent space for each class, and the projections of samples belonging to this class cluster around the prototype. The classification is then performed by computing the distances to prototype representations of each class in the latent space. By reducing parameters of the model, the prototype-based classifier can achieve better performance than the deep classifier when labeled samples are insufficient. Note that we refine prototypes during self-training by allowing unlabeled samples with pseudo-labels to update the prototypes. Specifically, we compute the average of shared representations belonging to each class in a batch as prototypes:

$$\mathbf{c}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} E_s(\mathbf{x}_i, \boldsymbol{\theta}_s), \tag{8}$$

where n_k is the number of samples belonging to class k in a batch. Then we calculate a distribution by applying softmax function to distances between a shared representation with a prototype:

$$p(y = k|\mathbf{x}) = \frac{exp(-d(\mathbf{z}_s, \mathbf{c}_k))}{\sum_{k'} exp(-d(\mathbf{z}_s, \mathbf{c}'_k))}, \tag{9}$$

where $d(\cdot)$ is a distance measure function. We use the squared Euclidean distance in this work. The classification loss is defined as:

$$\mathcal{L}_C = -\lambda_c \log p(y = k|\mathbf{x}), \tag{10}$$

where λ_c is the scale factor.

Self-training for Conditional Distribution Adaptation. As described in [5,19], only aligning the marginal probability distributions between source and target is not enough to guarantee successful domain adaptation. Because this only enforces alignment of the global domain statistics with no class specific transfer. Formally, we can achieve $P_s(E_s(\mathrm{x}_s)) \approx P_{t_i}(E_s(\mathrm{x}_{t_i}))$ by introducing adversarial training, but $P_s(y_s|E_s(\mathrm{x}_s)) \neq P_{t_i}(y_{t_i}|E_s(\mathrm{x}_{t_i}))$ may still hold, where $P_s(y_s|E_s(\mathrm{x}_s))$ can be regarded as the classifier trained with source data.

Here, we tackle this problem by further reducing the difference of conditional probability distributions among source domain and target domains. In practice, we replace conditional probability distributions with class-conditional probability distributions, because the posterior probability is quite involved [16].

However, it is nontrivial to adapt class-conditional distributions, as most of the target samples are unlabeled. We address this problem by producing pseudo-labels for unlabeled target samples, and train the whole model in a bootstrapping way. As we perform more learning iterations, the number of target samples with correct pseudo-labels grows and progressively enforces distributions to align class-conditionally.

To be specific, we first train our model on labeled source and target samples. Then, we use the model to generate a probability distribution over classes for each unlabeled target sample. If the probability of a sample on a certain class is higher than a predetermined threshold τ, the sample would be added to the training set with the class as its pseudo-label.

Loss Function and Model Training. We alternately optimize the four modules of our model.

For E_p, the goal of training is to minimize the following loss:

$$\mathcal{L}_P = \mathcal{L}_{Recon} + \mathcal{L}_{Diff} + \mathcal{L}_{D_p} \tag{11}$$

For E_s, the goal of training is to minimize the following loss:

$$\mathcal{L}_S = \mathcal{L}_{Recon} + \mathcal{L}_{Diff} - \mathcal{L}_{D_s} + \mathcal{L}_C \tag{12}$$

For F and D, the losses are $\mathcal{L}_{Recon}$ and $\mathcal{L}_D$, respectively. The detailed training process is shown in algorithm 1.

5 Experiments

5.1 Dataset

We evaluate our proposed model on the Amazon benchmark dataset [1]. It is a sentiment classification dataset[1], which contains Amazon product reviews from four different domains: Books (B), DVD (D), Electronics (E), and Kitchen appliances (K). We remove reviews with neutral labels and encode the remaining reviews into 5000 dimensional feature vectors of unigrams and bigrams with binary labels indicating sentiment.

We pick two product as the source domain and the target domain in turn, and the other two domains as the auxiliary target domains, so that we construct 12 single-source-three-target domain adaptation tasks. For each task, the source domain contains 2,000 labeled examples, and each target domain contains 50 labeled examples and 2,000 unlabeled examples. To fine-tune the hyper-parameters, we randomly select 500 labeled examples from the target domain as the developing dataset.

