Robust Target Training for Multi-Source Domain Adaptation

Zhongying Deng[1, 2]
z.deng@surrey.ac.uk
Da Li[3]
dali.academic@gmail.com
Yi-Zhe Song[1,2]
y.song@surrey.ac.uk
Tao Xiang[1, 2]
t.xiang@surrey.ac.uk

[1] University of Surrey
Guildford, UK
[2] iFlyTek-Surrey Joint Research Center on Artificial Intelligence
[3] Samsung AI Center
Cambridge, UK

Abstract

Given multiple labeled source domains and a single target domain, most existing multi-source domain adaptation (MSDA) models are trained on data from all domains jointly in one step. Such an one-step approach limits their ability to adapt to the target domain. This is because the training set is dominated by the more numerous and labeled source domain data. The source-domain-bias can potentially be alleviated by introducing a second training step, where the model is fine-tuned with the unlabeled target domain data only using pseudo labels as supervision. However, the pseudo labels are inevitably noisy and when used unchecked can negatively impact the model performance. To address this problem, we propose a novel Bi-level Optimization based Robust Target Training (BORT^2) method for MSDA. Given any existing fully-trained one-step MSDA model, BORT^2 turns it to a labeling function to generate pseudo-labels for the target data and trains a target model using pseudo-labeled target data only. Crucially, the target model is a stochastic CNN which is designed to be intrinsically robust against label noise generated by the labeling function. Such a stochastic CNN models each target instance feature as a Gaussian distribution with an entropy maximization regularizer deployed to measure the label uncertainty, which is further exploited to alleviate the negative impact of noisy pseudo labels. Training the labeling function and the target model poses a nested bi-level optimization problem, for which we formulate an elegant solution based on implicit differentiation. Extensive experiments demonstrate that our proposed method achieves the state of the art performance on three MSDA benchmarks, including the large-scale DomainNet dataset. Our code will be available at https://github.com/Zhongying-Deng/BORT2

1 Introduction

Deep convolutional neural networks (CNNs) have advanced significantly in the past decade. In particular, when trained with a large quantity of annotated data [5], CNNs have achieved remarkable performance gains over conventional non-CNN-based methods in almost all

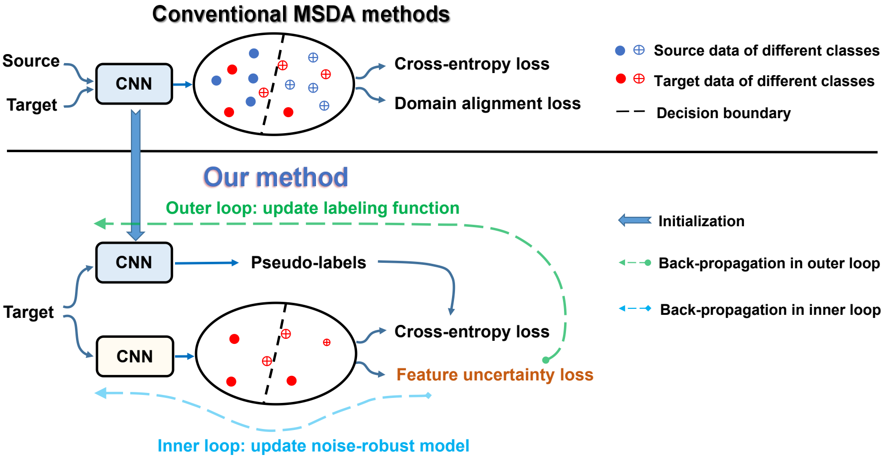

Conventional MSDA methods
Source
Target
CNN
Cross-entropy loss
Domain alignment loss
Source data of different classes
Target data of different classes
Decision boundary
Our method
Outer loop: update labeling function
Initialization
Back-propagation in outer loop
Back-propagation in inner loop
CNN
Pseudo-labels
Target
CNN
Cross-entropy loss
Feature uncertainty loss
Inner loop: update noise-robust model

Figure 1: Our method vs. conventional MSDA methods. Top: Conventional MSDA models are trained in one step using all domains aiming to extract domain-agnostic features. Bottom: Our method adds a second step training using the target domain data only. Concretely, the first-step model is fine-tuned to become a labeling function providing supervision for the final target MSDA model (yellow). A stochastic CNN layer is introduced in the final model to make it robust against label noise in the pseudo labels produced by the labeling function on the target data. Both CNNs (labeling function and final model) are learned jointly as a bi-level optimization problem consisting of an inner and outer loop, which is solved using implicit differentiation.

computer vision tasks, including image classification [10, 11, 25, 27], semantic segmentation [15] and object detection [23]. However, this exceptional performance relies on the I.I.D. assumption that the training and test data come from the same underlying distribution independently. When a trained model is applied to data from a different distribution to the training set, its performance often drops significantly. This issue is known as domain shift [2], and domain adaptation methods are developed to address it. A variety of unsupervised domain adaptation (UDA) methods have been proposed [1, 3, 9, 16, 17, 28, 31]. Early UDA studies have been focused on the single-source setting [7, 9, 29], i.e., adapting a model trained on a single labeled source domain to an unlabeled target domain. Nonetheless, when annotated data collected from multiple source domains are available, training with multiple source domains is expected to help. Therefore, the multi-source domain adaptation (MSDA) setting has received increasing attention since it was first introduced in [22].

