A Prototype-Oriented Framework for Unsupervised Domain Adaptation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Existing methods for unsupervised domain adaptation often rely on minimizing the divergence between source and target samples in the latent space. To avoid the sampling variability, class imbalance, and data-privacy concerns that often plague these methods, we instead provide a probabilistic transport framework to align target features with class prototypes. Our approach does not require additional networks and introduces only one additional hyper-parameter. We demonstrate the general applicability of our method on a wide range of scenarios, including single-source, multi-source, class-imbalance, and source-private domain adaptation. With the same complexity as using the source model alone, the proposed method achieves competitive performance with state-of-the-art methods in terms of accuracy.

1 Introduction

In many real-world applications, such as healthcare and autonomous driving, data labeling can be expensive and time-consuming. To make predictions on a new unlabeled dataset, one may naively use an existing supervised model trained on a large labeled dataset. However, even subtle changes in the data-collection conditions, such as lighting or background for natural images, can cause a model's performance to degrade drastically [1]. This shift in the input data distribution is referred to in the literature as covariate shift [2]. By leveraging the labeled samples from the source domain and unlabeled samples from the target domain, unsupervised domain adaptation aims to overcome this issue, making the learned model generalize well in the target domain [3].

Ben-David et al. [4, 5] provide an $\mathcal{H}$-divergence based theoretical upper bound on the target error. Ganin [6] popularizes learning an invariant representation between the source and target domains to minimize this divergence. Several works follow this trend and use different divergences to minimize the inter-domain gap [7, 8]. Recent works in domain-adaptation theory, however, suggest that matching the marginal distributions of the source and target in the latent space can lead to sub-optimal performance [9]. If the label proportions shift between domains, learning domain invariant representation does not lead to improvements over using source data alone to train the model [10].

Numerous prior methods [7, 11–15] focus on using source and target samples for feature alignment in the latent space. While this approach can reduce the discrepancy between domains, using source samples for feature alignment has the following problems. First, several commonly used methods that can be used to quantify the difference between two empirical distributions, such as maximum mean discrepancy (MMD) [16] and Wasserstein distance [17, 18], suffer from sampling variability when used to match the source and target marginal distributions [19, 20]. Second, while we typically assume that the two domains share the same label space, we cannot guarantee that the samples drawn from the source and target domains will have overlapping classes in each mini-batch. If we pull the support of the source and target feature representations from different classes closer together, the classifier will be more likely to misclassify those examples. Finally, aligning features to features

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

means that we need access to both source and target data simultaneously. In applications such as personal healthcare, we may not have access to source data directly; instead, we may only be given access to the trained model and the target data.

We propose an algorithm that constructs class prototypes to behave as representative samples from the latent space of the source domain. Using the prototypes instead of source features avoids the following problems: **1)** sampling variability of the source features, **2)** instance class-mismatching in a mini-batch, and **3)** source-data privacy concerns. As we expect the classifier to make better predictions on the target data in regions where the source density is sufficiently high [21], it is natural to consider encouraging the feature encoder to map the target data close to these prototypes. Motivated by the cluster assumption [22] (decision boundaries should not cross high-density regions of the data), we further provide a framework to transport the target features to these class prototypes and vice versa. Our bi-directional transport framework avoids the high computational complexity that comes with solving the optimal transport problem, while simultaneously achieving more competitive performance. Moreover, we extend our method to address the potential shift in label proportions, a problem that is common and has been studied [23–27], but which has also been often overlooked in prior works [6–8].

Compared to existing methods, our approach has several appealing aspects. First, our method does not rely on adversarial training to achieve competitive performance, making the algorithm converge much faster. Moreover, learnable prototypes not only avoid expensive computation but also bypass accessing source data directly. This attribute makes our algorithm applicable to the settings where data privacy is a major concern. Unlike clustering-based approaches that typically require multiple forward passes before an update, our algorithm processes data in mini-batches for each update and is trained in an end-to-end manner.

We highlight the main contributions as follows: **1)** We utilize the linear classifier's weights as class prototypes and propose a general framework to align target features to these prototypes. **2)** We further introduce the minimization of a bi-directional transport cost for alignment and illustrate its superior performance over related methods. **3)** We test the proposed method under multiple challenging yet practical conditions: single-source, multi-source, class-imbalance, and source-private domain adaptation. In comparison to state-of-the-art algorithms, the proposed prototype-oriented method achieves highly competitive performance in domain adaptation with the same complexity as the source model.

2 Prototype-oriented conditional transport

In this section, we propose Prototype-oriented Conditional Transport (PCT), a holistic method for domain adaption consisting of three parts: learning class prototypes, aligning the target features with learned prototypes using bidirectional conditional transport [28], and estimating the target class proportions. Our method does not introduce additional networks for aligning domain features, and the model can be learned in an end-to-end manner.

In domain adaptation, we are given a labeled dataset from the source domain, $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s} \sim \mathcal{D}_s$, and an unlabeled dataset from the target domain, $\{\boldsymbol{x}_j^t\}_{j=1}^{n_t} \sim \mathcal{D}_t^{\boldsymbol{x}}$. We focus on the closed-category domain adaptation and assume that the source and target domains share the same label space, $i.e.$, $y_i^s, y_j^t \in \{1, 2, \ldots, K\}$, where K denotes the number of classes. The goal of domain adaptation is to learn a model with low risk on the target samples. The model typically consists of a feature encoder, $F_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^{d_f}$, parameterized by $\boldsymbol{\theta}$, and a linear classification layer $C_{\boldsymbol{\mu}} : \mathbb{R}^{d_f} \to \mathbb{R}^K$, parameterized by $\boldsymbol{\mu}$. In prior works [6–8], the feature encoder is a pre-trained neural network, and the classifier is a randomly initialized linear mapping. To simplify the following notation, we denote $\boldsymbol{f}_i^s = F_{\boldsymbol{\theta}}(\boldsymbol{x}_i^s)$ and $\boldsymbol{f}_j^t = F_{\boldsymbol{\theta}}(\boldsymbol{x}_j^t)$ as the feature representations of the source data $\boldsymbol{x}_i^s$ and target data $\boldsymbol{x}_j^t$, respectively.