[1] https://www.cs.jhu.edu/mdredze/datasets/sentiment/.

5.2 Compared Method

We compare MTDAN with the following baselines:

(1) **ST:** The basic neural network classifier without any domain adaptation trained on the labeled data of the source domain and the target domain.
(2) **CoCMD:** This is the state-of-the-art pairwise SSDA method on the Amazon benchmark dataset [21]. The shared encoder, private encoder and reconstruction decoder used in this model are the same as ours.
(3) **MTDA-ITA:** This is the state-of-the-art single-source-multi-target UDA method on three benchmark datasets for image classification [9]. We implemented the framework and extend it to semi-supervised DA method. The shared encoder, private encoder, reconstruction decoder and domain classifier used in this model are the same as ours.
(4) **c-MTDAN:** We combine all the target domains into a single one, and train it using MTDAN. Similarly, we also report the performance of c-CoCMD and c-MTDA-ITA.
(5) **s-MTDAN:** We do not use any auxiliary target domains, and train MTDAN on each source-target pair.

5.3 Implementation Details

Considering that each input sample in the dataset is a tf-idf feature vector without word ordering information, we use a multilayer perceptron (MLP) with an input layer (5000 units) and one hidden layer (50 units) and sigmoid activation functions to implement both shared and private encoders. The reconstruction decoder consists of one dense hidden layer (2525 units), tanh activation functions, and relu output functions. The domain discriminator is composed of a softmax layer with n-dimensional outputs, where n is the number of the source and target domains. For MTDA-ITA, we follow the framework proposed by [9], and use the above modules to replace the original modules in the framework. Besides, the task classifier for MTDA-ITA is a fully connected layer with softmax activation functions.

The network is trained with Adam optimizer [13] and with learning rate 10^{-4}. The mini-batch size is 50. The hyper-parameters $\lambda_r, \lambda_d, \lambda_c$ and λ_{diff} are empirically set to $1.0, 0.5, 0.1$ and 1.0 respectively. The threshold τ for producing pseudo-labels is set to 0.8. Following previous studies, we use classification accuracy metric to evaluate the performances of all approaches.

5.4 Results

The performances of the proposed model and other state-of-the-art methods are shown in Table 1. Key observations are summarized as follows. (1) The proposed model MTDAN achieves the best results in almost all tasks, which proves the effectiveness of our approach. (2) c-CoCMD has worse performance in all tasks compared with CoCMD, although c-CoCMD exploits labeled and unlabeled data from auxiliary target domains for training. Similar observation can also be observed by comparing MTDA-ITA with c-MTDA-ITA and MTDAN with c-MTDAN. This demonstrates that simply combine all target domains into a single one is not an effective method to solve the multi-target DA problem. (3) Our model outperforms CoCMD by an average of nearly 2.0%, which indicates that our model can effectively leverage the labeled and unlabeled data from multiple target domains. Similarly, our model performs better than its variant, s-MTDAN, which does not leverage the data from auxiliary target domains. This also shows that it is helpful to mine knowledge from auxiliary target domains. (4) Although MTDA-ITA is also a multi-target domain adaptation method, its performance is worse than that of MTDAN. This can be due to (i) self-training is a superior method than entropy regularization to exploit unlabeled target data, (ii) the prototype-based classifier is more efficient than the deep classifier in semi-supervised scenarios, (iii) we introduce orthogonality constraints to further reduce the redundancy between shared and private representations. (5) In the K→E task, MTDAN performs slightly worse than s-MTDAN. This can be explained that domain K is closer to domain E than the other domains as shown in Fig. 2 (a), and MTDAN leads to negative transfer when using relevant target domains to help domain adaptation. (6) s-MTDAN outperforms CoCMD in 9 of the 12 tasks, note that both of them do not use the auxiliary domains. This indicates that our model is more effective than CoCMD in pairwise domain adaptation task. (7) All models achieve better performance than the basic ST model, which demonstrates that domain adaptation methods are crucial when there exist a domain gap between the source domain and the target domain.