Most MSDA methods [22, 30, 31, 37] adopt an one-step training strategy. As shown in Figure 1, they learn models with a shared backbone to extract domain-agnostic features. In this way, different domains can be aligned in a common feature space. However, completely aligning all the domains in one space is extremely difficult and sometimes even counterproductive [22]. This is because an one-step MSDA is prone to be biased to the source domains. In particular, since the source domains data are typically in larger quantity (multiple sources vs. one target) and are of higher quality (labeled vs. unlabeled), the one-step trained model would naturally favor the source domains. For instance, it has been observed that the batch norm statistics in a learned MSDA model can be highly source-domain biased [4, 20]. Since a MSDA model is only intended to be used in the target domain, such a bias thus must be addressed.

A naive way to alleviate this source-domain-bias is to introduce a second training step using the unlabeled target domain data only. Concretely, given an one-step MSDA model fully trained using both source and target domain data, the model is fine-tuned in the second step with the target domain data only. Since the target data are unlabeled, a self-training strategy is required, e.g., one can use the pseudo labels generated by the current model for the second-step training in an iterative fashion. Indeed, we find empirically that given any existing one-step MSDA model, adding a simple pseudo-label based second step training consistently brings a boost to its performance.

Though such a naive two-step approach can alleviate the source-domain-bias, it brings about another source of bias, i.e., the bias toward erroneous pseudo labels. More specifically, a well-trained first-step MSDA model would not be able to label all target domain data correctly. Otherwise, no second-step adaption is necessary in the first place. These noisy labels, once used directly as supervision, can amplify/re-enforce their bias through the iterations. Simply introducing a threshold to use the model confidence as a pseudo label quality measure can help to a certain extent. But again if we can fully trust the current model to tell us which label is correct, we perhaps do not need the second-step model adaption to start with.

In this work, we propose a novel bi-level optimization based robust target training (BORT^2) method for two-step MSDA (see Figure 1). In the first step, an existing one-step MSDA model is adopted and full-trained on both source and target domains. In the second step, BORT^2 uses it as a labeling function to generate pseudo-labels for the target domain data. The model is then trained using the pseudo-labeled target data only.

We introduce two novel designs to tackle the pseudo-label noise bias. First, the target model is designed to be robust against any noisy labels generated by the labeling function. Specifically, we introduce a stochastic CNN layer in the target model which models each target instance feature as a Gaussian distribution, consisting of a data dependent mean and variance. We then employ an entropy maximization loss to learn different feature uncertainties (i.e., variances caused by label noise) of different instances as per [33, 34]. With this uncertainty measure built in, it is now possible for the target model to identify and subsequently reduces the impact of the noisy labels on model training.

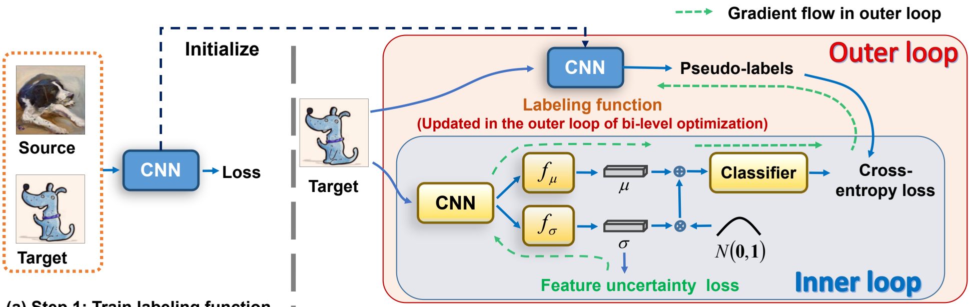
Second, we propose to train both the labeling function and the target model alternatively in a bi-level optimization with an efficient implicit differentiation based solution. That is, the first step (labeling function) and second step (target model) training becomes the outer and inner loops of a nested optimization that alternates between the two steps/loops. In this way, the labeling function can also be improved to produce less noise. However, solving this bi-level optimization problem is non-trivial for two reasons. (a) The labeling function, a deep CNN itself can now be viewed as a set of 'hyper-parameters' for the target stochastic CNN model. Nevertheless, 'hyper-parameter' optimization [18] typically requires a proper validation set for the outer loop learning objective. In our case, the target domain data is only pseudo labeled with noise, which may harm the optimization when directly used in a validation set. Our solution is to take advantage of the intrinsic uncertainty measure of our stochastic CNN to provide the outer loop learning signal. Concretely, in the inner loop we update the target model using the pseudo labels generated by the labeling function. We employ Gumbel-softmax [12] here when generating the pseudo labels to enable the differentiation of the labeling function. The outer loop computes the predicted feature entropy (uncertainty) of the current training (mini-batch) data using optimized target model in the inner loop. Given that smaller feature uncertainty usually implies an higher probability of accurate labels [33], the predicted feature uncertainty is minimized to help optimize the labeling function. (b) The hyper-parameters in our cases are the model parameters of a deep CNN, so are in the order of millions thus posing problems for gradient propagation. To overcome this challenge, we use the Neumann series based implicit function theorem [18] in our bi-level optimization to avoid the computational overload of caching the inner loop optimization trajectories, while maintaining the model convergence in the inner loop optimization.