2.1 Learning class prototypes

Most existing works [6–8] focus on aligning the source and target features in a latent space. By contrast, we propose to characterize the features of each class with a class prototype and align the target features with these class prototypes instead of the source features. This approach has several advantages. First, the feature alignment between two domains would be more robust to outliers in the source domain as we avoid using the source samples directly. Second, we do not need to worry about

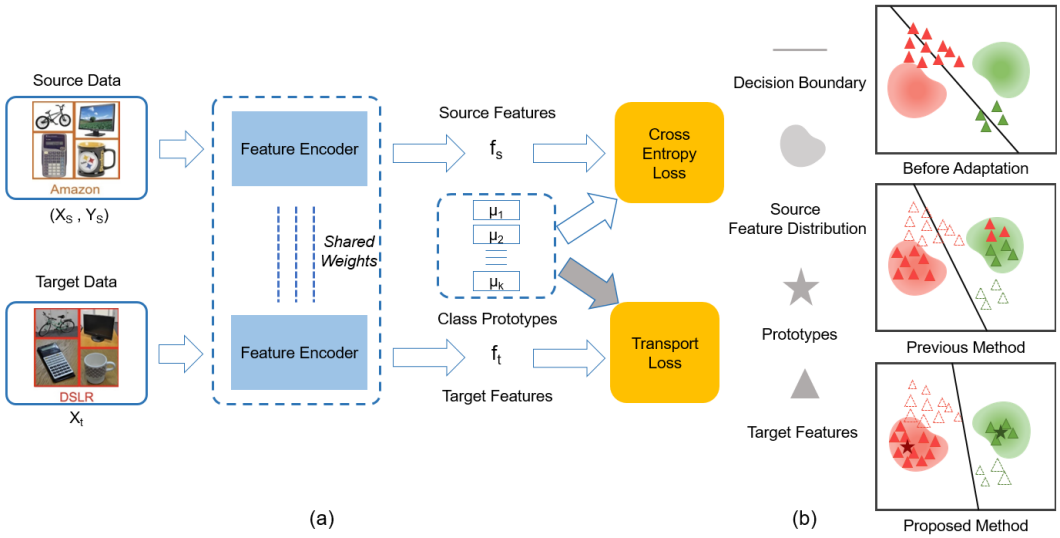


Figure 1: In Figure (a), we exhibit a diagram of Prototype-oriented Conditional Transport (PCT). Unlike existing methods that align target and source features, our method aligns target features with class prototypes, the linear classifier's weights. The gray arrow indicates that the gradients of the prototypes do not back-propagate through the transport loss. In Figure (b), we illustrate the cluster assumption in the latent space. When there is class imbalance, matching the marginal feature distribution will inevitably encourage the feature encoder to map target data to the wrong region in the latent space of the source domain, leading to misclassification. We avoid this issue by using prototypes and calculate the transport probabilities based on similarity between features and prototypes.

the missing classes in the source domain like we do when we align the features of a mini-batch of source samples and target samples. Prototypes ensure that every class is represented for each training update. Last but not least, we show that using the inferred prototypes instead of source features allows us to adapt to the target domain even without accessing the source data in the adaptation stage, which is an appealing trait when preserving the source data privacy is a concern (see Table 5).

Previous methods [29, 30, 15] construct each class prototype as the average latent feature for that class extracted by the feature encoder, which is computationally expensive due to the forward passing of the training samples. We propose to construct class prototypes in the same latent space but with learnable parameters: $[\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K] \in \mathbb{R}^{d_f \times K}$, where the dimension of each prototype, d_f, is the same as the hidden dimension after the feature encoder $F_{\boldsymbol{\theta}}$. We learn each class prototype in a way that encourages the prototype to be close to the source samples associated with that class in the feature space. In particular, given the prototypes and a source sample $\boldsymbol{x}_i^s$ with feature $\boldsymbol{f}_i^s$, we define the predictive probability for $\boldsymbol{x}_i^s$ to be classified to class k as $p_{ik}^s := \frac{\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_i^s + b_k)}{\sum_{k'=1}^K \exp(\boldsymbol{\mu}_{k'}^T \boldsymbol{f}_i^s + b_{k'})}$, where b_k is a bias term. Then we use the cross-entropy loss to learn the prototypes:

$$\mathcal{L}_{\text{cls}} = \mathbb{E}_{(\boldsymbol{x}_i^s, y_i^s) \sim \mathcal{D}_s} \left[\sum_{k=1}^K -\log p_{ik}^s \mathbf{1}_{\{y_i^s = k\}} \right]. \tag{1}$$

We note that this way of learning the class prototypes is closely connected to learning the standard linear classification layer $C_{\boldsymbol{\mu}}$ on source-only data with the cross-entropy loss. The neural network weights in the classification layer can be interpreted as the class prototypes. Therefore, compared with source-only approaches, constructing prototypes in this way does not introduce additional parameters. As we show in Figure 2a, our method requires much fewer parameters than other domain-adaptation methods. Moreover, this approach requires much less computation than other prototype-based methods by avoiding averaging the latent features for each class.

2.2 Bi-directional prototype-oriented conditional transport

In this section, we will discuss how we encourage the feature encoder to align the target data with class prototypes. Our approach is motivated by the cluster assumption, which has been widely

adopted in both semi-supervised learning [22, 31, 32] and domain-adaptation literature [33–36]. The cluster assumption states that the input data distribution consists of separated clusters and that instances belonging to the same cluster have the same class labels. This means that the decision boundaries should not cross high-density regions. To achieve this goal, we minimize the expected pairwise cost between target features and prototypes with respect to two different joint distributions. Each joint distribution is decomposed into a conditional and a marginal distribution. The first conditional distribution, $\pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t)$, is designed to encourage the target features to be close to each prototype, whereas the other conditional distribution, $\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_j^t \mid \boldsymbol{\mu}_k)$, helps prevent a trivial solution. By minimizing this expected cost, the target feature will be close to the prototypes, far from the decision boundaries.