Table 1. Average classification accuracy with 5 runs on target domain testing dataset. The best is shown in bold. c-X: combining all target domains into a single one and performing pairwise domain adaptation with model X. s-X: performing pairwise domain adaptation between the original source and target domains with model X

Method	B→D	B→E	B→K	D→B	D→E	D→K	E→B	E→D	E→K	K→B	K→D	K→E
ST	81.6	75.8	78.2	80.0	77.0	80.4	74.7	75.4	85.7	73.8	76.6	85.3
CoCMD	83.1	83.0	85.3	81.8	83.4	85.5	76.9	78.3	87.3	77.2	79.6	87.2
c-CoCMD	82.7	82.2	84.5	80.6	83.0	84.8	76.3	77.6	87.1	75.9	79.4	86.1
MTDA-ITA	83.8	83.2	83.7	81.8	83.6	85.4	76.6	78.9	87.7	77.0	78.8	86.8
c-MTDA-ITA	83.3	82.3	83.2	81.4	83.0	85.0	76.0	79.3	87.6	76.7	78.5	87.0
s-MTDAN	83.3	83.9	84.7	81.6	83.7	84.7	78.0	80.2	87.9	78.6	79.9	**87.8**
c-MTDAN	84.0	84.0	85.5	81.7	84.3	85.9	80.2	80.7	88.1	79.8	80.5	87.0
MTDAN	**84.5**	**84.3**	**86.0**	**82.3**	**85.3**	**87.2**	**80.5**	**81.2**	**88.9**	**80.0**	**80.9**	87.4

5.5 Ablation Studies

We performed ablation experiments to verify the importance of each component of our proposed model. We report the results of removing orthogonality constraints loss (set λ_{diff}=0), self-training process, the prototype-based classifier (replaced by the deep classifier) respectively.

As we can see from Table 2, removing each of the above components causes performance degradation. To be specific, disabling self-training degrades the performance to the greatest extent, with an average decrease of 5.1%, which shows the importance of mining information from the unlabeled data of target domains. Similarly, replacing prototype-based classifiers with deep classifiers also leads to performance degradation, with an average decrease of 1.4%, which shows that the prototype-based classifiers is more effective than deep classifiers in semi-supervised scenarios. Besides, disabling the orthogonality constraints loss leads to a performance degradation of 0.7%, which indicates that encouraging the disjoint of shared and private representations can make the shared feature space more common among all domains.

We did not test the performance degradation caused by disabling reconstruction loss and multi-class adversarial training loss, because they have been proved in previous work [3,9]. To summarize, each of the proposed components helps improve classification performance, and using all of them brings the best performance.

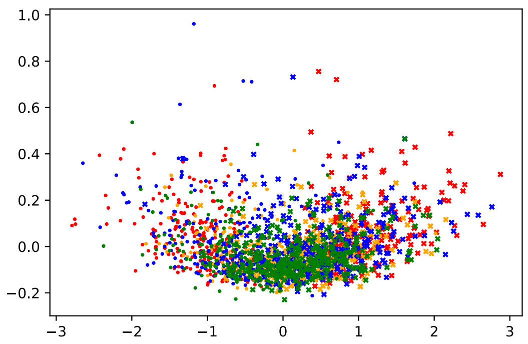
Table 2. Ablations. Performance of the proposed model when one component is removed or replaced. woDiff means without orthogonality constraints loss, woSelf means without self-training procedures, woProto means replace the prototype-based classifier with the deep classifier.

Method	B→D	B→E	B→K	D→B	D→E	D→K	E→B	E→D	E→K	K→B	K→D	K→E
MTDAN-*woDiff*	84.1	83.6	85.9	81.5	85.0	86.7	79.7	80.5	88.3	79.6	80.0	87.0
MTDAN-*woSelf*	82.8	77.6	80.0	81.6	78.8	81.5	74.8	75.6	86.7	74.5	77.3	86.7
MTDAN-*woProto*	83.3	83.8	84.1	81.8	83.8	86.9	79.2	78.9	87.9	78.0	80.3	86.4
MTDAN	**84.5**	**84.3**	**86.0**	**82.3**	**85.3**	**87.2**	**80.5**	**81.2**	**88.9**	**80.0**	**80.9**	**87.4**

