# Implicit Task-Driven Probability Discrepancy Measure for Unsupervised Domain Adaptation

**Mao Li, Kaiqi Jiang, Xinhua Zhang**
Department of Computer Science, University of Illinois at Chicago
Chicago, IL 60607
{mli206,kjiang10,zhangx}@uic.edu

## Abstract

Probability discrepancy measure is a fundamental construct for numerous machine learning models such as weakly supervised learning and generative modeling. However, most measures overlook the fact that the distributions are not the end-product of learning, but are the input of a downstream predictor. Therefore, it is important to warp the probability discrepancy measure towards the end tasks, and towards this goal, we propose a new bi-level optimization based approach so that the two distributions are compared not uniformly against the entire hypothesis space, but only with respect to the optimal predictor for the downstream end task. When applied to margin disparity discrepancy and contrastive domain discrepancy, our method significantly improves the performance in unsupervised domain adaptation, and enjoys a much more principled training process.

## 1 Introduction

Discrepancy measures on two distributions underpin a large variety of machine learning tasks, and have been studied extensively since the dawn of modern probability [1]. For example, in generative models, such a measure is applied to align the generated distribution with the empirical one, and prevalent examples include 1) the $f$-divergence that admits a convenient variational form hence can be effectively evaluated via sample-based adversarial optimization [2, 3]; 2) integral probability metric [IPM, 4] that seeks the largest discrepancy in function expectation over a reproducing kernel Hilbert space (RKHS) [MMD GAN, 5–7], 1-Lipschitz continuous functions [Wasserstein GAN, 8, 9], or unit $L_2$ norm functions [Fisher GAN, 10], etc.

In domain adaptation [DA, 11, 12], probability discrepancy is also the key construct in the feature adaptation approach, where a feature extractor $\phi$ is sought to align the source and target distributions transformed by $\phi$ [13, 14]. The aforementioned measures can be applied directly in this context.

It has been long noted that the discrepancy should be tailored to the function class of interest, e.g., those for which we would like to compute expectations. This principle has been applied to density estimation [15] amongst others, where the RKHS is selected to match the downstream task such as image categorization based on the compressed pixel distribution. Naturally this motivation can be easily implemented in IPMs by customizing the generating function space.

However such tailoring remains oblivious to the loss and available labels of the end task. Intuitively, if the latent features in DA are to be used for classification, then whether the loss is AUC or F-score should ideally influence the probability discrepancy. The seminal $\mathcal{H}\Delta\mathcal{H}$-divergence is designed for classification accuracy [16], with a few extensions to Bayesian and other losses [17–20]. Despite being data-dependent, however, they are *unsupervised* without accounting for the available labels. Likewise, if a generative model is used to augment data so as to improve segmentation accuracy [21], then the adversarial network in GANs should not only be able to distinguish between real and synthetic, but also "align", in an appropriate sense, with the segmentation labels at hand.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Warping probability discrepancies towards a task has been lightly touched in unsupervised DA (UDA). [22] trains two classifiers that not only boost the source-domain accuracy, but also maximally disagree on the target domain. Unfortunately, it is only formulated as a procedure, *not* a probability discrepancy. Most relevant to our work is the margin disparity discrepancy [MDD, 23], which is based on the $\mathcal{H}\Delta\mathcal{H}$-divergence where two fictitious classifiers $h$ and $h'$ are jointly optimized to maximally reveal the two distributions' difference. [23] took the key insight that $h$ can be tied with the source-domain predictor, and can thus be optimized to simultaneously reduce the source-domain risk and the $\mathcal{H}\Delta\mathcal{H}$-divergence. However, in spite of its effectiveness in both theory and practice, we discover that the specific formulation conflicts with the $\mathcal{H}\Delta\mathcal{H}$-divergence — the latter tries to *maximize* over $h$ so as to promote the divergence, while MDD tries to *minimize* it (Section 3). This undermines the power of MDD in distinguishing two distributions as illustrated in Figure 1. Flipping the sign and min/max cannot resolve the issue.

Our first contribution, therefore, is to develop a new task-driven discrepancy framework that overcomes this obstacle. The key inspiration is that MDD relies on the *pseudo-label* in the target domain (i.e., speculation of their labels based, e.g., on the source-domain head), and this is also the case for some other measures such as the contrastive domain discrepancy [CDD, 24], which promotes the proximity between the class mean of the two domains for each class, and pushes apart the mean of different classes. Such a commonality motivates us to generate the target-domain pseudo-label based on the *optimal* source-domain classifier $h^*$. In MDD, this provides a natural substitute for the fictitious classifier $h$ (Section 3.2) which *no longer needs to be optimized over*, thereby solving the aforementioned problem. As our second contribution, we extend this strategy to CDD in Section 4. The overall formulation becomes a bi-level optimization solvable by implicit differentiation (hence the modifier "implicit" in the method's name).

We note in passing that pseudo-label is commonly used in self-training for UDA [25–28]. However, most methods require various refinements of it in order to mitigate its inaccuracy due to distributional shift [29]. Examples include label sharpening [30], entropy reweighting [31], cycle training [29]. We instead directly use the output of $f^*$ as the pseudo-label in probability discrepancy, outperforming state of the art on a range of datasets (Section 6).

UDA has recently received considerable interest, and most algorithms rely on ad-hoc heuristics; we will mention a few below. Many of them require perusing the code and configuration script. As such, our main goal is *not* to develop yet another highly engineered model that performs better, but to present a *principled* formulation solvable by *off-the-shelf* optimizers. Although our implicit task-driven discrepancy can be straightforwardly applied to generative models, we deem it a better use of space to fully demonstrate its power in UDA. Such a probability discrepancy can also be easily extended to measure (conditional) independence, which has witnessed immediate application in fair and disentangled representation learning [32–35].

## 2 Preliminaries

In UDA, there is a source domain and a target domain, and they are respectively represented as a joint distribution $S$ and $T$ on an input-output space $\mathcal{X} \times \mathcal{Y}$. We will denote their marginal distributions via subscripts, e.g., $S_x$ and $T_y$. The $\mathcal{Y}$ domain can be multiclass with labels $[C] := \{1, 2, \dots, C\}$. We are provided with labeled examples in the source domain, denoted as an empirical distribution $\tilde{S}$. On the target domain, however, we can only access unlabeled examples, i.e., an empirical distribution $\tilde{T}_x$ which only encompasses the input part of an empirical distribution $\tilde{T}$. In short, let the empirical distributions consist of $\{x_i^s, y_i^s\}_{i=1}^{n_s}$ and $\{x_j^t\}_{j=1}^{n_t}$ for the source and target domains respectively.

The goal of UDA is to find a classifier that predicts well on the target domain $T$. This is often referred to as inductive learning, while, in contrast, transductive learning is only concerned with the prediction on the empirical distribution $\tilde{T}$, whose feature component $\tilde{T}_x$ is available at training time.

The classification model, shared by both domains, consists of a feature extractor (e.g., ResNet) parameterized by $\phi$ and a head $h_\theta$ parameterized by $\theta$. Letting $\ell$ be the loss over the ground-truth label $y$ and the prediction $h_\theta(\phi(x))$, we seek the $\phi$ and $\theta$ that minimize the target-domain risk

$$\mathbb{E}_{(x,y)\sim T}\, \ell(y, h_\theta(\phi(x))), \quad \text{or its empirical counterpart} \quad \mathbb{E}_{(x,y)\sim \tilde{T}}\, \ell(y, h_\theta(\phi(x))). \tag{1}$$

In order to leverage the labeled data from the source domain and the unlabeled target-domain data, the feature adaption approach enforces low empirical risk on the source domain (thanks to the availability

of labels there) and encourages that the source domain distribution, after being transformed by the feature extractor $\phi$, "aligns" well with that of the target domain [13, 14, 36, 37]. This is achieved by

$$\min_{\phi,\theta} \mathbb{E}_{(x,y)\sim\tilde{S}} \ell(y, h_\theta(\phi(x))) + d(\phi\#\tilde{S}_x, \phi\#\tilde{T}_x), \tag{2}$$

where $\phi\#\tilde{S}_x$ is the pushforward distribution of $\tilde{S}_x$, and $d$ denotes some discrepancy measure between two distributions. The intuition is that by "mixing" the latent distributions across the two domains through $\phi$, the favorable accuracy of $h_\theta$ on the source domain can be transferred to the target domain. For simplicity, we will denote $P := \phi\#S_x$ and $\tilde{P} := \phi\#\tilde{S}_x$, and explicitize its dependency on $\phi$ by writing $P_\phi$ whenever necessary. With $z = \phi(x)$, we can derive a conditional distribution of $y$ given $z$ based on $S$, and we denote it as $S_{y|z}$. Similarly, let $Q := \phi\#T_x$, $\tilde{Q} := \phi\#\tilde{T}_x$, and define $T_{y|z}$ analogously.

## 3 Implicit Task-Driven Margin Disparity Discrepancy

There has been a plethora of research on sample-based discrepancy measure between two distributions. Examples include maximum mean discrepancy [MMD, 38], and (neural) variational optimization [39] which effectively subsumes a number of adversarial learning based measures [2, 14].