2.2.1 Transport from target domain to class prototypes

First, we define the conditional transport cost from the target features to class prototypes. The transport probabilities from a target feature $\boldsymbol{f}_j^t$ to each prototype $\boldsymbol{\mu}_k$ is defined as a conditional probability:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t) = \frac{p(\boldsymbol{\mu}_k)\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{k'=1}^{K} p(\boldsymbol{\mu}_{k'})\exp(\boldsymbol{\mu}_{k'}^T \boldsymbol{f}_j^t)}, \tag{2}$$

where, through the lens of Bayes' rule, $p(\boldsymbol{\mu}_k)$ is the discrete prior distribution over the classes for the target domain, and $\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)$ plays the role of a likelihood term, measuring the similarity between class prototypes and target features. Intuitively, target features are more likely to be transported to prototypes which correspond to dominant classes in the target domain or which are closer to the target features (or both). Note that in practice we often do not have access to the target class distribution $p(\boldsymbol{\mu}_k)$. We can use a uniform distribution for $p(\boldsymbol{\mu}_k)$. However, this could be sub-optimal, especially when classes are seriously imbalanced in the target domain. To address this issue, we propose a way to estimate $p(\boldsymbol{\mu}_k)$ in section 2.3.

We now define the expected transport cost from the target features to class prototypes as:

$$\mathcal{L}_{t\to\mu} = \mathbb{E}_{\boldsymbol{x}_j^t \sim \mathcal{D}_t^{\boldsymbol{x}}} \mathbb{E}_{\boldsymbol{\mu}_k \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t)} \left[c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) \right], \tag{3}$$

where $c(\cdot,\cdot)$, a point-to-point transport cost measuring the dissimilarity between two features, is defined with the cosine distance as

$$c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) = 1 - \cos(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t), \quad \cos(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) = \frac{\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t}{||\boldsymbol{\mu}_k||_2 ||\boldsymbol{f}_j^t||_2}. \tag{4}$$

We also consider other transport costs and present the results in Section 4.2. Note that the inner expectation of the expected transport loss $\mathcal{L}_{t\to\mu}$ is analytical, so it is easy to obtain an unbiased estimation of $\mathcal{L}_{t\to\mu}$ with a mini-batch from the target domain $\mathcal{D}_t^{\boldsymbol{x}}$.

In this direction, we are assigning each target sample to the prototypes according to their similarities and the class distribution. Intuitively, minimizing the expected transport loss encourages each target feature to get closer to neighbor class prototypes, reducing the violation of the cluster assumption. If we think of each prototype as the mode of the distribution of source features for a class, this loss encourages a mode-seeking behavior. Still, this loss alone might lead to sub-optimal alignment. The feature encoder can map most of the target data to only a few prototypes. We connect this loss to entropy minimization to elucidate this point.

Connection with Entropy Minimization. The expected transport cost from the target domain to prototypes can be viewed as a generalization of the entropy minimization [22], an effective regularization in many prior domain-adaptation works [37, 38, 20]. If the transport cost is defined as $c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) = -\log p_{jk}^t = -\log \frac{\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{k'=1}^{K}\exp(\boldsymbol{\mu}_{k'}^T \boldsymbol{f}_j^t)}$ and the transport weight is $\pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t) = p_{jk}^t = \frac{\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{k'=1}^{K}\exp(\boldsymbol{\mu}_{k'}^T \boldsymbol{f}_j^t)}$ (with a uniform prior), then the transport loss becomes: $\mathcal{L}_{t\to\mu} = -\mathbb{E}_{\boldsymbol{x}_j^t \sim \mathcal{D}_t^{\boldsymbol{x}}} \left[\sum_{k=1}^{K} p_{jk}^t \log p_{jk}^t \right]$, which is equivalent to minimizing the entropy on the target samples. Entropy minimization alone also has a mode-seeking behavior and has the issue of dropping some modes (class prototypes). In other words, one trivial solution is to assign the same one-hot encoding to all the target examples [39, 40].

2.2.2 Transport from class prototypes to target domain

To ensure that each prototype has target features located close by and avoid the issue of mode dropping, we propose an additional transport cost from another direction, i.e., from prototypes to target features. Given a mini-batch, $\{\boldsymbol{x}_j^t\}_{j=1}^M$, of target samples of size M, the transport probabilities from a prototype $\boldsymbol{\mu}_k$ to the empirical distribution of the target features, $\hat{p}(\boldsymbol{f}^t) = \sum_{j=1}^M \frac{1}{M}\delta_{\boldsymbol{f}_j^t}$, is defined as the conditional probabilities:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_j^t \mid \boldsymbol{\mu}_k) = \frac{\hat{p}(\boldsymbol{f}_j^t)\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{j'=1}^M \hat{p}(\boldsymbol{f}_{j'}^t)\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_{j'}^t)} = \frac{\exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{j'=1}^M \exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_{j'}^t)}. \tag{5}$$

As opposed to the transport probabilities from target features to class prototypes $\pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t)$, $\pi_{\boldsymbol{\theta}}(\boldsymbol{f}_j^t|\boldsymbol{\mu}_k)$ normalizes the probabilities across the target samples for each prototype which ensures that each prototype will be assigned to some target features. Then, the expected transport cost in this direction is defined as:

$$\mathcal{L}_{\mu\to t} = \mathbb{E}_{\{\boldsymbol{x}_j^t\}_{j=1}^M \sim \mathcal{D}_t^{\boldsymbol{x}}} \mathbb{E}_{\boldsymbol{\mu}_k \sim p(\boldsymbol{\mu}_k)} \mathbb{E}_{\boldsymbol{f}_j^t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{f}_j^t \mid \boldsymbol{\mu}_k)} \left[c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) \right]. \tag{6}$$

As the two inner expectations are analytical, we can draw a mini-batch of target samples to estimate the expected transport cost.