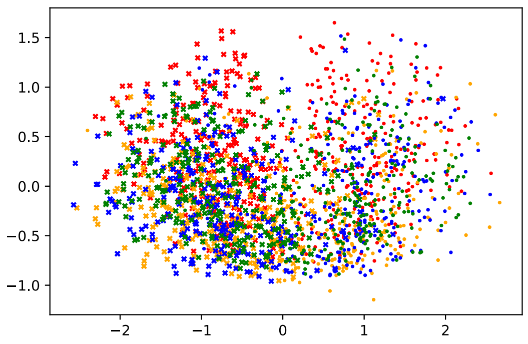
5.6 Feature Visualization

In order to understand the behavior of the proposed model intuitively, we project the shared and private encoder outputs into two-dimensional space with principle component analysis (PCA) [30] and visualize them. For comparison, we also show the visualization result of the basic ST model. Due to space constraints, we only show the visualization results of MTDAN with B as the source domain, E as the target domain, D and K as the auxiliary target domains. The results are shown in Fig. 2.

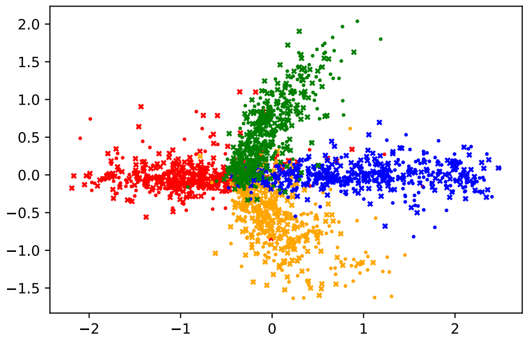
Figure 2 (a) shows the encoder output distribution of the ST model. As we can see, the distributions of domain B and domain D (called group 1) are similar and the distributions of domain E and domain D (called group 2) are similar, while the distributions of cross-group domains are relatively different. That's why the ST model gets worse classification performance when the source domain



(a) ST



(b) MTDAN-shared



(c) MTDAN-private

Fig. 2. Feature visualization for the embedding of source and target data. The red, blue, yellow and green symbols denote the samples from B, D, E and K respectively. The symbol 'x' is used for positive samples and '.' is for negative samples. (a) the distribution of the encoder output of ST, (b) the distribution of shared representations of MTDAN, (c) the distribution of private representations of MTDAN. For ST and MTDAN, we take B as the source domain and D, E and K as the target domains.

and the target domain belong to different groups. Besides, there is no obvious boundary between positive and negative samples, which is consistent with the poor performance of the ST model.

Figure 2 (b) shows the distribution of the shared encoder output of the MTDAN model. We can see that the shared representations of the source and target domains are very close, which demonstrates that our model can effectively align the marginal distributions among the source and multiple target domains. Meanwhile, for each class of samples, the shared representations of the source and target domains are also very close, which demonstrates that our model can effectively align the class-conditional distributions among multiple domains. Comparing (a) and (b), we can find that the boundary of positive and negative samples in (b) is more obvious than that in (a), which means the shared representations of MTDAN model have superior class separability.

Figure 2 (c) shows the distribution of the private encoder output of the MTDAN model. We can see that the private representations have good domain separability, partially because the domain discriminator D encourages the private encoder E_p to generate domain-specific feature representations.

6 Conclusion

In this paper, we propose MTDAN, a unified framework for semi-supervised multi-target domain adaptation. We utilize multi-class adversarial training to align the marginal probability distributions among source domain and multiple target domains. Meanwhile, we perform self-training on target unlabeled data to align the conditional probability distributions among the domains. We further introduce Prototypical Networks to replace the deep classifiers, and extend it to semi-supervised scenarios. The experimental results on sentiment analysis dataset demonstrate that our method can effectively leverage the labeled and unlabeled data of multiple target domains to help the source model achieve generalization, and is superior to the existing methods. The proposed framework could be used for other domain adaptation tasks, and we leave this as our future work.

Acknowledgment. This work was supported by National Natural Science Foundation of China (Grant No: 91646203, 91846204, 61532010, 61941121, 61532016 and 61762082). The corresponding author is Xiaofeng Meng.