We make the following contributions: (i) We propose to adopt a two-step training strategy for MSDA to overcome the source-domain-bias and observe empirically that even a naive pseudo-label based two-step approach brings clear performance boost to a variety of existing MSDA models. (ii) To deal with the noisy pseudo labels used for the second-step training, we further propose a novel noise robust training method termed BORT^2, which exploits stochastic CNN for robustness against label noise, and bi-level optimization with joint labeling function training. (iii) We show that the proposed BORT^2 is model agnostic and applicable to any base DA methods (verified with six different MSDA methods). State-of-the-art performance is obtained on three popular MSDA benchmarks, including Digit-Five [37], PACS [13] and DomainNet [22].

2 Related Work

Single-Source Domain Adaptation. Most single source domain adaptation methods alleviate domain shift by aligning feature distributions between the source and target domains. Some works achieve such feature alignment by minimizing different distance measures, such as maximum mean discrepancy (MMD) [9, 16] or Kullback-Leibler (KL) divergence [38]. Some other works employ adversarial training, such as the classic domain adversarial training like DANN [8] and the more recent prediction discrepancy based feature/classifier adversarial training, e.g., MCD [24]. Our method does not aim for source-target feature alignment. Instead, we focus on how to effectively utilize the target domain to train a model without source bias.

Multi-Source Domain Adaptation (MSDA). MSDA tackls more practical senerio where multiple source domains are available. Most MSDA methods still attempt to align feature distributions of different domains by using a shared backbone [22, 31, 35]. MDAN [35] and DCTN [31] exploit domain adversarial training by training multiple domain discriminators for different source-target domain pairs. M^{3}SDA-β [22] introduces the moment-based distribution distance for different domains. CMSS [32] learns a curriculum manager for source sample selection to enable better source/target alignment. LtC-MSDA [30] explores shared class knowledge among domains by constructing a knowledge graph on the class-wise prototypes of different domains, and exploits such knowledge for better inference. DAC-Net [6], which extracts domain-invariant features by imposing a consistency loss on the distributions of channel attention weights of different domains. DRT [14] turns multiple source domains into a single source domain problem by using a dynamic model and conduct the feature alignment in a single-source fashion. Since the shared backbone/classifier inevitably introduces source bias, MDDA [36] and STEM [21] adopts different backbones/classifiers for different domains. Although multiple backbones can alleviate the source bias, they introduce more parameters, especially when there are multiple source domains in MSDA. Different from these single-step MSDA methods, our work takes a different perspective to alleviate the domain shift and propose a two-step training pipeline. Benefiting from the novel noise robust training scheme, our model can be trained on the target domain only, resulting in better performance than those one-step alternatives.



(a) Step 1: Train labeling function on both source and target domains

(b) Step 2: Train noise-robust model only on the pseudo-labeled target domain

Figure 2: Overview of our BORT^2. It has two training steps. Step 1 trains a labeling function on both source and target domains. Step 2 trains a target model (the yellow CNN) with only pseudo-labeled target data. The pseudo-labels generated by the labeling function is used for supervised training of the noise-robust target model in the inner loop. The noise-robust model is fed with only target images and outputs predictions for cross-entropy calculation. It models the final feature representation as a Gaussian distribution, with the standard deviation representing the feature uncertainty caused by label noise. An entropy maximization loss is used for learning such feature uncertainty. This entropy loss and the cross-entropy loss are minimized in the inner loop to optimize the noise-robust model. Here, the labeling function is actually a hyper-network for optimizing the noise-robust model. So in the outer loop, we estimate the hyper-parameters of the labeling function for better label quality via bi-level optimization, which is achieved by minimizing the feature uncertainty.