Finally, our loss consists of two terms: the classification loss $\mathcal{L}_{\text{cls}}$ (Eq.1) and the bi-directional transport loss $\mathcal{L}_{\text{tls}}$:

$$\mathcal{L}_{\text{tls}} = \mathbb{E}_{\boldsymbol{x}_j^t \sim \mathcal{D}_t^{\boldsymbol{x}}} \mathbb{E}_{\boldsymbol{\mu}_k \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t)} \left[c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) \right] + \mathbb{E}_{\{\boldsymbol{x}_j^t\}_{j=1}^M \sim \mathcal{D}_t^{\boldsymbol{x}}} \mathbb{E}_{\boldsymbol{\mu}_k \sim p(\boldsymbol{\mu}_k)} \mathbb{E}_{\boldsymbol{f}_j^t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{f}_j^t|\boldsymbol{\mu}_k)} \left[c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) \right]. \tag{7}$$

Note that we treat $\boldsymbol{\mu}$ as fixed in $\mathcal{L}_{\text{tls}}$, and we do not back-propagate the gradient from $\mathcal{L}_{\text{tls}}$ to update $\boldsymbol{\mu}$ in this loss. We found empirically that this leads to more stable training. It also allows us to apply our method in the source-data-private setting where we only have access to the source model.

2.3 Learning class proportions in the target domain

We propose to infer the class proportions $\{p(\boldsymbol{\mu}_k)\}_{k=1}^K$ in the target domain by maximizing the log-likelihood of the unlabeled target data while fixing the class prototypes $\boldsymbol{\mu}$. Directly optimizing the marginal likelihood is intractable, so we use the EM algorithm [41–43] to derive the following iterative updates (see the derivation in Appendix A). We first initialize with a uniform prior: $p(\boldsymbol{\mu}_k)^0 = \frac{1}{K}$, and obtain new estimates at each update step l (starting from 0):

$$p(\boldsymbol{\mu}_k)^{l+1} = \frac{1}{M}\sum_{j=1}^M \pi_{\boldsymbol{\theta}}^l(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t), \quad \text{where} \quad \pi_{\boldsymbol{\theta}}^l(\boldsymbol{\mu}_k|\boldsymbol{f}_j^t) = \frac{p(\boldsymbol{\mu}_k)^l \exp(\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)}{\sum_{k'=1}^K p(\boldsymbol{\mu}_{k'})^l \exp(\boldsymbol{\mu}_{k'}^T \boldsymbol{f}_j^t)}. \tag{8}$$

Intuitively, the average predicted probabilities over target examples for each class are used to estimate the target proportions. However, in the early stages of the training, the classifier might not produce accurate estimates of the predictive probabilities. To alleviate this issue, we use a momentum update to stabilize the learning of class proportions: $p(\boldsymbol{\mu}_k)^{l+1} \leftarrow \beta p(\boldsymbol{\mu}_k)^l + (1-\beta)p(\boldsymbol{\mu}_k)^{l+1}$, where β controls the exponential decay rate.

3 Related work

Feature distribution alignment. Early works on domain adaptation with deep learning focus on feature alignment to align either the marginal distributions [44, 6, 11, 45] or the joint distributions [7, 8] of the deep features of neural networks. Most works use adversarial-based objectives, which is equivalent to minimizing the Jensen-Shannon divergence (JSD) [46]. When the source and target domains have non-overlapping supports, the JS divergence fails to supply useful gradients [47, 48]. To remedy this issue, researchers propose using the Wasserstein distance, which arises from the optimal transport problem [49]. Courty et al. [12] develop Joint Distribution Optimal Transport (JDOT) [12], defining the transport cost to be the joint cost in the data and prediction spaces. Several works [12, 15] extend this framework for deep neural networks. However, solving for the optimal couplings without any relaxation is a linear programming problem, which has a complexity of $\mathcal{O}(M^3 \log M)$ [50]. To compute the transport probabilities, our algorithm has the complexity of $\mathcal{O}(d_f MK)$, the complexity of computing the predictive probabilities. All of the works in this category focus on aligning the target features with source features, whereas we align the target features with prototypes.

Prototype-based alignment. We make two distinctions to existing works in this area. First, prior domain-adaptation works for classification [30, 51, 35] and segmentation [52, 53] utilize prototypes for pseudo-label assignments. By contrast, we use prototypes to behave as representative samples of the source features for the transport loss. Second, all of these works use some form of average latent features to construct class prototypes, which is computationally expensive. We instead adopt a parametric approach to learn the prototypes, avoiding the costly computation.

Learning under a shift in class proportions. Although a shift in class proportions between two domains is a common problem in many applications, it is still largely under-explored. Over the years, researchers have viewed the question through different lenses: applying kernel distribution embedding [54, 55, 23], using an EM update [41, 56], placing a meta-prior over the target proportions [57], and casting the problem under causal and anti-causal learning [23–27]. Recently, Tachet des Combes et al. [10] propose aligning the target feature distribution with a re-weighted version of the source feature distribution. While this method achieves consistent improvements over feature-alignment algorithms, it still relies on learning domain-invariant representations. As we have discussed, this approach suffers from the problems of sampling variability and class-mismatching in a mini-batch, whereas the proposed method uses class prototypes and prior estimation to avoid these issues.

Source-private adaptation. Finally, the proposed method can also be applied to a source-private setting where we only have access to the source model and target data while adapting to the target domain [35, 45, 58, 59]. Liang et al. [35] introduce a clustering-based approach to generate pseudo-labels for the target data. This approach requires constructing class centers using a weighted average of the latent features. Different from this work, our prototypes behave as class centers and is more amenable to mini-batch training.