References

1. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440–447 (2007)
2. Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)

3. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Advances in Neural Information Processing Systems, pp. 343–351 (2016)
4. Cicek, S., Soatto, S.: Unsupervised domain adaptation via regularized conditional alignment. arXiv preprint arXiv:1905.10885 (2019)
5. Gabourie, A.J., Rostami, M., Pope, P.E., Kolouri, S., Kim, K.: Learning a domain-invariant embedding for unsupervised domain adaptation using class-conditioned distribution alignment. In: 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 352–359. IEEE (2019)
6. Ganin, Y., Lempitsky, V.S.: Unsupervised domain adaptation by backpropagation. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 1180–1189 (2015)
7. Ganin, Y., et al.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(1), 2030–2096 (2016)
8. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 597–613. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_36
9. Gholami, B., Sahu, P., Rudovic, O., Bousmalis, K., Pavlovic, V.: Unsupervised multi-target domain adaptation: An information theoretic approach. arXiv preprint arXiv:1810.11547 (2018)
10. Guo, J., Shah, D.J., Barzilay, R.: Multi-source domain adaptation with mixture of experts. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 4694–4703 (2018)
11. He, R., Lee, W.S., Ng, H.T., Dahlmeier, D.: Adaptive semi-supervised learning for cross-domain sentiment classification. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 3467–3476 (2018)
12. Hosseini-Asl, E., Zhou, Y., Xiong, C., Socher, R.: Augmented cyclic adversarial learning for low resource domain adaptation. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019 (2019)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
14. Koniusz, P., Tas, Y., Porikli, F.: Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4478–4487 (2017)
15. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 97–105 (2015)
16. Long, M., Wang, J., Ding, G., Sun, J., Yu, P.S.: Transfer feature learning with joint distribution adaptation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2200–2207 (2013)
17. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Unsupervised domain adaptation with residual transfer networks. In: Advances in Neural Information Processing Systems, pp. 136–144 (2016)
18. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2208–2217 (2017). JMLR. org

19. Luo, Z., Zou, Y., Hoffman, J., Fei-Fei, L.F.: Label efficient learning of transferable representations across domains and tasks. In: Advances in Neural Information Processing Systems, pp. 165–177 (2017)
20. Motiian, S., Jones, Q., Iranmanesh, S., Doretto, G.: Few-shot adversarial domain adaptation. In: Advances in Neural Information Processing Systems, pp. 6670–6680 (2017)
21. Peng, M., Zhang, Q., Jiang, Y.g., Huang, X.J.: Cross-domain sentiment classification with target domain specific information. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2505–2513 (2018)
22. Ruder, S., Plank, B.: Strong baselines for neural semi-supervised learning under domain shift. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, 15–20 July 2018, Volume 1: Long Papers, pp. 1044–1054 (2018)
23. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. arXiv preprint arXiv:1904.06487 (2019)
24. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. J. Stat. Plann. Inference **90**(2), 227–244 (2000)
25. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems, pp. 4077–4087 (2017)
26. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4068–4076 (2015)
27. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7167–7176 (2017)
28. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
29. Wang, M., Deng, W.: Deep visual domain adaptation: a survey. Neurocomputing **312**, 135–153 (2018)
30. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometr. Intell. Lab. Syst. **2**(1–3), 37–52 (1987)
31. Yao, T., Pan, Y., Ngo, C.W., Li, H., Mei, T.: Semi-supervised domain adaptation with subspace learning for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2142–2150 (2015)
32. Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., Saminger-Platz, S.: Central moment discrepancy (CMD) for domain-invariant representation learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings (2017)
33. Zhao, H., des Combes, R.T., Zhang, K., Gordon, G.J.: On learning invariant representations for domain adaptation. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA, pp. 7523–7532 (2019)
34. Zhao, H., Zhang, S., Wu, G., Moura, J.M., Costeira, J.P., Gordon, G.J.: Adversarial multiple source domain adaptation. In: Advances in Neural Information Processing Systems, pp. 8559–8570 (2018)
35. Zhuang, F., Cheng, X., Luo, P., Pan, S.J., He, Q.: Supervised representation learning: Transfer learning with deep autoencoders. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)