3 Methodology

In this section, we will introduce the details of our proposed two-step training pipeline for MSDA, including first a naive two-step MSDA method and then our main contribution, the noise robust target model training method BORT^2. The overall training pipeline of BORT^2 is shown in Figure 2 and Algorithm 1.

Problem Setting. This paper focuses on multi-source domain adaptation (MSDA) for image classification. In MSDA, it is typically assumed that there are K labeled source domains $\mathcal{S} = \{\mathcal{S}_1,...,\mathcal{S}_K\}$ to adapt to an unlabeled target domain $\mathcal{T}$. Each source domain has $N_{\mathcal{S}_k}$ image and label pairs $\{(x_i^{\mathcal{S}_k}, y_i^{\mathcal{S}_k})\}_{i=1}^{N_{\mathcal{S}_k}}$. The target domain only contains unlabeled images $\mathcal{T} = \{x_i^{\mathcal{T}}\}_{i=1}^{N_{\mathcal{T}}}$ yet shares the same label space as the source domains. A model is then trained on $\mathcal{D} = \mathcal{S}_1 \cup ... \cup \mathcal{S}_K \cup \mathcal{T}$ jointly and evaluated on a test set of the target domain.

Two-Step Training. Our two-step training pipeline includes a normal MSDA training step using both source and target domain data, and a pseudo label based target domain only training step. This pipeline is designed to alleviate the source domain bias.

3.1 First-Step MSDA Training

Let us denote the training model F_θ, which is parameterized as θ. In the first training step of a two-step pipeline, the MSDA model is learned with the supervision loss from the source domain data and an adaptation loss to align the source and target domains. The overall optimization objective is formulated as

$$\arg\min_{\theta} \sum_{x^s,y^s \sim \mathcal{S}, x^t \sim \mathcal{T}} \mathcal{L}_{ce}(F_\theta(x^s), y^s) + \mathcal{L}_{da}(F_\theta(x^s), F_\theta(x^t)), \tag{1}$$

where, $\mathcal{L}_{ce}$ is a cross entropy loss, and $\mathcal{L}_{da}$ is a domain adaptation loss such as adversarial training [8] and moment matching [22]. This covers most existing MSDA methods. We also introduce FixMatch-CM in Supplementary as a new variant of first-step MSDA method.

3.2 Naive Second-Step Training

As shown in our experiments (see Section 4), a simple second step target domain training using pseudo labels can already bring clear improvement on performance, given a variety of existing MSDA models (see Figure 3 for a highlight). Let us give some details on this naive training method. Note that, in the second training step, there are no labels from the target domain data. Therefore, to train a model on the target domain only, taking a naive approach, we first generate the predictions $p = F_\theta(x), x \sim \mathcal{T}$ using the MSDA model trained in Section 3.1. We then convert p to "hard" labels:

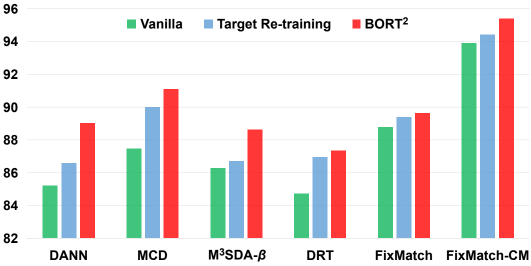


Figure 3: The performance of six one-step MSDA methods (Vanilla) on PACS is improved by a naive second-step target re-training. Our BORT^2 further improve the performance significantly.

$$\hat{y} = \arg\max(p). \tag{2}$$

Inspired by FixMatch [26], we also put a threshold τ to select the most confident "hard" labels. Meanwhile, we initialize a target domain model M_Ψ using F_θ, with M_Ψ trained as

$$\arg\min_{\Psi} \frac{1}{|\mathcal{T}|} \sum_i \mathbb{1}(\max(p_i) \geq \tau) \mathcal{L}_{ce}(M_\Psi(x_i), \hat{y}_i). \tag{3}$$

3.3 Bi-Level Optimization Based Noise-Robust Target Training

Even after thresholding, the pseudo labels generated for the naive approach is still noisy. Our BORT^2 is designed to solve two outstanding problems in the naive approach: 1) how to train a noise-robust model on the pseudo-labeled target domain with label noise. And 2) how to improve the labeling function further to provide higher-quality pseudo-labels. Two mechanisms are formulated in BORT^2 to solve these two problems respectively.