4 Experiments

In this section, we evaluate our method under four practical settings: single-source, multi-source, class-imbalance, and source-private domain adaptation. We present the setup, the results, and the analysis of the results in the upcoming sections.

Datasets. We use the following three datasets of varying sizes in our experiment: **1)** *Office-31* [3] consists of 4652 images coming from three domains: Amazon (A), Webcam (W), and DSLR (D). The total number of categories is 31. **2)** *Office-Home* [60] is a more challenging dataset than Office-31 since it contains $15{,}500$ images over 65 classes and four domains: Artistic images (Ar), Clip art (Cl), Product images (Pr), and Real-world (Rw). **3)** *DomainNet* [61] is a large-scale dataset for domain adaptation. It contains about $569{,}010$ images with 345 categories from six domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch.

Implementation details. We implement our method on top of the open-source transfer learning library (MIT license) [62], adopting the default neural network architectures for both the feature encoder and linear classifier. For the feature encoder network, we utilize a pre-trained ResNet-50 in all experiments except for multi-source domain adaptation, where we use a pre-trained ResNet-101. We fine-tune the feature encoder and train the linear layers from random initialization. The linear layers have the learning rate of 0.01, 10 times that of the feature encoder. The learning rate is multiplied by η at every iteration where $\eta = \eta_0(1 + \gamma \text{iter})^{-\alpha}$. We set η_0 to 0.01, γ to 0.0002, and α to 0.75. We utilize a mini-batch SGD with a momentum of 0.9. We set the batch size for the source data as $N = 32$ and that for the target data as $M = 96$. We use all the labeled source samples and unlabeled target samples [6–8]. We set β to 1 (a uniform prior) in all settings except for the sub-sampled target datasets. We perform a sensitivity analysis (see Appendix C) and set β empirically to 0.999 for the sub-sampled target version of Office-31 and 0.9999 for that of Office-Home. We report the average accuracy from three independent runs. All experiments are conducted using a single Nvidia Tesla V100 GPU except for the DomainNet experiment, where we use 4 V100 GPUs. More implementation details can be found in Appendix B.

4.1 Main results

Single-source setting. We perform single-source domain adaptation on the Office-31 and Office-Home datasets. In each experiment, one domain serves as the source domain and another as the target domain. We consider all permutations, leading to 6 tasks for the Office-31 dataset and 12 tasks for

Table 1: Accuracy (%) on Office-31 for unsupervised domain adaptation (ResNet-50).

Category	Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
	ResNet-50 [63]	68.4 ± 0.2	96.7 ± 0.1	99.3 ± 0.1	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	76.1
Adversarial	DANN [6]	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
	ADDA [45]	86.2 ± 0.5	96.2 ± 0.3	98.4 ± 0.3	77.8 ± 0.3	69.5 ± 0.4	68.9 ± 0.5	82.9
	CDAN [8]	94.1 ± 0.1	98.6 ± 0.1	$\mathbf{100.0 \pm .0}$	92.9 ± 0.2	71.0 ± 0.3	69.3 ± 0.3	87.7
	MDD [64]	94.5 ± 0.3	98.4 ± 0.1	$\mathbf{100.0 \pm .0}$	93.5 ± 0.2	74.6 ± 0.3	72.2 ± 0.1	88.9
Divergence	JAN [7]	85.4 ± 0.3	97.4 ± 0.2	99.8 ± 0.2	84.7 ± 0.3	68.6 ± 0.3	70.0 ± 0.4	84.3
	TPN [30]	91.2 ± 0.3	97.7 ± 0.2	99.5 ± 0.1	89.9 ± 0.2	70.5 ± 0.2	73.5 ± 0.1	87.1
OT	DeepJDOT [13]	88.9 ± 0.3	98.5 ± 0.1	99.6 ± 0.2	88.2 ± 0.1	72.1 ± 0.4	70.1 ± 0.4	86.2
	ETD [65]	92.1	**100.0**	**100.0**	88.0	71.0	67.8	86.2
	PCT (Ours)	$\mathbf{94.6 \pm 0.5}$	98.7 ± 0.4	99.9 ± 0.1	$\mathbf{93.8 \pm 1.8}$	$\mathbf{77.2 \pm 0.5}$	$\mathbf{76.0 \pm 0.9}$	**90.0**

Table 2: Accuracy (%) on Office-Home for unsupervised domain adaptation (ResNet-50).

Method	Ar → Cl	Ar → Pr	Ar → Rw	Cl → Ar	Cl → Pr	Cl → Rw	Pr → Ar	Pr → Cl	Pr → Rw	Rw → Ar	Rw → Cl	Rw → Pr	Avg
ResNet-50 [63]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DANN [6]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
CDAN [8]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
MDD [64]	54.9	73.7	77.8	60.0	71.4	71.8	61.2	53.6	78.1	72.5	**60.2**	82.3	68.1
JAN [7]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
TPN [30]	51.2	71.2	76.0	65.1	72.9	72.8	55.4	48.9	76.5	70.9	53.4	80.4	66.2
DeepJDOT [13]	48.2	69.2	74.5	58.5	69.1	71.1	56.3	46.0	76.5	68.0	52.7	80.9	64.3
ETD [65]	51.3	71.9	**85.7**	57.6	69.2	73.7	57.8	51.2	79.3	70.2	57.5	82.1	67.3
PCT (Ours)	**57.1**	**78.3**	81.4	**67.6**	**77.0**	**76.5**	**68.0**	**55.0**	**81.3**	**74.7**	60.0	**85.3**	**71.8**

the Office-Home dataset. We report the results on Office-31 in Table 1.PCT significantly outperforms the baselines, especially on the more difficult transfer tasks (D→A and W → A). Although MDD–the best baseline domain-adaptation method–uses a bigger classifier, PCT still has 1.1% higher average accuracy. In Figure 2a, we visualize the number of parameters versus the average accuracy on the Office-31 dataset. While PCT uses fewer parameters than most methods, it still achieves the highest average accuracy. We report the average accuracies on the Office-Home dataset in Table 2. PCT outperforms baseline methods on 10 of the 12 transfer tasks, yielding 3.7% improvement on average accuracy over MDD. The results in this setting demonstrate that aligning target features with prototypes is more effective than aligning with source features.