However, these methods are oblivious to the subsequent tasks that are based on $P$ and $Q$. For example, UDA can be aimed to classify well on these distributions. In domain-adversarial neural networks [DANN, 14], the discrepancy between $P$ and $Q$ is measured via the Jensen-Shannon divergence, reformulated in an adversarial objective as in the generative adversarial network [GAN, 40]. Moreover, MMD simply measures the largest possible difference in the function expectation over $P$ and $Q$:

$$\mathrm{MMD}(P, Q) := \sup_{f\in\mathcal{H}:\|f\|_{\mathcal{H}}\le 1} \left[ \mathop{\mathbb{E}}_{x\sim P} f(x) - \mathop{\mathbb{E}}_{x\sim Q} f(x) \right] = \left\| \mathop{\mathbb{E}}_{x\sim P} k(x,\cdot) - \mathop{\mathbb{E}}_{x\sim Q} k(x,\cdot) \right\|_{\mathcal{H}}, \tag{3}$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space (RKHS) induced by a kernel $k$. Obviously, it does not take into account whether $f$ is used for classification or regression. The celebrated $\mathcal{H}\Delta\mathcal{H}$-divergence addresses this problem by focusing on binary classification [16, Lemma 3]:

$$d_{\mathcal{H}\Delta\mathcal{H}}(P, Q) := \max_{h\in\mathcal{H}} \max_{h'\in\mathcal{H}} \mathcal{D}(h, h', P, Q), \tag{4}$$

$$\text{where} \quad \mathcal{D}(h, h', P, Q) := \left| \mathbb{E}_P[\![\operatorname{sign}\circ h' \neq \operatorname{sign}\circ h]\!] - \mathbb{E}_Q[\![\operatorname{sign}\circ h' \neq \operatorname{sign}\circ h]\!] \right|. \tag{5}$$

Here $\operatorname{sign}\circ h$ applies the sign function on the output of $h$. $\mathcal{H}$ is a hypothesis space (not necessarily an RKHS), and $[\![\cdot]\!]$ is the Iverson bracket that evaluates to 1 if $\cdot$ is true, and 0 otherwise. However, it still does not concern the label of the data (i.e., align only in an unsupervised fashion). To warp the measure to the end-task in a data-dependent fashion, [23] proposed the margin disparity discrepancy (MDD), which improved upon [22] by formulating a principled objective function instead of a heuristic procedure. According to Equation 24 of [23], MDD employs

$$d_{\mathrm{MDD}}(P, Q) = \min_{h\in\mathcal{H}} \left\{ \mathcal{R}(h; P) + \max_{h'\in\mathcal{H}} \mathcal{D}(h, h', P, Q) \right\}, \tag{6}$$

$$\text{where} \quad \mathcal{R}(h; P) := \mathbb{E}_{z\sim P,\, y|z\sim S_{y|z}} \ell(h(z), y) + \mathrm{reg}(h) \quad \text{is the regularized risk,} \tag{7}$$

and the 0-1 loss in $\mathcal{D}$ can be replaced by smooth surrogates such as hinge loss or cross-entropy loss. Here reg is any standard regularizer applied in regularized risk minimization, e.g., $\ell_2$ norm. The underlying insight is that when comparing $P$ and $Q$, one only needs to consider those $h$ that predict well on the (labeled) source domain, while leaving $h'$ to reveal the maximum discrepancy between $P$ and $Q$. Similar ideas have been leveraged in [22, 41].

### 3.1 Conflict between MDD and $\mathcal{H}\Delta\mathcal{H}$-divergence

Unfortunately, $d_{\mathrm{MDD}}$ turns out conflicting with the spirit of $\mathcal{H}\Delta\mathcal{H}$-divergence in an important way. Note that $h$ is *maximized* in $\mathcal{D}$ as in (4), while it is *minimized* in $d_{\mathrm{MDD}}$ as in (6). This raises a natural question: can the distribution discrepancy be sufficiently revealed when $\max_h$ is replaced by $\min_h$ in the definition of $\mathcal{D}$, i.e.,

$$d^{\min}_{\mathcal{H}\Delta\mathcal{H}}(P, Q) := \min_{h\in\mathcal{H}} \max_{h'\in\mathcal{H}} \mathcal{D}(h, h', P, Q). \tag{8}$$

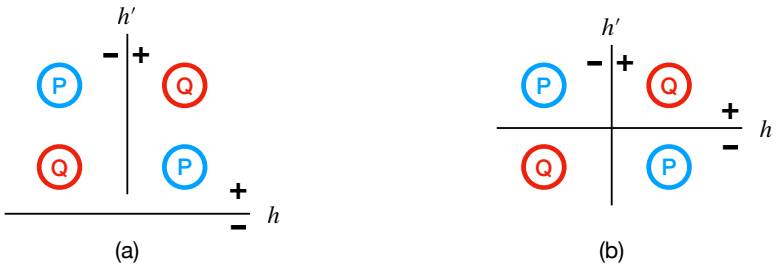


Figure 1: An example showing that changing $\max_h$ into $\min_h$ undercuts the power of discriminating two distributions. Here the source distribution $P$ has two blue clusters, and the target distribution $Q$ consists of two red clusters. The location of $h$ in (a) makes $\max_{h' \in \mathcal{H}} \mathcal{D}(h, h', P, Q) = 0$, meaning that the new discrepancy $d^{\min}_{\mathcal{H}\Delta\mathcal{H}}(P, Q)$ cannot distinguish the two distributions. In contrast, the $h$ in (b) makes $\max_{h' \in \mathcal{H}} \mathcal{D}(h, h', P, Q) = 1$, implying that the original $d_{\mathcal{H}\Delta\mathcal{H}}(P, Q)$ can distinguish.

It turns out such a change does undermine the discriminative power, and an example is illustrated in Figure 1. Here both the source and target domains have two separate clusters, and the hypothesis space is the horizontal or vertical half spaces (i.e., decision stumps). Sub-figure (a) shows that the minimum $h$ in $d^{\min}_{\mathcal{H}\Delta\mathcal{H}}$ is attained at the horizontal line, and it is easy to check that no matter where $h'$ is placed, $\mathcal{D}(h, h', P, Q) = 0$. In contrast, the $h$ and $h'$ shown in (b) attain $\mathcal{D}(h, h', P, Q) = 1$. So changing maximization of $h$ into minimization caused significant loss in the discrimination power. A more detailed discussion in conjunction with $\mathcal{R}$ as in (6) is available in Appendix A.

### 3.2 A new implicit task-driven MDD

Flipping back the optimization of $h$ turns out far more involved that it appears. It cannot be achieved by simply changing $\min_h$ into $\max_h$ in (6) with the source domain risk negated:

$$\max_{h \in \mathcal{H}} \Big\{ -\mathcal{R}(h; P) + \max_{h' \in \mathcal{H}} \mathcal{D}(h, h', P, Q) \Big\}, \tag{9}$$

This is because $P$ and $Q$ indeed depend on the feature extractor $\phi$. If we next minimize $d_{\text{MDD}}(P, Q)$ over $\phi$, then it implicitly promotes the source domain risk. If $d_{\text{MDD}}(P, Q)$ is instead maximized over $\phi$, then $\phi$ would attempt to increase the discrepancy $\mathcal{D}$. With a few trials, it becomes clear that the same issue persists in other combinations of flipping sign or min/max.

Our first contribution, hence, is to resolve this issue by turning $d_{\text{MDD}}$ into a *constrained* formulation:

$$\max_{h \in \mathcal{H}: \mathcal{R}(h;P) \le \lambda} \max_{h' \in \mathcal{H}} \mathcal{D}(h, h', P, Q), \tag{10}$$

where $\lambda$ is some pre-specified cap of loss. Constraining the performance of a classifier is quite commonly used in, e.g., gradient episodic memory to combat catastrophic forgetting [GEM, 42, 43]. However, GEM only solves a linear approximation instead of the exact problem, and it is arguably difficult to *differentiate through* for optimizing $\phi$ in (10). Therefore, we finally develop a bi-level optimization by setting $h$ to the optimal one for the source domain, and then using it in the discrepancy measure. We call it $i$-MDD because it will rely on implicit differentiation for training. The overall training objective can be written as:[1]

$$\min_{\phi} d_{i\text{-MDD}}(\tilde{P}_\phi, \tilde{Q}_\phi) + \alpha \mathcal{R}(h^*; \tilde{P}_\phi) \quad \text{where} \quad d_{i\text{-MDD}}(\tilde{P}_\phi, \tilde{Q}_\phi) := \max_{h' \in \mathcal{H}} \mathcal{D}(h^*, h', \tilde{P}_\phi, \tilde{Q}_\phi), \tag{11}$$

$$h^* := \arg\min_{h \in \mathcal{H}} \mathcal{R}(h; \tilde{P}_\phi). \tag{12}$$

Here $\alpha > 0$ is a tradeoff parameter. If we do not include $\mathcal{R}(h; \tilde{P}_\phi)$ in the objective, then the feature $\phi$ would receive no incentive to reduce the source-domain risk. This term in the objective function does not necessitate new implicit differentiation, because $h^*$ is exactly the minimizer of $\mathcal{R}(h; \tilde{P}_\phi)$. The architecture of $i$-MDD is shown in Figure 2, in comparison with MDD.