3.3.1 Stochastic Feature Uncertainty Modeling.

Inspired by the noisy-label learning methods in [33, 34], we introduce stochastic modeling in the fully-trained first-step model F_θ to turn it into a robust final model M_Ψ that can cope with the noisy pseudo labels used for supervision. More specifically, we introduce a stochastic layer to the final feature output of F_θ. Such a layer models each instance feature z_i^l produced by the l^{th} (final) feature layer of F_θ as a Gaussian distribution, i.e. $z_i^l \sim N(\mu_i, \sigma_i^2)$, in which μ, σ are generated as

$$\mu = f_{\Psi_\mu}(z^{l-1}), \quad \sigma = f_{\Psi_\sigma}(z^{l-1}), \tag{4}$$

where $z^{l-1} = f^{l-1} \circ \cdots \circ f^1(x) = f_{\Psi_0}(x)$, and f^i is a feature layer. $f_{\Psi_\mu}, f_{\Psi_\sigma}$ are the learnable layers that output μ, σ. x is the input with a pseudo label $\hat{y}$ sampled from the training set $\{x, \hat{y}\}_i$. And, a reparameterization trick is employed for enabling the back propagation as $z_i = \mu_i + \sigma_i \cdot \varepsilon$, where $\varepsilon \sim N(0, \mathbb{I})$. Then, a classifier $g_{\Psi_1}(.)$ is followed to classify z_i. The learning objective formula of the robust final model M_Ψ is

$$\underset{\Psi=\{\Psi_0,\Psi_\mu,\Psi_\sigma,\Psi_1\}}{\arg\min} \mathcal{L}_{trn} = \frac{1}{|\mathcal{T}|} \sum_{x_i \sim \mathcal{T}} \mathbb{1}(\max(p_i) \geq \tau) \mathcal{L}_{ce}(g_{\Psi_1}(z_i), \hat{y}_i) + \lambda \mathcal{L}_{ment}(f_{\Psi_\sigma}(f_{\Psi_0}(x_i))), \tag{5}$$

consisting of a cross-entropy loss $\mathcal{L}_{ce}$ and an entropy maximization loss $\mathcal{L}_{ment}(\sigma_i) = (m - \sum \log(\sigma_i))^+$ where m is a margin to bound the uncertainty. During the optimization, the optimizer will choose to assign larger standard deviation to the noisy labels as it will cancel its learning signal out, otherwise the loss will be enlarged significantly [33]. In other words, the model is able to automatically identify those uncertain therefore noisy instance labels and discount their influence on model training.

3.3.2 Bi-level Optimization of Labeling Function.

In this section, we will introduce how we further improve the labeling function to generate better-quality pseudo labels. The final model M_Ψ is trained with the pseudo labels generated by the first-step model F_θ. This means that the trained model is conditioned on the pseudo labels, i.e., the labeling function F_θ. Optimizing the function F_θ thus becomes an 'hyperparameter' optimization (HO) problem, which can be formulated as

$$\underset{\theta}{\text{argmin}}\, \mathcal{L}_{val}\big(\underset{\Psi}{\text{argmin}}\, \mathcal{L}_{trn}(M_\Psi, \mathcal{T}_{trn}; F_\theta), \mathcal{T}_{val}\big), \tag{6}$$

where θ can be regarded as the hyperparemters of model M_Ψ. $\mathcal{T}_{trn}$ and $\mathcal{T}_{val}$ are training and validation sets respectively, and $\mathcal{L}_{val}$ is the validation objective minimized to optimize θ.

In this bi-level optimization, the inner loop learning objective $\mathcal{L}_{trn}$ is the same as Eq. (5), except that the pseudo-label $\hat{y}_i$ is generated by using Gumbel-Softmax [12, 19] as $\hat{y}_i = \text{GumbelSoftmax}(F_\theta(x_i))$ to enable the back-propagation of F_θ in the outer loop optimization. Note that, typically $\mathcal{T}_{val}$ is a held-out validation set, which is used to compute the validation loss of the *best-response* model[18] M_Ψ to optimize the hyperparameters θ. However, in our case the target domain data is only pseudo-labeled with noise. Directly using a validation set constructed from those noisy pseudo labels will harm the outer loop optimization.

In the entropy maximization in Eq. (5), we know that the optimizer will choose to assign larger entropy (uncertainty) to the noisy labels. Therefore, the entropy can be explicitly used as a measure of how noisy a predicted label is. That is to say, labels with lower uncertainty are more likely to be accurate labels. Thus, we choose to use the entropy loss as our validation loss in Eq. (6), i.e.

$$\mathcal{L}_{val}(M_\Psi^*, \mathcal{T}_{val}) = \frac{1}{|\mathcal{T}_{val}|} \sum_{x_i \sim \mathcal{T}_{val}} \big(\sum \log(f_{\Psi_\sigma^*}(f_{\Psi_0^*}(x_i)))\big), \tag{7}$$

where $\Psi^* = \text{argmin}_\Psi\, \mathcal{L}_{trn}(M_\Psi, \mathcal{T}_{trn}; F_\theta)$ is the converged model in the inner loop under the hyperparameter θ. Note that we use $\mathcal{T}_{val} = \mathcal{T}_{trn}$ here. Our objective is to optimize the labeling