Multi-source setting. In this setting, we evaluate our method using DomainNet [61], a large-scale dataset. For each task, we select one domain as the target domain and use the remaining five domains as the source. We use the same data splitting scheme as the original paper [61]. We adopt a single classifier and do not use domain labels. We compare against the standard benchmarks introduced in Peng et al. [61] and report the results in Table 3. While PCT is not designed specifically for multi-source domain adaptation, our method still outperforms those that utilize domain labels and a classifier for each source domain (reported under the mid-line of the table) on four of the six tasks. Since there are more variations of data in the source domain, we hypothesize that the increase in performance is because prototypes help mitigate the sampling variability problem.

Table 3: Accuracy $(\%)$ on DomainNet for ResNet101-based MSDA methods.

Models	$\mathcal{R}$ → Clipart	$\mathcal{R}$ → Infograph	$\mathcal{R}$ → Painting	$\mathcal{R}$ → Quickdraw	$\mathcal{R}$ → Real	$\mathcal{R}$ → Sketch	Avg
ResNet101 [24]	47.6 ± 0.5	13.0 ± 0.4	38.1 ± 0.5	13.3 ± 0.4	51.9 ± 0.9	33.7 ± 0.5	32.9 ± 0.5
DAN [11]	45.4 ± 0.5	12.8 ± 0.9	36.2 ± 0.6	15.3 ± 0.4	48.6 ± 0.7	34.0 ± 0.5	32.1 ± 0.6
RTN [66]	44.2 ± 0.6	12.6 ± 0.7	35.3 ± 0.6	14.6 ± 0.8	48.4 ± 0.7	31.7 ± 0.7	31.1 ± 0.7
JAN [7]	40.9 ± 0.4	11.1 ± 0.6	35.4 ± 0.5	12.1 ± 0.7	45.8 ± 0.6	32.3 ± 0.6	29.6 ± 0.6
DANN [6]	45.5 ± 0.6	13.1 ± 0.7	37.0 ± 0.7	13.2 ± 0.8	48.9 ± 0.7	31.8 ± 0.6	32.6 ± 0.7
ADDA [45]	47.5 ± 0.8	11.4 ± 0.7	36.7 ± 0.5	14.7 ± 0.5	49.1 ± 0.8	33.5 ± 0.5	32.2 ± 0.6
SE [36]	24.7 ± 0.3	3.9 ± 0.5	12.7 ± 0.4	7.1 ± 0.5	22.8 ± 0.5	9.1 ± 0.5	16.1 ± 0.4
MCD [67]	54.3 ± 0.6	22.1 ± 0.7	45.7 ± 0.6	7.6 ± 0.5	58.4 ± 0.7	43.5 ± 0.6	38.5 ± 0.6
DCTN [68]	48.6 ± 0.7	23.5 ± 0.6	48.8 ± 0.6	7.2 ± 0.4	53.5 ± 0.6	47.3 ± 0.5	38.2 ± 0.6
M ^{3} SDA [61]	57.2 ± 1.0	24.2 ± 1.2	51.6 ± 0.4	5.2 ± 0.5	61.6 ± 0.9	49.6 ± 0.6	41.5 ± 0.7
M ^{3} SDA- β [61]	58.6 ± 0.5	26.0 ± 0.9	52.3 ± 0.6	6.3 ± 0.6	62.7 ± 0.5	49.5 ± 0.8	42.6 ± 0.6
ML-MSDA [69]	61.4 ± 0.8	$\mathbf{26.2} \pm 0.4$	51.9 ± 0.2	$\mathbf{19.1} \pm 0.3$	57.0 ± 1.0	50.3 ± 0.7	44.3 ± 0.6
PCT (Ours)	$\mathbf{67.2 \pm 0.5}$	26.1 ± 0.2	$\mathbf{55.0 \pm 0.2}$	16.2 ± 0.2	$\mathbf{67.1 \pm 0.2}$	$\mathbf{53.7 \pm 0.6}$	**47.6**

Sub-sampled setting. In many cases, the label proportions could significantly change from one dataset to another, resulting in a decrease in the model's performance. To test our algorithm under this setting, we follow the experimental protocol in Tachet des Combes et al. [10]. We keep only thirty percent of the first $\lfloor K/2 \rfloor$ classes to simulate class imbalance. On the sub-sampled dataset, we directly take the results from [10] for the sub-sampled source data and perform additional experiments using

Table 4: Average accuracy (%) on the sub-sampled version of Office-31 and Office-Home (ResNet-50).

Method	sub-S O-31	sub-T O-31	sub-S O-H	sub-T O-H
ResNet-50 [63]	75.7	76.1	51.4	58.2
DANN [11]	76.2	75.9	51.8	57.3
JAN [7]	78.2	78.1	53.9	65.7
CDAN [8]	81.6	83.5	56.3	65.5
IWDAN [10]	82.6	79.2	57.6	N/A
IWJAN [10]	82.6	82.8	55.9	N/A
IWCDAN [10]	83.9	83.5	61.2	N/A
PCT-Uniform (Ours)	$\mathbf{87.9 \pm 0.4}$	85.4 ± 0.3	$\mathbf{67.8 \pm 0.3}$	69.8 ± 0.2
PCT-Learnable (Ours)	$\mathbf{87.9 \pm 0.4}$	$\mathbf{86.3 \pm 0.1}$	$\mathbf{67.8 \pm 0.3}$	$\mathbf{70.2 \pm 0.2}$

Table 5: Average Accuracy (%) on the source-private Office-31 for unsupervised domain adaptation (ResNet-50).