[1] One might wonder why the max over $h$ in (10) is turned into min over $h$ in (12). This is because $\lambda$ is set to $\min_h \mathcal{R}(h; P)$. Analogously, maximizing $f(x, y)$ over $(x-1)^2 \le 0$ is equivalent to evaluating $f(1, y)$, because 0 is the minimum of $(x-1)^2$ attained at $x = 1$. Or to bear more resemblance to (11) and (12), it is equal to $f(x^*, y)$ where $x^* = \text{argmin}_x (x-1)^2$.

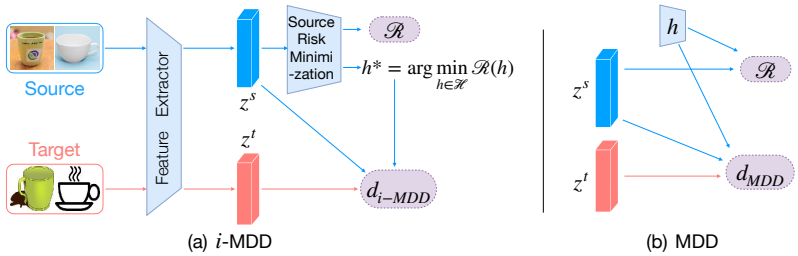


Figure 2: Illustration of $i$-MDD and MDD. The $h^*$, fed into $d_{i\text{-MDD}}$ in $i$-MDD, is the minimizer of $\mathcal{R}$.

### 3.3 Practical discussions: differentiable surrogates

Since the 0-1 loss in $\mathcal{D}$ is not amenable to differentiable training, we follow [23] to morph it into the cross-entropy loss (CE). In particular, suppose $h$ outputs a $C$ dimensional logit vector, and $p = \text{softmax}(h)$. Similarly, $p' = \text{softmax}(h')$. Then the standard $\text{CE}(p', p) = -\sum_i p_i \log p'_i \geq 0$. To combat exploding or vanishing gradient, [3] proposed a modified CE: $\text{MCE}(p', p) = \sum_i p_i \log(1 - p'_i) \leq 0$. Then [23] adopts the approximation

$$\mathcal{D}(h, h', \tilde{P}_\phi, \tilde{Q}_\phi) \approx \mathbb{E}_{\tilde{Q}_\phi} \text{MCE}(p', ind \circ p) - \gamma \mathbb{E}_{\tilde{P}_\phi} \text{CE}(p', ind \circ p), \quad (\gamma > 0) \tag{13}$$

where $ind : \mathbb{R}^C \to \{0, 1\}^C$ is the indicator function mapping a vector $v$ to the $i^*$-th canonical vector with $i^* = \arg\max_i v_i$. In practice, the formulation has two issues. First, the right-hand side of (13) is unbounded from below, making it possible for $\phi$ (the minimizing variable) to push it to the negative infinity when solved by stochastic saddle-point optimization. As a result, the implementation of [23] tuned the step size delicately. Secondly, the indicator function $ind$ blocks the backpropagation through the branch of $h$, jeopardizing the proper optimization. We tried removing the indicator function but observed negative infinity even after finely tuning the step size.

In contrast, our new $i$-MDD is immune to these issues, where (13) is used without including the indicator function. In our experiment, we observed that the head $h_\theta$ only needs to be linear in order to achieve state-of-the-art performance. This provided considerable convenience because the optimization for $h^*$ in (12) can be accomplished very efficiently with high precision by convex solvers such as LIBLINEAR [44]. Similarly, it is clear that $h^*$ does *not* depend on $h'$, but on $\tilde{P}_\phi$ only (i.e., $\phi$). Therefore, we can first solve $h^*$ in (12), and then fix it when solving $h'$ in (11), which results in another convex problem thanks to the linearity of $h'$. These conveniences significantly benefit computation and convergence properties.

Although MDD can forgo the stochastic saddle-point optimization and also evaluate $d_{\text{MDD}}$ exactly, the inner *joint* maximization over $h$ and $h'$ leads to a non-concave function, hence impairing the precision of backpropagation. Even if the indicator function is imposed and optimization is only over $h'$, we found a linear $h'$ was insufficient to deliver accurate predictions.

### 3.4 Bi-level optimization

Bi-level optimization has recently received intensive study [45–48], and they can be easily applied to $i$-MDD. Thanks to the linearity of $h$ and $h'$, the backpropagation can be performed in a *closed form*. Denote the ultimate objective value in (11) as $J$. Letting $z_i^s = \phi(x_i^s)$ and $z_j^t = \phi(x_j^t)$, we only need to derive new strategies to compute $\partial J / \partial z_i^s$ and $\partial J / \partial z_j^t$, based on which backpropagation through the feature extractor will be standard. Towards this end, most of the implicit differentiation approaches rely on multiplying a given vector to the Hessian of the loss $\ell$ in (12) with respect to $h$ [45]. Interestingly, for linear multi-class classifiers with cross-entropy loss, the formula has already been derived by [49, Appendix D], and we quote their results in Appendix B for completeness, along with the detailed analysis of computational complexity.

To summarize, the crux of $i$-MDD is to replace the $h$ in the $\mathcal{H}\Delta\mathcal{H}$-divergence by the optimal source domain classifier $h^*$ under the current $\phi$. This is in line with the pseudo-label approach and $h^*$ can be applied to the target domain to provide a soft label. Indeed this principle can be applied to other class-aware discrepancy measures, and our next contribution is to warp the contrastive domain discrepancy [CDD, 24] towards the end task.

## 4 Task-driven Contrastive Domain Discrepancy

Underpinning CDD is the hard pseudo-label $\hat{y}_j^t \in [C]$ assigned to each target domain example $z_j^t$. [24] adopted clustering on $z_j^t$, where each class corresponds to a cluster, and its center is initialized by the mean of the source domain $z_i^s$. Naturally, $\hat{y}_j^t$ is set to the cluster that $z_j^t$ belongs to at convergence. Then the discrepancy between $\tilde{P}$ and $\tilde{Q}$ is defined as (distilled from Equations 3 and 4 in [24])

$$d_{\text{CDD}}(\tilde{P}, \tilde{Q}) = \underbrace{\frac{1}{C} \sum_{c \in [C]} \left\| \mu_c^s - \mu_c^t \right\|_{\mathcal{H}}^2}_{\text{intra-class discrepancy}} - \beta \cdot \underbrace{\frac{1}{C(C-1)} \sum_{c \neq c'} \left\| \mu_c^s - \mu_{c'}^t \right\|_{\mathcal{H}}^2}_{\text{inter-class discrepancy}}, \tag{14}$$

$$\text{where} \quad \mu_c^s := \text{mean}\{k(z_i^s, \cdot) : i \in [n_s] \text{ and } y_i^s = c\}, \quad \forall\, c \in [C] \tag{15}$$

$$\mu_c^t := \text{mean}\{k(z_j^t, \cdot) : j \in [n_t] \text{ and } \hat{y}_j^t = c\}, \quad \forall\, c \in [C]. \tag{16}$$

Here $\beta > 0$ is a tradeoff coeffient. The underlying motivation is to align the class-wise center between source and target domains (the intra-class discrepancy), and push apart the centers of different classes (the inter-class discrepancy). Although the source-domain label is used to initialize clustering, the prediction head $h$ is not involved in $d_{\text{CDD}}$, hence not sufficiently driven by the end task.

In addition, a number of heuristics are required for CDD to perform well. Firstly, after clustering, only the target-domain examples that are close to the center are included to compute the mean $\mu_c^t$. This introduces one hyperparameter to tune. Secondly, domain specific batch-normalization is required. Finally, the bandwidth of the RBF kernel needs to be learned for *each pair* of $(c, c')$ in the implementation. To remove **all** these nuisances and formulate a *principled* optimization, we next warp CDD towards tasks based on bi-level optimization.

### 4.1 Implicit task-driven CDD

Our key insight is that the head $h^*$ in (12) constitutes a natural source of pseudo-label that is superior to clustering. Firstly, $h^*$ is uniquely determined thanks to the convexity originating from the linearity of $h$. Moreover, clustering is a "procedure" which is not amenable to differentiation despite some recent progress in reversible learning [46]. In contrast, differentiation through $h^*$ is straightforward as discussed above.