Algorithm 1: Training Procedure of BORT^2

```
input  : Training data $\{(x_i^{\mathcal{S}_1}, y_i^{\mathcal{S}_1}), \ldots, (x_i^{\mathcal{S}_K}, y_i^{\mathcal{S}_K})\}_{i=1}^{B}$, $\{x_i^{\mathcal{T}}\}_{i=1}^{B}$.
output: The target domain model $M_\psi$.
while not converge do
    Update labeling function $F_\theta$ via any existing MSDA methods.
end
while not converge do
    Update target domain model $M_\psi$ via Eq. (5).
    Bi-level optimization for labeling function $F_\theta$:
        Inner loop optimization according to Eq. (5) with Gumbel-softmax.
        Outer loop optimization via minimizing Eq. (6) using Neumann
          approximation [18].
end
```

function such that the predicted feature uncertainty of *training data* is low when using the generated labels from the labeling function. Therefore, it makes more sense to validate the feature uncertainty of the training set for the sake of optimizing our labelling function.

During the outer optimization, the hypergradient [18] of θ is computed as

$$\frac{\partial \mathcal{L}_{val}(M_{\Psi^*})}{\partial \theta} = \frac{\partial \mathcal{L}_{val}}{\partial M_{\Psi^*}} \frac{\partial M_{\Psi^*}}{\partial \theta}, \tag{8}$$

where $\frac{\partial \mathcal{L}_{val}}{\partial M_{\Psi*}}$ can be straightforwardly computed using existing deep learning tools, e.g. PyTorch. $\frac{\partial M_{\Psi^*}}{\partial \theta}$ can be decomposed into $\frac{\partial M_{\Psi^*}}{\partial \theta} = -\left[\frac{\partial^2 \mathcal{L}_{trn}}{\partial \Psi \partial \Psi}\right]^{-1} \times \frac{\partial^2 \mathcal{L}_{trn}}{\partial \Psi \partial \theta}$ according to Implicit Function Theorem [18]. Computing the inverse Hessian is not tractable in the high dimensional space. Therefore, we use a recently published Neumann approximation [18].

4 Experiments

We experiment on three popular MSDA datasets, including PACS [13], Digit-Five, and DomainNet [22]. The experimental setting are provided in Supplementary Material.

4.1 Comparative Results

Competitors We compare our method with the following competitors introduced in Section 2: DANN [8], MCD [24], MDAN [35], DCTN [31], M^{3}SDA-β [22], LtC-MSDA [30], DAC-Net [6], CMSS [32], MDDA [36], DRT [14] and STEM [21]. Most of these methods try to reduce domain gap via a one-step training, thus can hardly alleviate source-domain-bias.

Table 1: MSDA results on PACS. Best results are in bold.

Method	Art.	Cartoon	Sketch	Photo	Avg.
Oracle	99.53	99.84	99.53	99.92	99.71
Source-only	81.22	78.54	72.54	95.45	81.94
MDAN [35]	83.54	82.34	72.42	92.91	82.80
DCTN [31]	84.67	86.72	71.84	95.60	84.71
M^{3}SDA-β [22]	84.20	85.68	74.62	94.47	84.74
MDDA [36]	86.73	86.24	77.56	93.89	86.11
LtC-MSDA [30]	90.19	90.47	81.53	97.23	89.85
DAC-Net [6]	91.39	91.39	84.97	97.93	91.42
BORT^2 (*Ours*)	**95.02**	**94.51**	**93.23**	**98.74**	**95.38**

Table 2: MSDA results on Digit-Five. * denotes that standard deviations are not reported in the paper.

Method	**MNIST**	**USPS**	**MNIST-M**	**SVHN**	**Synthetic**	**Avg.**
Oracle	99.5±0.03	99.1±0.05	95.0±0.29	90.7±0.26	97.8±0.02	96.4
Source-only [32]	92.3±0.91	90.7±0.54	63.7±0.83	71.5±0.75	83.4±0.79	80.3
DANN [8]	97.9±0.83	93.4±0.79	70.8±0.94	68.5±0.85	87.3±0.68	83.6
DCTN [31]	96.2±0.80	92.8±0.30	70.5±1.20	77.6±0.40	86.8±0.80	84.8
MCD [24]	96.2±0.81	95.3±0.74	72.5±0.67	78.8±0.78	87.4±0.65	86.1
M^{3}SDA-β [22]	98.4±0.68	96.1±0.81	72.8±1.13	81.3±0.86	89.6±0.56	87.6
CMSS [32]	99.0±0.08	97.7±0.13	75.3±0.57	88.4±0.54	93.7±0.21	90.8
LtC-MSDA [30]	99.0±0.40	98.3±0.40	85.6±0.80	83.2±0.60	93.0±0.50	91.8
DRT [14]	**99.3**±0.05	98.4±0.12	81.0±0.34	86.7±0.38	93.9±0.34	91.9
DAC-Net [6]	99.2±0.03	**98.7**±0.11	86.0±0.44	91.6±0.16	97.1±0.18	94.5
STEM [21]*	99.4	98.4	89.7	89.9	**97.5**	95.0
BORT^2 (*Ours*)	98.8±0.08	98.4±0.08	**93.0**±0.06	**91.9**±0.19	**97.5**±0.08	**95.9**

Table 3: MSDA results on DomainNet.