Method	Source Model Only	SHOT-Pseudo Label [35]	SHOT-IM [35]	SHOT [35]	PCT(Ours)
Avg. Accuracy	79.3	87.6 ± 0.5	87.3 ± 0.5	$\mathbf{88.6 \pm 0.4}$	88.4 ± 0.6

the same sub-sampling scheme on the target data. The baselines in this setting are standard domain adaptation methods (DAN, JAN, and CDAN) and their importance weighted versions introduced in [10]. We present the results in Table 4. On the sub-sampled source data, PCT outperforms the second-best method (IWCDAN) by 4% on Office-31 and 6.6% on Office-Home. On the sub-sampled target data, PCT with a uniform prior still outperforms the baseline, 2.8% higher average accuracy than IWCDAN's. Using a learnable prior consistently improves upon using a uniform prior by 0.9% on Office-31 and by 0.4% on Office-Home. The improvements confirm our intuition that prototypes help with the class imbalance in both the source and target domain while estimating the target proportion further benefit the target domain. We visualize the estimated proportions on the target data in Appendix C, confirming our intuition that the proportions are inferred correctly.

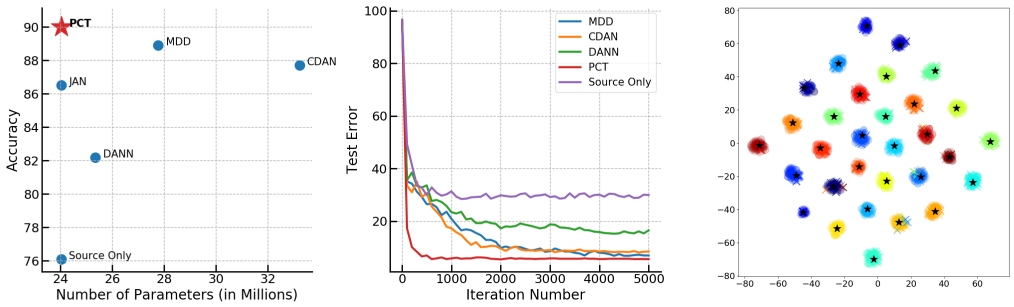
Source-data-private setting. In many practical applications, practitioners might not directly have access to the source data. Instead, a trained model on the source data is provided. In this setting, the goal is to adapt to the target domain while only operating on the given model. We compare our method with Source Hypothesis Transfer (SHOT) [35], a state-of-the-art method proposed specifically for this setting. SHOT contains two losses: an information maximization (IM) loss and a pseudo-labeling loss. We follow their experimental protocol by first training the model on the source data alone. During the adaptation stage, we only use the target data to perform model adaptation. We use the objective function in Equation 7 to update the feature encoder while fixing the prototypes. We report the result in Table 5. It is not surprising that the average accuracy drops by 1.6% from the standard setting where we have access to source data in Table 1. Notably, our transport loss outperforms the IM loss by 1.1% and the pseudo label loss by 0.8%. The pseudo-labeling loss in SHOT also requires constructing class centers, which does not scale well with large datasets. Our approach uses class prototypes to avoid this issue. While the average accuracy for our method is 0.2% lower than both of their losses combined, the p-value for the independent two-sample t-test on the sample mean is 0.32, which is not statistically significant.

4.2 Analysis of results

Ablation study. To examine the effect of each component in our framework, we perform ablation studies and present the results in Table 6. **1)** *Effect of each loss.* We examine the effect of each loss on the average test accuracy on the Office-31 dataset. We remove each transport loss while keeping the cross-entropy loss. The bi-directional transport loss leads to the best accuracy, while the drop in accuracy is more significant if we remove $\mathcal{L}_{\mu \to t}$. This result is not surprising because, without $\mathcal{L}_{\mu \to t}$, the model can map target data to only a few prototypes, leading to a degenerate solution. **2)** *Divergence choice.* Next, we present the result using optimal transport as the divergence. We consider two variants of Prototype-oriented Optimal Transport (POT): exact linear program (POT) and Sinkhorn relaxation (POT-Sinkhorn). In each variant, we solve for the optimal couplings using the optimal transport formulation. After obtaining the transport probabilities, we update the feature encoder using the obtained probabilities as the weights for the transport cost. We can see that

Table 6: Average accuracy (%) of PCT on Office-31 under different variants (ResNet-50).

PCT w/o ($\mathcal{L}_{t\to\mu}$)	PCT w/o ($\mathcal{L}_{\mu\to t}$)	POT	POT-Sinkhorn	w/o stop-grad	$c(\boldsymbol{\mu}_k, \boldsymbol{f}_j^t) = \exp(-\boldsymbol{\mu}_k^T \boldsymbol{f}_j^t)$	PCT (Ours)
88.7	85.2	86.95	87.6	89.2	87.0	90.0



(a) (b) (c)

Figure 2: Analysis of parameter efficiency, convergence, and t-SNE visualization.

POT performs better than DeepJDOT in Table 1, showing the effectiveness of using prototypes to define the transport costs. Still, both versions of POT significantly underperforms PCT. **3)** *Gradient stopping.* We also show the algorithm's performance without stopping the gradient of $\boldsymbol{\mu}$ in the transport loss. PCT gains an additional 0.8% in accuracy with the gradient stopping strategy. **4)** *Cost function.* Finally, we explore the cost function inspired by the radial basis kernel. We can see that the cosine distance in PCT gives 3% higher average accuracy. In short, the choice of transport cost, gradient-stopping strategy, and both transport losses all contribute to the success of our method.

Convergence comparison. We plot test accuracy versus iteration number on the task (A $\to$ W) in Figure 2b to compare the convergence rate of different algorithms. PCT quickly converges within the first one thousand iterations, and the test accuracy does not fluctuate thereafter. This phenomenon is not surprising since we use prototypes instead of the source features to align with target features. We expect prototypes to behave as representative samples of the source features, making the model converge quickly.