This intuition can be directly implemented by redefining the class centers in the target domain based on the $h^*$-induced soft pseudo-label for each example $z_j^t$. Recall $h^*(z_j^t)$ produces the $C$-dimensional logit (unnormalized score) for the $C$ classes, and the softmax of it yields a $C$-dimensional probability vector $p_j^t$, whose $c$-th element encodes the probability of belonging to class $c$. Accordingly, we can morph the target-domain center $\mu_c^t$ into

$$\mu_c^t(h) := \sum_{j=1}^{n_t} (p_j^t)_c \, z_j^t \Bigg/ \left( 10^{-6} + \sum_{j=1}^{n_t} (p_j^t)_c \right), \quad \text{where} \quad p_j^t = \text{softmax}(h(z_j^t)) \in \mathbb{R}^C. \tag{17}$$

Note the kernel $k$ is removed and we directly used $z_j^t$. We also added a small smoothing factor $10^{-6}$ in case all examples are unlikely to belong to class $c$. To summarize, our training objective is

$$\min_{\phi} d_{i\text{-CDD}}(\tilde{P}_\phi, \tilde{Q}_\phi) + \alpha \mathcal{R}(h^*; \tilde{P}_\phi) \tag{18}$$

$$\text{where} \quad d_{i\text{-CDD}}(\tilde{P}_\phi, \tilde{Q}_\phi) := \frac{1}{C} \sum_{c \in [C]} \left\| \mu_c^s - \mu_c^t(h^*) \right\|_{\mathcal{H}}^2 - \beta \frac{1}{C(C-1)} \sum_{c \neq c'} \| \mu_c^s - \mu_{c'}^s \|_{\mathcal{H}}^2 \tag{19}$$

$$h^* := \arg\min_{h \in \mathcal{H}} \mathcal{R}(h; \tilde{P}_\phi). \tag{20}$$

It is clearly identical to $i$-MDD in (11) except that the $d_{i\text{-MDD}}$ is replaced by $d_{i\text{-CDD}}$. Compared with $d_{\text{CDD}}$ in (14), we slightly changed the inter-class term from between source and target domains $(\mu_c^s - \mu_{c'}^t)$ into within source domain only $(\mu_c^s - \mu_{c'}^s)$. This simplifies optimization because the centers of the source domain do not depend on $h^*$. Meanwhile, different classes are still pushed apart in *both* domains because 1) it is enforced on the source domain, and 2) the source domain centers $\mu_c^s$ are aligned with those of the target domain $\mu_c^t(h^*)$. Backpropagation and bi-level optimization are similar to $i$-MDD, with even reduced complexity as no optimization (over $h'$) is involved in $d_{i\text{-CDD}}$.

### 4.2 Cache-augmented training

It was noted in [24] that the limited size of mini-batch may leave only a small number of examples for each class (or even none), especially when there are many classes. This hampers the computation of class means. They thus resorted to a class-aware sampling strategy where only a subset of classes are picked at each iteration, and samples are drawn only for *these* classes. This again relies on the result of clustering for the target domain, exacerbating the fallout of not backpropagating through it.

To address this issue, we followed [50, 51] by caching the latent representations $z$ in the most recent iterations via a circular queue for each class. This allows the class means to be computed more accurately, and the backpropagation is still conducted only on the current mini-batch examples.

We emphasize that our overall optimization remains principled even with cache augmentation, an observation that has not been made in literature to the best of our knowledge. Since $\phi$ is updated with a small step size and only a small number of latest iterations are cached, the continuity of the algorithm ensures that the $z$ computed from a stale $\phi$ is still close to the value if it *were* computed with the latest $\phi$. As a result, the bias of the gradient can be bounded linearly by the step size times the staleness (i.e., the length of the queue / mini-batch size). We relegate the details to Appendix C.

## 5 Related Works in Unsupervised Domain Adaptation via Feature Adaptation

Although our motivation is to develop a task-driven probability discrepancy measure while UDA is used only as an example application, we would also like to place our approach in the context of UDA literature. A detailed and recent survey is available in [52], and we will only focus on one category of methods that are most related to our approach, namely feature adaptation based methods. These methods seek feature extractors so that the source and target domains are aligned in the feature space, hence called domain-invariant feature representations [53]. Although methods may differ in whether different domains share the feature extractors in part, in whole, or none, the most prominent variation lies in the alignment metric.

Conventional probability discrepancy measures include Jensen-Shannon divergence used by GAN and DANN, and Wasserstein distance [54, 55]. The MMD in (3) has multiple variants such as multiple kernels [13] and joint MMD [56]. When the kernel is not universal, e.g., polynomial, MMD essentially compares the statistics such as the variance (second-order), and various comparison metrics have been studied [e.g., 57, 58]. CDD further accounts for the source-domain label information (but not the target-domain head) via the intra-class and inter-class distances.

Several adversarial methods try to align the domains by demoting the features' discriminative power in identifying the domain. The idea can be traced back to at least GAN, and example variants include DANN and [31, 59–61].

## 6 Experimental Results

We finally validate the implicit task-driven discrepancy by comparing $i$-MDD and $i$-CDD against state-of-the-art methods for unsupervised domain adaptation, especially MDD and CDD. Ablation studies will also be carried out to examine the influence of various components. More details on the experiment setup and results are available in Appendix D.

### 6.1 Comparison of target-domain accuracy

**Datasets.** We adopted three public domain datasets for UDA benchmarking.

- **Office-31** [62] is a standard dataset for real-world domain adaptation. It consists of 4,652 images belonging to 31 unbalanced classes. These images are collected from three distinct domains: **A**mazon (from Amazon website), **W**ebcam (from web camera) and **D**SLR (by digital SLR camera).
- **Office-Home** [63] is a more challenging dataset for visual domain adaptation. It contains 15,500 images of daily objects in office or home environment, belonging to 65 categories. The images are sampled from four domains: **Ar**tistic images, **Cl**ip Art, **Pr**oduct images, and **R**eal-**w**orld images.

- **ImageCLEF-DA** [64] consists of images from three domains: **C**altech-256, **I**mageNet ILSVRC 2012 and **P**ascal VOC 2012. Each domain has 12 categories and each class contains 50 images.

**Baselines.** We compared our $i$-MDD and $i$-CDD with the following state-of-the-art UDA methods: Deep Adaptation Networks **(DAN)** [13], Domain Adversarial Neural Network **(DANN)** [14], Residual Transfer Network **(RTN)** [65], Joint Adaptation Networks **(JAN)** [64], the Entropy Conditioning Variant of Conditional Domain Adversarial Network **(CDAN+E)** [31], Multi-Adversarial Domain Adaptation **(MADA)** [66], Conditional Domain Adversarial Network with Batch Spectral Penalization **(BSP+CDAN)** [67], **CDD** [24] (which named it Contrastive Adaptation Network), Cluster Alignment with a Teacher with Robust Gradient Reversal **(rRevGrad+CAT)** [68], **MDD** [23], MDD with Implicit Alignment **(MDD+IA)** [69], and Adversarial Spectral Adaptation Network **(ASAN)** [70].

We also considered a variant of CDD (named **vCDD**) where $\mu_c^s - \mu_{c'}^t$ is replaced by $\mu_c^s - \mu_{c'}^s$ in source domain *only*, and the class-aware sampling in [24] is replaced by cache augmentation. This allows us to compare $i$-CDD with the exact counterpart that does not use bi-level optimization.

Additional comparisons with some state-of-the-art methods that are *not* based on feature adaptation are available in Appendix D.3.

**Implementation details.** We followed the commonly used experimental protocol for unsupervised domain adaptation from [14]. We report the average accuracy and standard deviation of five independent runs. For $i$-MDD we mainly used the hyper-parameters from [23], i.e., the margin factor $\gamma$ in (13) was chosen from $\{2, 3, 4\}$ and was kept the same for all tasks on the same dataset. For $i$-CDD, the trade-off coefficient $\beta$ between intra-class loss and inter-class loss in (14) is chosen from $\{0.1, 0.01, 0.001\}$. The cache size for each class is 30.

We implemented our methods in PyTorch. The head classifier (in both $i$-CDD and $i$-MDD) and the auxiliary classifier ($h'$ in $i$-MDD) are both 1-layer neural network with width 1024. We did not restrict MDD and CDD to single-layer $h$ or $h'$.

For optimization, we used mini-batch SGD with Nesterov momentum 0.9. The initial learning rate was 0.004, which was adjusted according to [14]. The mini-batch size is 150 for each domain. More detailed explanation of hyper-parameter selection is presented in the supplementary materials, along with the sensitivity analysis of them. ResNet-50 pretrained on ImageNet was used as the feature extractor in all methods. Since our aim is to improve the probability discrepancy measure for UDA, we employed the standard backbone ResNet-50 instead of integrating heavier-weight feature extractors, ad-hoc engineering heuristics, or generic feature improvements [e.g., 71].

**Results.** The accuracy of target-domain prediction is presented in Table 1 for Office-31, Table 2 for Office-Home, and Table 3 for ImageCLEF. Clearly $i$-CDD achieves the highest average accuracy among all methods over all datasets. As we zoom into each pair of domain, it is also either the best performer or close to the best. Secondly, by comparing vCDD with $i$-CDD and MDD with $i$-MDD, it is clear that the implicit (i.e., bi-level) formulation can significantly boost the performance upon the standard joint optimization, except $i$-MDD on ImageCLEF where it is a tie. This validates our original motivation. Thirdly, vCDD outperforms CDD on two datasets and ties on Office-31, implying that computing the inter-class discrepancy based solely on the source domain is superior to that based on both source and target domains. This makes sense because ground-truth labels are only available for the source, and the pseudo-labels for the target domain can be noisy and detrimental.