Method	**Clipart**	**Infograph**	**Painting**	**Quickdraw**	**Real**	**Sketch**	**Avg.**
Oracle	79.7±0.16	41.0±0.18	71.4±0.11	72.6±0.70	83.7±0.13	70.59±0.06	69.8
Source-only [22]	47.6±0.52	13.0±0.41	38.1±0.45	13.3±0.39	51.9±0.85	33.7±0.54	32.9
DANN [8]	45.5±0.59	13.1±0.72	37.0±0.69	13.2±0.77	48.9±0.65	31.8±0.62	32.6
DCTN [31]	48.6±0.73	23.5±0.59	48.8±0.63	7.2±0.46	53.5±0.56	47.3±0.47	38.2
MCD [24]	54.3±0.64	22.1±0.70	45.7±0.63	7.6±0.49	58.4±0.65	43.5±0.57	38.5
M^{3}SDA-β [22]	58.6±0.53	26.0±0.89	52.3±0.55	6.3±0.58	62.7±0.51	49.5±0.76	42.6
CMSS [32]	64.2±0.18	28.0±0.20	53.6±0.39	16.0±0.12	63.4±0.21	53.8±0.35	46.5
LtC-MSDA [30]	63.1±0.50	28.7±0.70	56.1±0.50	16.3±0.50	66.1±0.60	53.8±0.60	47.4
DRT [14]	69.7±0.24	**31.0**±0.56	59.5±0.43	9.9±1.03	68.4±0.28	59.4±0.21	49.7
DAC-Net [6]	72.5±0.04	27.6±0.10	57.8±0.06	23.0±0.14	66.7±0.10	59.5±0.12	51.2
STEM [21]	72.0	28.2	**61.5**	25.7	**72.6**	60.2	**53.4**
BORT^2 (*Ours*)	**74.0**±0.04	29.1±0.19	59.6±0.06	**28.0**±0.02	69.3±0.04	**60.3** ±0.14	**53.4**

PACS From Table 1, we can see that BORT^2 is superior to these competitors on all four transfer tasks, leading to an average accuracy of 3.96% improvement over other baselines. On some difficult setups, such as Sketch and Art Painting as target domains, BORT^2 outperforms the second best method by 8.26% and 3.63% respectively. This demonstrate the strong robustness of our BORT^2 under large domain shifts.

Digit-Five As shown in Table 2, BORT^2 achieves significant improvement over the previous state-of-the-art methods, e.g. 4% better than DRT in average accuracy, 1.4% than DAC-Net and ∼1% than STEM. In particular, our BORT^2 obtains comparable performance to the oracle result, demonstrating the high-quality pseudo-labels generated. On the MNIST-M domain, BORT^2 shows biggest improvement over the other competitors (with 3.3%).

DomainNet Table 3 shows that BORT^2 achieves comparable performance with STEM, but does not adopt classifier ensemble strategy as STEM. In addition, BORT^2 beats the other competitors considerably, with more than 2.2% performance gain. On the most challenging target domain Quickdraw, our BORT^2 obtains more than 2.0% improvement over the other methods. This further verifies the effectiveness of BORT^2 for addressing large domain shift, thanks to its robust target training.

4.2 Further Analysis

Importance of a Second Step Training. We verify the contribution of our proposed robust target training here. From Figure 3, we can see that a simple second step target domain

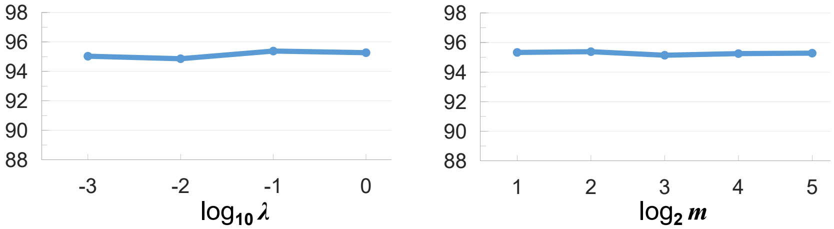


Figure 4: Sensitivity of λ and m in L_{ment}.

training using pseudo labels improves all six differnet base MSDA methods, resulting in accuracy improvements of 1.37% on DANN [8], 2.53% on MCD [24], 0.42% on M^{3}SDA-β [22], 2.21% on DRT [14] and 0.59%, 0.54% on two FixMatch [26] variants (or see #3 vs. #4 in Table 4). Incorporating our proposed robust training further improves this second step training, with a up to 2.43% accuracy gain.