Visualization. We visualize the t-SNE plot of the source and target features as well as the prototypes in Figure 2c. Figure 2c shows that the prototypes (black stars) are close to both the source (dots) and target (crosses) features, illustrating that the target features are not close to the decision boundaries. The visualization indicates that our algorithm successfully adapts the model to the target data.

5 Broader Impact

Any methods that deal with classification can suffer from dataset bias caused by domain shift between training and testing data. While domain adaptation can help mitigate the problem [70], it cannot eliminate the issue because of the combinatorial nature of too many exogenous factors [71]. Also, as with any computationally intensive endeavor, care must be taken to use sustainable energy sources.

6 Conclusion

We offer a holistic framework for unsupervised domain adaptation through the lens of probabilistic transport. Via extensive experiments, we show that our algorithm works well on multiple datasets, is robust against class imbalance, and can perform domain adaptation without accessing source data directly. Without adding additional networks, our algorithm achieves competitive performance with state-of-the-art methods on several benchmarks.

References

[1] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *European conference on computer vision*, pages 154–166. Springer, 2008.

[2] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[3] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.

[5] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010.

[6] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1180–1189. JMLR.org, 2015.

[7] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2208–2217. PMLR, 06–11 Aug 2017.

[8] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[9] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019.

[10] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain adaptation with conditional distribution matching and generalized label shift. In *Advances in Neural Information Processing Systems*, 2020.

[11] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 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*, pages 97–105, 2015.

[12] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. *CoRR*, abs/1803.10081, 2018.

[14] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10285–10295, 2019.

[15] Renjun Xu, Pelen Liu, Liyan Wang, Chao Chen, and Jindong Wang. Reliable weighted optimal transport for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[17] Leonid V Kantorovich. On the translocation of masses. *Journal of Mathematical Sciences*, 133 (4):1381–1382, 2006.

[18] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[19] Matthieu Lerasle, Zoltán Szabó, Timothée Mathieu, and Guillaume Lecué. Monk outlier-robust mean embedding estimation by median-of-means. In *International Conference on Machine Learning*, pages 3782–3793. PMLR, 2019.

[20] Yogesh Balaji, Rama Chellappa, and Soheil Feizi. Robust optimal transport with applications in generative modeling and domain adaptation. In *Advances in Neural Information Processing Systems*, 2020.

[21] Fredrik D. Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 527–536, 16–18 Apr 2019.

[22] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. In *CAP*, pages 281–296, 2005.

[23] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827. PMLR, 2013.

[24] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

[25] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pages 2839–2848. PMLR, 2016.

[26] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[27] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*, 2019.

[28] Huangjie Zheng and Mingyuan Zhou. Comparing probability distributions with conditional transport. *arXiv preprint arXiv:2012.14100*, 2020.

[29] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[30] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[32] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[33] Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. *arXiv preprint arXiv:1206.6438*, 2012.

[34] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[35] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

[36] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*, 2017.

[37] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2517–2526, 2019.

[38] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8050–8058, 2019.

[39] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *arXiv preprint arXiv:1711.10288*, 2017.

[40] Xiaofu Wu, Quan Zhou, Zhen Yang, Chunming Zhao, Longin Jan Latecki, et al. Entropy minimization vs. diversity maximization for domain adaptation. *arXiv preprint arXiv:2002.01690*, 2020.

[41] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

[42] Guoliang Kang, Liang Zheng, Yan Yan, and Yi Yang. Deep adversarial attention alignment for unsupervised domain adaptation: the benefit of target expectation maximization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.

[43] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *International Conference on Machine Learning*, pages 222–232. PMLR, 2020.

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

[45] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[46] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[47] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[48] Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

[49] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[50] Quentin Mérigot and Edouard Oudet. Discrete optimal transport: complexity, geometry and applications. *Discrete & Computational Geometry*, 55(2):263–283, 2016.

[51] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019.

[52] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. *arXiv preprint arXiv:1910.13049*, 2019.

[53] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International conference on machine learning*, pages 5423–5432. PMLR, 2018.

[54] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50:110–119, 2014.

[55] Tuan Duong Nguyen, Marthinus Christoffel, and Masashi Sugiyama. Continuous target shift adaptation in supervised learning. In *Asian Conference on Machine Learning*, pages 285–300. PMLR, 2016.

[56] Yee Seng Chan and Hwee Tou Ng. Word sense disambiguation with distribution estimation. In *IJCAI*, volume 5, pages 1010–5. Citeseer, 2005.

[57] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28, 2009.

[58] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020.

[59] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 615–625, 2021.

[60] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[61] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[62] Mingsheng Long Junguang Jiang, Bo Fu. Transfer-learning-library. `https://github.com/thuml/Transfer-Learning-Library`, 2020.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[64] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413. PMLR, 2019.

[65] Mengxue Li, Yi-Ming Zhai, You-Wei Luo, Peng-Fei Ge, and Chuan-Xian Ren. Enhanced transport distance for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13936–13944, 2020.

[66] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *arXiv preprint arXiv:1602.04433*, 2016.

[67] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *CoRR*, abs/1712.02560, 2017.

[68] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3964–3973, 2018.

[69] Zhenpeng Li, Zhen Zhao, Yuhong Guo, Haifeng Shen, and Jieping Ye. Mutual learning network for multi-source domain adaptation. *arXiv preprint arXiv:2003.12944*, 2020.

[70] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer, 2017.

[71] Boqing Gong, Fei Sha, and Kristen Grauman. Overcoming dataset bias: An unsupervised domain adaptation approach. In *NIPS Workshop on Large Scale Visual Recognition and Retrieval*, volume 3. Citeseer, 2012.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 4.1
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4 adn Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4
   (b) Did you mention the license of the assets? [Yes] See Section 4
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is included in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]