Overall, $i$-CDD is superior to $i$-MDD. This makes sense because $i$-CDD not only matches the center of each class between source and target, but also promotes the inter-class discrepancy, i.e., pushing apart the center of different classes. The latter "contrastive" component appears quite beneficial.

Finally, MDD+IA can often outperform MDD, and although $i$-MDD achieves significantly higher accuracy than MDD+IA on Office-31, it is less competitive on the other two datasets. This does not invalidate our implicit task-driven principle, and we can implicitize MDD+IA for future work.

### 6.2 Ablation study

We next examine the influence of several important components of $i$-MDD and $i$-CDD, including the cache size (queue length) in $i$-CDD, the dimensionality of hidden representation, $i$-CDD equipped with the class-aware sampling [24]. All the ablation studies were conducted on Ar:Cl in Office-Home.

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

| Method | A → W | D → W | W → D | A → D | D → A | W → A | Avg |
|---|---|---|---|---|---|---|---|
| ResNet-50 | 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 |
| DAN | 80.5 ± 0.4 | 97.1 ± 0.2 | 99.6 ± 0.1 | 78.6 ± 0.2 | 63.6 ± 0.3 | 62.8 ± 0.2 | 80.4 |
| DANN | 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 |
| RTN | 84.5 ± 0.2 | 96.8 ± 0.1 | 99.4 ± 0.1 | 77.5 ± 0.3 | 66.2 ± 0.2 | 64.8 ± 0.3 | 81.6 |
| JAN | 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 |
| CDAN+E | 94.1 ± 0.1 | 98.6 ± 0.1 | **100.0** ± 0.0 | 92.9 ± 0.2 | 71.0 ± 0.3 | 69.3 ± 0.3 | 87.7 |
| MADA | 90.0 ± 0.1 | 97.4 ± 0.1 | 99.6 ± 0.1 | 87.8 ± 0.2 | 70.3 ± 0.3 | 66.4 ± 0.3 | 85.2 |
| BSP+CDAN | 93.3 ± 0.2 | 98.2 ± 0.2 | **100.0** ± 0.0 | 93.0 ± 0.2 | 73.6 ± 0.3 | 72.6 ± 0.3 | 88.5 |
| CDD | 94.5 ± 0.3 | **99.1** ± 0.2 | 99.8 ± 0.2 | 95.0 ± 0.3 | **78.0** ± 0.3 | 77.0 ± 0.3 | 90.6 |
| rRevGrad+CAT | 94.4 ± 0.1 | 98.0 ± 0.2 | **100.0** ± 0.0 | 90.8 ± 1.8 | 72.2 ± 0.2 | 70.2 ± 0.1 | 87.6 |
| MDD | 94.5 ± 0.3 | 98.4 ± 0.1 | 100.0 ± 0.0 | 93.5 ± 0.2 | 74.6 ± 0.3 | 72.2 ± 0.1 | 88.9 |
| MDD+IA | 90.3 ± 0.2 | 98.7 ± 0.1 | 99.8 ± 0.0 | 92.1 ± 0.5 | 75.3 ± 0.2 | 74.9 ± 0.3 | 88.8 |
| ASAN | **95.6** ± 0.4 | 98.8 ± 0.2 | **100.0** ± 0.0 | 94.4 ± 0.9 | 74.7 ± 0.3 | 74.0 ± 0.9 | 90.0 |
| vCDD | 95.1 ± 0.7 | 98.4 ± 0.3 | 99.5 ± 0.3 | 94.8 ± 0.7 | 76.2 ± 0.5 | 76.9 ± 0.6 | 90.6 |
| $i$-CDD | 95.4 ± 0.4 | 98.5 ± 0.2 | **100.0** ± 0.0 | **96.3** ± 0.3 | 77.2 ± 0.3 | **78.3** ± 0.2 | **90.9** |
| $i$-MDD | 94.8 ± 0.5 | 98.4 ± 0.3 | **100.0** ± 0.0 | 94.2 ± 0.5 | 75.1 ± 0.5 | 74.1 ± 0.7 | 89.4 |

Table 2: Accuracy (%) on Office-Home for unsupervised domain adaptation (based on 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 | 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 |
| DAN | 43.6 | 57.0 | 67.9 | 45.8 | 56.5 | 60.4 | 44.0 | 43.6 | 67.7 | 63.1 | 51.5 | 74.3 | 56.3 |
| DANN | 45.6 | 59.3 | 70.1 | 47.0 | 58.5 | 60.9 | 46.1 | 43.7 | 68.5 | 63.2 | 51.8 | 76.8 | 57.6 |
| JAN | 45.9 | 61.2 | 68.9 | 50.4 | 59.7 | 61.0 | 45.8 | 43.4 | 70.3 | 63.9 | 52.4 | 76.8 | 58.3 |
| CDAN+E | 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 |
| BSP+CDAN | 52.0 | 68.6 | 76.1 | 58.0 | 70.3 | 70.2 | 58.6 | 50.2 | 77.6 | 72.2 | 59.3 | 81.9 | 66.3 |
| CDD | 51.6 | 71.2 | 76.7 | 59.8 | 70.8 | 70.8 | 59.8 | 49.9 | 77.4 | 70.6 | 58.8 | 80.5 | 66.5 |
| MDD | 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 |
| MDD+IA | 56.2 | **77.9** | **79.2** | **64.4** | 73.1 | **74.4** | 64.2 | 54.2 | **79.9** | 71.2 | 58.1 | 83.1 | 69.5 |
| ASAN | 53.6 | 73.0 | 77.0 | 62.1 | 73.9 | 72.6 | 61.6 | 52.8 | 79.8 | 73.3 | 60.2 | **83.6** | 68.6 |
| vCDD | 56.2 | 74.2 | 77.0 | 62.4 | 72.3 | 71.4 | 61.7 | 61.4 | 78.7 | 71.3 | 60.6 | 81.7 | 69.3 |
| $i$-CDD | **60.8** | 77.5 | 78.8 | **64.3** | **74.3** | 73.4 | **65.3** | **61.9** | 78.7 | 72.1 | **61.8** | 81.8 | **70.8** |
| $i$-MDD | 56.5 | 74.7 | 78.3 | 61.9 | 72.4 | 72.3 | 63.2 | 55.6 | 78.4 | 71.4 | 59.7 | 81.7 | 68.8 |

**Impact of cache size in $i$-CDD and vCDD.** Figure 3 shows the fluctuation of prediction accuracy for vCDD and $i$-CDD. The accuracy first grows when the length of the queue for each class increases from 1 to 30, corroborating the benefit of cache in improving the accuracy of center means. But then it starts to decay, suggesting that the stale samples accrued start to hurt.

Since there is a large number of class compared with the mini-batch size, the cluster mean cannot be estimated accurately. For example, Ar:Cl of Office-Home has 65 classes while the GPU memory limited our mini-batch size to 150. The cache augments the pool of latent feature values, hence improving the mean estimation. However, an overly large queue size may leave the stored values stale, i.e., inconsistent with the true value if it *were* computed from the current ResNet $\phi$.

Empirically, we found it generally effective to set the cache size to around 50% of the data size (number of images) of each domain. For example, there are about 2000 images in the Amazon website domain of Office-31, and we set the queue length to 30 for each of the 31 classes. This amounted to a cache size of $30 \times 31 = 930$ images, which is about half of 2000. It well balanced the sample size with staleness, and cost only a small amount of memory and computation thanks to the low dimensionality of the latent feature space.

**Impact of latent dimensionality.** Figure 4 demonstrates the prediction accuracy of vCDD and $i$-CDD, when the dimensionality of latent feature ($z_i^s$ and $z_j^t$) is varied in $\{128, 256, 512, 1024\}$.

Table 3: Accuracy (%) on ImageCLEF for unsupervised domain adaptation (based on ResNet-50)