Table 4: Ablation study of BORT^2 on PACS.

#	Methods	Avg
1	BORT^2	**95.38**
2	BORT^2 (w/o bi-level optimization)	94.80
3	BORT^2 (w/o noise-robust model)	94.43
4	FixMatch-CM	93.89

Importance of Optimizing Labeling Function. In the second step of BORT^2, we propose to optimize the labeling function by a bi-level optimization. To verify its effectiveness, we remove the outer loop in Eq. (7) from #1 but keep the stochastic modelling. This leads to a model without bi-level optimization, further resulting in a fixed labeling function. From Table 4 #2 we can see that without this bi-level optimization, the performance decreases by 0.58% from #1. This indeed shows that optimizing the labeling function is helpful to improve the quality of pseudo-labels.

Importance of Noise-Robust Training. We further evaluate the noise-robust training used in the second step of BORT^2 by replacing the feature uncertainty based stochastic model in #2 with a vanilla CNN. This leads to a naive second-step training in Table 4#3. Comparing #3 with #2, we observe a performance drop, suggesting that this stochastic modelling is helpful.

Sensitivity of Hyper-Parameters Recall that in our proposed BORT^2, we have two hyper-parameters: the weight λ and threshold m in the entropy maximization loss L_{ment} (see Eq. (5)). We first fix m to 4 and vary λ from 0.001 to 1. The results are in Figure 4 (left panel). It is clear that the performance is generally stable, and the best performance of 95.38% is obtained at $\lambda = 0.1$ (i.e., $\log_{10}\lambda = -1$). We then set λ to 0.1 and adjust m from 2 to 32. The results (right panel) show that the performance is also insensitive to m, with the best accuracy achieved at $m = 4$ (i.e., $\log_2 m = 2$).

See Supplementary for more experimental results.

5 Conclusion

We have proposed a novel two-step training method for MSDA task, namely bi-level optimization based robust target training (BORT^2). BORT^2 first learns a labeling function using both the source and target data, then trains a noise-robust model only on the pseudo-labeled target domain. The noise-robust model exploits feature uncertainty to detect label noise and alleviate its negative impact. We further employ a bi-level optimization method to optimize the labeling function for better label quality. Extensive experiments on three MSDA datasets demonstrate that our BORT^2 achieves new state-of-the-art performance.

References

[1] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Normalized wasserstein for mixture distributions with applications in adversarial learning and domain adaptation. In *ICCV*, 2019.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *ML*, 2010.

[3] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, 2018.

[4] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, 2019.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[6] Zhongying Deng, Kaiyang Zhou, Yongxin Yang, and Tao Xiang. Domain attention consistency for multi-source domain adaptation. In *BMVC*, 2021.

[7] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[9] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[13] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *CVPR*, 2017.

[14] Yunsheng Li, Lu Yuan, Yinpeng Chen, Pei Wang, and Nuno Vasconcelos. Dynamic transfer for multi-source domain adaptation. In *CVPR*, 2021.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[16] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[17] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016.

[18] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[19] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[20] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *CVPR*, 2018.

[21] Van-Anh Nguyen, Tuan Nguyen, Trung Le, Quan Hung Tran, and Dinh Phung. Stem: An approach to multi-source domain adaptation with guarantees. In *ICCV*, 2021.

[22] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[24] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[28] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[29] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[30] Hang Wang, Minghao Xu, Bingbing Ni, and Wenjun Zhang. Learning to combine: Knowledge aggregation for multi-source domain adaptation. In *ECCV*, 2020.

[31] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, 2018.

[32] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Shrivastava. Curriculum manager for source selection in multi-source domain adaptation. In *ECCV*, 2020.

[33] Tianyuan Yu, Da Li, Yongxin Yang, Timothy M Hospedales, and Tao Xiang. Robust person re-identification by modelling feature uncertainty. In *ICCV*, 2019.

[34] Tianyuan Yu, Yongxin Yang, Da Li, Timothy Hospedales, and Tao Xiang. Simple and effective stochastic neural networks. In *AAAI*, 2021.

[35] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *NeurIPS*, 2018.

[36] Sicheng Zhao, Guangzhi Wang, Shanghang Zhang, Yang Gu, Yaxian Li, Zhichao Song, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source distilling domain adaptation. In *AAAI*, 2020.

[37] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *arXiv preprint arXiv:2003.07325*, 2020.

[38] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *IJCAI*, 2015.