| Method | I → P | P → I | I → C | C → I | C → P | P → C | Avg |
|---|---|---|---|---|---|---|---|
| ResNet-50 | 74.8 ± 0.3 | 83.9 ± 0.1 | 91.5 ± 0.3 | 78.0 ± 0.2 | 65.5 ± 0.3 | 91.2 ± 0.3 | 80.7 |
| DAN | 74.5 ± 0.4 | 82.2 ± 0.2 | 92.8 ± 0.2 | 86.3 ± 0.4 | 69.2 ± 0.4 | 89.8 ± 0.4 | 82.5 |
| DANN | 75.0 ± 0.6 | 86.0 ± 0.3 | 96.2 ± 0.4 | 87.0 ± 0.5 | 74.3 ± 0.5 | 91.5 ± 0.6 | 85.0 |
| RTN | 75.6 ± 0.3 | 86.8 ± 0.1 | 95.3 ± 0.1 | 86.9 ± 0.3 | 72.7 ± 0.3 | 92.2 ± 0.4 | 84.9 |
| JAN | 76.8 ± 0.4 | 88.0 ± 0.2 | 94.7 ± 0.2 | 89.5 ± 0.3 | 74.2 ± 0.3 | 91.7 ± 0.3 | 85.8 |
| CDAN+E | 77.7 ± 0.3 | 90.7 ± 0.2 | **97.7** ± 0.3 | 91.3 ± 0.3 | 74.2 ± 0.2 | 94.3 ± 0.3 | 87.7 |
| MADA | 75.0 ± 0.3 | 87.9 ± 0.2 | 96.0 ± 0.3 | 88.8 ± 0.3 | 75.2 ± 0.2 | 92.2 ± 0.3 | 85.8 |
| CDD | 77.0 ± 0.5 | 89.4 ± 0.3 | 97.2 ± 0.3 | 91.5 ± 0.2 | 76.2 ± 0.5 | 95.6 ± 0.6 | 87.8 |
| rRevGrad+CAT | 77.2 ± 0.2 | 91.0 ± 0.3 | 95.5 ± 0.3 | 91.3 ± 0.3 | 75.3 ± 0.6 | 93.6 ± 0.5 | 87.3 |
| MDD | 78.5 ± 0.2 | 91.1 ± 0.4 | 97.0 ± 0.2 | **92.1** ± 0.4 | 77.6 ± 0.3 | 93.8 ± 0.4 | 88.4 |
| MDD+IA | 78.3 ± 0.2 | 91.8 ± 0.2 | 96.7 ± 0.3 | 93.0 ± 0.2 | **79.0** ± 0.3 | 94.2 ± 0.2 | 88.8 |
| ASAN | 78.9 ± 0.4 | 92.3 ± 0.5 | 97.4 ± 0.5 | **92.1** ± 0.3 | 76.4 ± 0.7 | 94.4 ± 0.2 | 88.6 |
| vCDD | 78.8 ± 0.4 | 92.1 ± 0.1 | 97.0 ± 0.3 | 91.3 ± 0.3 | 78.2 ± 0.3 | 96.2 ± 0.4 | 88.9 |
| $i$-CDD | **79.8** ± 0.4 | **92.6** ± 0.3 | 97.2 ± 0.4 | **92.0** ± 0.3 | 78.6 ± 0.3 | **96.5** ± 0.2 | **89.4** |
| $i$-MDD | 78.5 ± 0.6 | 91.6 ± 0.5 | 96.5 ± 0.4 | 91.4 ± 0.3 | 76.8 ± 0.6 | 95.4 ± 0.3 | 88.4 |

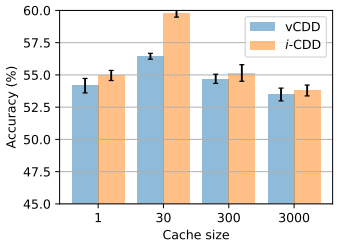


Figure 3: Accuracy v.s. cache size *for each class*

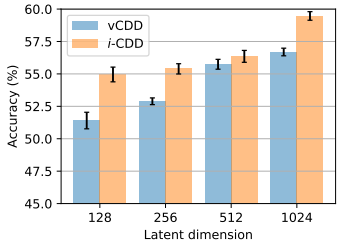


Figure 4: Accuracy v.s. latent dimensionality

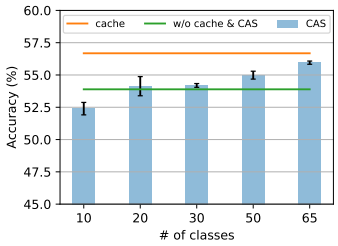


Figure 5: Class-aware sampling v.s. cache augmentation

Evidently, increasing the dimensionality tends to improve the accuracy for both methods, but at the cost of more computation.

**Class-aware sampling v.s. cache augmentation.** The problem of low sample for each class in a mini-batch (ref Section 4.2) was addressed by [24] via class-aware sampling (CAS), where a small number of classes (e.g., 10) are randomly selected, and a mini-batch only draws samples from these classes. Essentially, each iteration is based only on a subset of classes, while our $i$-CDD and vCDD still allow all classes to participate via cache augmentation. It is therefore of interest to compare CAS with cache. As shown in Figure 5, vCDD using CAS enjoys a monotonic growth of accuracy as more and more classes are involved in each iteration. When all the 65 classes are used, CAS gets close to cache augmentation. Without cache or CAS, the performance is lower (green line). This partly explains the success of vCDD, which is later improved further by $i$-CDD via the bi-level formulation.

Additional ablations studies are available in Appendix D.4, including the impact of batch size and standard deviations.

## 7 Conclusion

In this paper, we proposed warping probability discrepancy measures towards the end tasks by leveraging the pseudo-labels produced by the optimal predictor. Application to unsupervised domain adaptation significantly outperformed the state of the art in prediction accuracy, and the training is formulated as a principled optimization problem solvable by standard optimizers. For future work, it will be interesting to extend this technique to warping (conditional) independence measures, and to apply to structured and dynamic settings.

## Acknowledgements

We thank the reviewers for their constructive comments. This work is supported by NSF grant RI:1910146 and NIH grant R01CA258827.

## References

[1] S. T. Rachev. *Probability metrics and the stability of stochastic models*. Wiley, Chichester, 1991.

[2] S. Nowozin, B. Cseke, and R. Tomioka. $f$-GAN: training generative neural samplers using variational divergence minimization. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

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

[4] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[5] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. MMD GAN: towards deeper understanding of moment matching network. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[6] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. *In International Conference on Machine Learning (ICML)*. 2015.

[7] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *In Conference on Uncertainty in Artificial Intelligence (UAI)*. 2015.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. *In* D. Precup and Y. W. Teh, eds., *Proceedings of the 34th international conference on machine learning*, vol. 70 of *Proceedings of machine learning research*, pp. 214–223. PMLR, Sydney, Australia, Aug 2017.

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[10] Y. Mroueh and T. Sercu. Fisher GAN. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[11] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, eds. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2008.

[12] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. A survey on transfer learning. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[13] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *In International Conference on Machine Learning (ICML)*. 2015.

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[15] L. Song, X. Zhang, A. Smola, A. Gretton, and B. Schölkopf. Tailoring density estimation via reproducing kernel moment matching. *In International Conference on Machine Learning (ICML)*. 2008.

[16] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman Vaughan. A theory of learning from different domains. *Machine Learning Journal*, 72(1-2):151–175, 2010.

[17] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *In Conference on Computational Learning Theory (COLT)*. 2009.

[18] M. Mohri and A. M. Medina. New analysis and algorithm for learning with drifting distributions. *In Conference on Computational Learning Theory (COLT)*. 2012.

[19] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. A PAC-Bayesian approach for domain adaptation with specialization to linear classifiers. *In International Conference on Machine Learning (ICML)*. 2013.

[20] C. Cortes, M. Mohri, and A. M. Medina. Adaptation algorithm and theory based on generalized discrepancy. *In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2015.

[21] V. Sandfort, K. Yan, P. Pickhardt, and R. Summers. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. *Scientific Reports*, 9, 2019.

[22] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[23] Y. Zhang, T. Liu, M. Long, and M. Jordan. Bridging theory and algorithm for domain adaptation. *In International Conference on Machine Learning (ICML)*. 2019.

[24] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[25] D. hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *In Workshop on challenges in representation learning, ICML*. 2013.

[26] A. Kumar, T. Ma, and P. Liang. Understanding self-training for gradual domain adaptation. *In International Conference on Machine Learning (ICML)*. 2020.

[27] V. Prabhu, S. Khare, D. Kartik, and J. Hoffman. SENTRY: Selective entropy optimization via committee consistency for unsupervised domain adaptation. *arXiv:2012.11460*, 2020.

[28] A. Mey and M. Loog. A soft-labeled self-training approach. *In Proc. Intl. Conf. Pattern Recognition*. 2016.

[29] H. Liu, J. Wang, and M. Long. Cycle self-training for domain adaptation. *arXiv:2103.03571*, 2021.

[30] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[31] M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[32] N. Quadrianto, V. Sharmanska, and O. Thomas. Discovering fair representations in the data domain. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[33] E. Adeli, Q. Zhao, A. Pfefferbaum, E. V. Sullivan, L. Fei-Fei, J. C. Niebles, and K. M. Pohl. Representation learning with statistical independence to mitigate bias. *In IEEE Winter Applications of Computer Visions (WACV)*. 2021.

[34] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schoelkopf, and O. Bachem. On the fairness of disentangled representations. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[35] Y. Atzmon, F. Kreuk, U. Shalit, and G. Chechik. A causal view of compositional zero-shot recognition. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[36] B. Li, Y. Wang, T. Che, S. Zhang, S. Zhao, P. Xu, W. Zhou, Y. Bengio, and K. Keutzer. Rethinking distributional matching based domain adaptation. *arXiv:2006.13352*, 2020.

[37] F. D. Johansson, D. Sontag, and R. Ranganath. Support and invertibility in domain-invariant representations. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[38] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schoelkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[39] N. Wan, D. Li, and N. Hovakimyan. $f$-divergence variational inference. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

[41] B. Gholami, P. Sahu, M. Kim, and V. Pavlovic. Task-discriminative domain alignment for unsupervised domain adaptation. *In 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019.

[42] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[43] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with A-GEM. *In International Conference on Learning Representations (ICLR)*. 2019.

[44] R.-E. Fan, J.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, Aug 2008.

[45] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[46] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization. *In International Conference on Machine Learning (ICML)*. 2017.

[47] S. Jenni and P. Favaro. Deep bilevel learning. *In European Conference on Computer Vision (ECCV)*. 2018.

[48] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine. Meta-learning with implicit gradients. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[49] Y. Yu, X. Zhang, and D. Schuurmans. Generalized conditional gradient for sparse estimation. *arXiv:1410.4828*, 2014.

[50] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[51] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang. Joint detection and identification feature learning for person search. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[52] G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Trans. Intell. Syst. Technol.*, 11(5):1–46, 2020.

[53] H. Zhao, R. T. Des Combes, K. Zhang, and G. Gordon. On learning invariant representations for domain adaptation. *In International Conference on Machine Learning (ICML)*. 2019.

[54] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[55] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. *In European Conference on Computer Vision (ECCV)*. 2018.

[56] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. *In International Conference on Machine Learning (ICML)*. 2017.

[57] P. Morerio, J. Cavazza, and V. Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *In International Conference on Learning Representations (ICLR)*. 2018.

[58] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *In National Conference of Artificial Intelligence (AAAI)*. 2020.

[59] S. Purushotham, W. Carvalho, T. Nilanon, and Y. Liu. Variational adversarial deep domain adaptation for health care time series analysis. *In International Conference on Learning Representations (ICLR)*. 2017.

[60] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. *In International Conference on Computer Vision (ICCV)*. 2015.

[61] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[62] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *In Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, p. 213–226. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 364215560X.

[63] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[64] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. *In Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 2208–2217. PMLR, 06–11 Aug 2017.

[65] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *In Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[66] Z. Pei, Z. Cao, M. Long, and J. Wang. Multi-adversarial domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, Apr 2018.

[67] X. Chen, S. Wang, M. Long, and J. Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. *In Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 1081–1090. PMLR, 09–15 Jun 2019.

[68] Z. Deng, Y. Luo, and J. Zhu. Cluster alignment with a teacher for unsupervised domain adaptation. *In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. October 2019.

[69] X. Jiang, Q. Lao, S. Matwin, and M. Havaei. Implicit class-conditioned domain alignment for unsupervised domain adaptation. *In Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 4816–4827. PMLR, 13–18 Jul 2020.

[70] C. Raab, P. Vath, P. Meier, and F.-M. Schleif. Bridging adversarial and statistical domain transfer via spectral adaptation networks. *In Proceedings of the Asian Conference on Computer Vision (ACCV)*. November 2020.

[71] X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan. Transferable normalization: Towards improving transferability of deep neural networks. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[72] J. Liang, D. Hu, and J. Feng. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. *In International Conference on Machine Learning (ICML)*. 2020.

[73] Q. Wang and T. P. Breckon. Unsupervised domain adaptation via structured prediction based selective pseudo-labeling. *In National Conference of Artificial Intelligence (AAAI)*. 2020.

[74] J. Wang, J. Chen, J. Lin, L. Sigal, and C. W. de Silva. Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by Gaussian-guided latent alignment. *Pattern Recognition*, p. 107943, 2021.

# Supplementary Material

**All code and data are available at**

https://www.dropbox.com/sh/8e2enu3mwl7oxwk/AAAT8_xqkyLzLMqxqFH6tTjWa?dl=0

## A Example Comparing $d_{\text{MDD}}$ and $d_{i\text{-MDD}}$ in Conjunction with $\mathcal{R}$

We now compare $d_{\text{MDD}}(P, Q)$ and $d_{i\text{-MDD}}(P, Q)$ in the context of $\mathcal{R}$. To be self-contained, we copy their definitions from (6) and (11) to here:

$$d_{\text{MDD}}(P, Q) := \min_{h \in \mathcal{H}} \{\mathcal{D}(h) + \lambda \mathcal{R}(h)\}, \tag{21}$$

$$\text{where} \quad \mathcal{D}(h) := \max_{h' \in \mathcal{H}} \mathcal{D}(h, h', P, Q), \qquad \mathcal{R}(h) := \mathop{\mathbb{E}}_{(z,y) \in P} \ell(h(z), y) + \text{reg}(h). \tag{22}$$

And

$$d_{i\text{-MDD}}(P, Q) := \mathcal{D}(h^*), \quad \text{where} \quad h^* := \arg\min_{h \in \mathcal{H}} \mathcal{R}(h). \tag{23}$$

Here for simplicity, we abused the symbol $\mathcal{D}$ in (22) by maximizing out $h'$ in the original $\mathcal{D}$. No confusion will arise because the input argument clearly distinguishes the meaning. We also kept the dependency on $P$ and $Q$ implicit in all terms. The tradeoff weight $\lambda$ is not the one in (10).

**Case 1**: $\lambda$ is small. In this case, $d_{\text{MDD}}$ places a low weight on fitting the source-domain data, which differs substantially from the motivation of $d_{i\text{-MDD}}$. This is obviously not a good choice, and in general, MDD does not operate in this regime.

**Case 2**: $\lambda$ is large. This appears to make $d_{\text{MDD}}$ close to $d_{i\text{-MDD}}$ because the large value of $\lambda$ will push $h$ to focus on minimizing $\mathcal{R}$, which is consistent with the definition of $h^*$ in $d_{i\text{-MDD}}$. However, with large $\lambda$, the value of $\lambda\mathcal{R}(h)$ under the optimal $h$ for $\mathcal{D}(h) + \lambda\mathcal{R}(h)$ can get very large which significantly overshadows $\mathcal{D}$, making $d_{\text{MDD}}$ overlook the discrepancy measure $\mathcal{D}$.

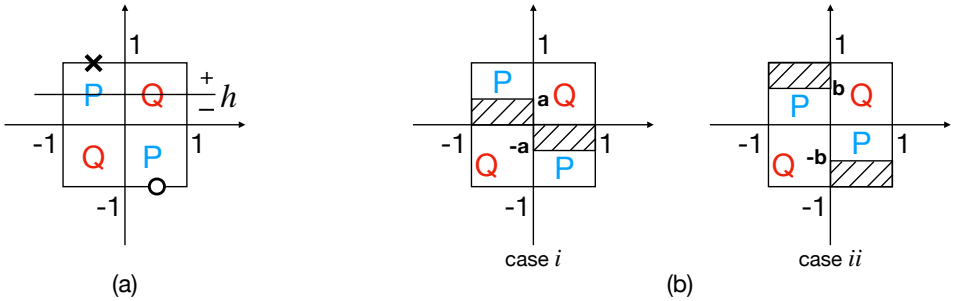


Figure 6: Examples for comparing $d_{\text{MDD}}$ and $d_{i\text{-MDD}}$. (a): for large $\lambda$. (b): for medium $\lambda$.

To see an example, consider a variant of Figure 1 where the data uniformly fills $[-1, 1] \times [-1, 1]$ as plotted in Figure 6 (a). $P$ and $Q$ are the source and target domains, respectively. In the top-left area $P$, suppose only one example (marked by x with vertical coordinate 1) is confidently labeled as positive, and the rest examples are highly inconfidently labeled, hence not to contribute to the risk $\mathcal{R}$. Similarly, there is only one confidently labeled example ($\circ$) in the bottom-right area of $P$, and it is negative with vertical coordinate $-1$. Since $h$ (as a hypothesis) shifts vertically, we will also use $h$ to denote its coordinate on the vertical axis. As was explained in the caption of Figure 1, $\mathcal{D}(h) = 1 - h$. Since the distance between $h$ and the positive x is $1 - h$, the probability of $x$ being positive, according to $h$, is sigmoid$(1 - h)$. Similarly, the probability of $\circ$ being negative, according to $h$, is $1 - \text{sigmoid}(-1 - h)$. Putting them together, we get the following $\mathcal{R}$ with cross-entropy loss and no regularization

$$\min_{h \in [0,1]} \lambda \underbrace{\left(\log(1 + e^{h-1}) + \log(1 + e^{-1-h})\right)}_{\mathcal{R}(h)} + \underbrace{1 - h}_{\mathcal{D}(h)}. \tag{24}$$

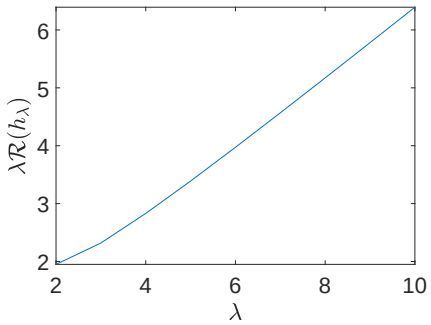


Figure 7: $\lambda\mathcal{R}(h_\lambda)$ as a function of $\lambda$

Whenever $\lambda > 2$, the optimal $h_\lambda$ is in $(0, 1)$ and can be solved by a quadratic equation. Figure 7 shows that $\lambda\mathcal{R}(h_\lambda)$ diverges linearly in $\lambda$. Flipping $h$ to $[-1, 0]$ produces the same issue.

In contrast, $d_{i\text{-MDD}}$ is immune to this problem because $\mathcal{R}$ is used only to determine $h^*$, while the $d_{i\text{-MDD}}$ value itself is solely contributed by $\mathcal{D}$. Although the $i$-MDD objective in (11) also has a coefficient $\alpha$ on $\mathcal{R}$, the optimization there is on the feature $\phi$, not on $h$ any more.

**Case 3**: $\lambda$ is medium. Here we will study two distributions as shown in Figure 6 (b), and analyze how $i$-MDD produces reasonable preferences of "better aligned distribution", and how MDD produces less justifiable preferences.

Same as the scenario of large $\lambda$, we do not change the feature distribution of source and target domains, hence keeping $\mathcal{D}(h) = 1 - |h|$. Instead, we vary the confidence of labels in the source domain in order to generate new risk $\mathcal{R}$. In case i (left of Figure 6 (b)), we activate (make the label confident) the positive examples in the top-left $P$ if, and only if, its vertical coordinate is in $[0, a]$ ($a \in [0, 1]$). Similarly, we activate the negative examples in the bottom-right $P$ if, and only if, its vertical coordinate is in $[-a, 0]$. The activated areas are shaded.

In case ii, (right of Figure 6 (b)), we activate the positive examples in the top-left $P$ if, and only if, its vertical coordinate is in $[b, 1]$ ($b \in [0, 1]$). Similarly, we activate the negative examples in the bottom-right $P$ if, and only if, its vertical coordinate is in $[-1, -b]$. The activated areas are shaded.

Adopting a tiny regularizer $\epsilon|h|$ with very small $\epsilon > 0$, it is clear that in both cases, and *regardless of the value of $a$ and $b$*, the optimal $h^*$ is 0. Therefore, the $d_{i\text{-MDD}}$ value is 1, which properly quantifies the discrepancy between $P$ and $Q$ regardless of the disclosure of source-domain labels.

However, the computation for $d_{\text{MDD}}$ is a little more involved. We first plot $\mathcal{R}(h)$ as a function of $h$ here:

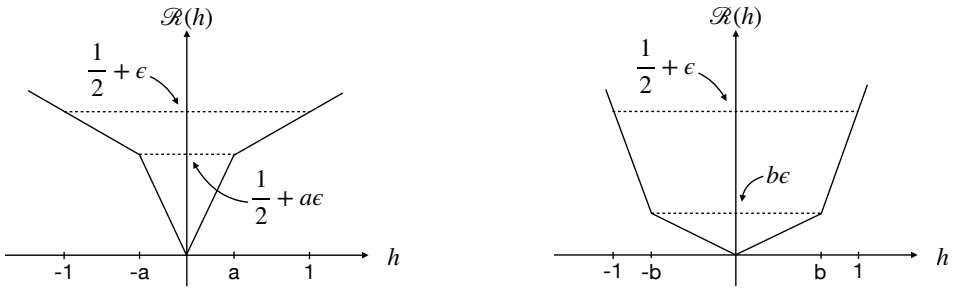


Figure 8: Plot of $\mathcal{R}(h)$ for case i and ii in Figure 6 (b)

Then the plot of $\mathcal{D}(h) + \lambda\mathcal{R}(h)$ is

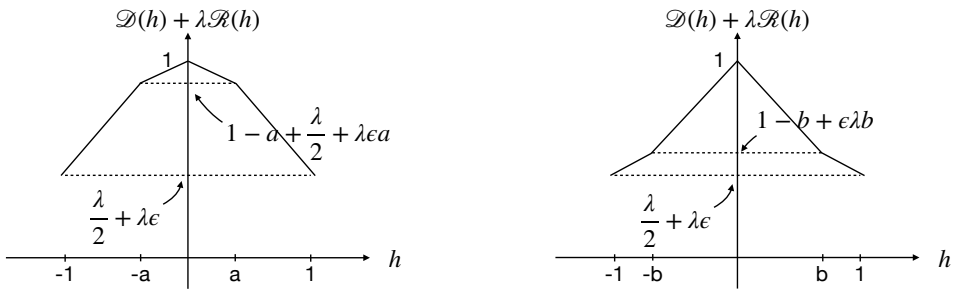


Figure 9: Plot of $\mathcal{D}(h) + \lambda\mathcal{R}(h)$ for case i and ii in Figure 6 (b)

So we have

$$d_{\text{MDD}} = \begin{cases} \min\{1, \frac{\lambda}{2} + \lambda\epsilon, 1 - a + \frac{\lambda}{2} + \lambda a\epsilon\} & \text{case i} \\ \min\{1, \frac{\lambda}{2} + \lambda\epsilon, 1 - b + \lambda b\epsilon\} & \text{case ii} \end{cases}. \tag{25}$$

If $\lambda > 2$, then $d_{\text{MDD}} = 1$ for case i, while that for case ii is strictly less than 1 unless $b = 0$ ($\epsilon$ is infinitesimally small). So case ii is always preferred.

If $\lambda \leq 2$, then $d_{\text{MDD}} = \frac{\lambda}{2} + \lambda\epsilon$ for case i. So there are only two situations left depending on $b$ for case ii.

- $b \in [0, 1 - \frac{\lambda}{2}]$: both cases have $d_{\text{MDD}} = \frac{\lambda}{2} + \lambda\epsilon$, i.e., equally preferred. This is the desirable outcome.
- $b \in [1 - \frac{\lambda}{2}, 1]$: then $d_{\text{MDD}} = 1 - b + \lambda b\epsilon$ for case ii, and it is therefore preferred to case i.

To summarize, $d_{\text{MDD}}$ always prefers case ii to case i, except when $\lambda < 2$ and $b \in [0, 1 - \frac{\lambda}{2}]$, in which case it is a tie. This is clearly not desirable because, by symmetry, there is no reason to prefer case ii. It is also particularly concerning that the value of $a$ in case i does not make any difference to the preference. As such, $d_{\text{MDD}}$ is not as good as a $d_{i\text{-MDD}}$ in this example.

## B Detailed Formula for Bi-level Optimization

Let $\phi$ be the feature extractor which produces latent states $z^s := \phi(x^s)$ and $z^t := \phi(x^t)$. Let $m$ be the number of latent features, i.e., the dimensionality of $z^s$ and $z^t$. Recall $C$ is the number of classes. Denote

$$M(h, \phi) := \max_{h'} \mathcal{D}(h', h, \phi). \tag{26}$$

For convenience, we will denote the optimal $h'$ as $h'(h, \phi)$.

Given $\phi$, the $h$ can be determined by minimizing the risk on $\tilde{P}$ as in (12):

$$h_\phi := \arg\min_h \mathcal{R}(h, \phi). \tag{27}$$

Our overall optimization objective is

$$\min_\phi M(h_\phi, \phi) + \alpha\mathcal{R}(h_\phi, \phi) \Longleftrightarrow \min_\phi \left\{ M(h_\phi, \phi) + \alpha \min_h \mathcal{R}(h, \phi) \right\}. \tag{28}$$

To optimize $\phi$, we just need to compute the gradient in $\phi$. Since both $M$ and $\mathcal{R}$ depend on $\phi$ only through $z^s$ and $z^t$, we can consider the following equivalent objective

$$J(z) := M(h_z, z) + \alpha \min_h \mathcal{R}(h, z), \quad \text{where} \quad h_z := \arg\min_h \mathcal{R}(h, z). \tag{29}$$

Once the derivative $\frac{\partial J}{\partial z}$ is computed, the original derivative in $\phi$ can be easily computed through backpropagation. We will use mini-batches with size $b$.

**Step 1.** The second term in (29), $\min_h \mathcal{R}(h,z)$, admits a straightforward calculation of the derivative in $z$ thanks to the Danskin's theorem: $\nabla_z^\top \mathcal{R}(h_z,z) = \frac{\partial}{\partial z}|_{h_z,z}\mathcal{R}(h,z)$. Here $\nabla_z^\top$ stands for the transpose of the gradient in $z$ — hence a row vector — of $\mathcal{R}(h_z,z)$ (regarded as a function of $z$ only).

**Step 2.** The first term in (29), $M(h_z,z)$, poses the most challenge due to the bi-level optimization, and we can address it by using the techniques in [45]. Firstly, Eq 3 therein allows us to write

$$\nabla_z^\top M(h_z,z) = \underbrace{\tfrac{\partial}{\partial z}\big|_{h_z,z} M(h,z)}_{:=(a)} \quad - \quad \underbrace{v^\top \times \tfrac{\partial^2}{\partial h \partial z^\top}\Big|_{h_z,z} \mathcal{R}(h,z)}_{:=(b)} \tag{30}$$

$$\text{where} \quad v^\top = \tfrac{\partial}{\partial h}\big|_{h_z,z} M(h,z) \quad \times \quad \left[\tfrac{\partial^2}{\partial h \partial h^\top}\Big|_{h_z,z} \mathcal{R}(h,z)\right]^{-1}. \tag{31}$$

We will next show how to compute them in analytic forms, i.e., with no autodiff.

**Step 2a.** Here $(a)$ is easy to compute: first find $h'(h_z,z)$ and then $(a) = \frac{\partial}{\partial z}\mathcal{D}(h',h,z)$ evaluated at $(h'(h_z,z),h_z,z)$.

**Step 2b.** $v$ can be computed by using Algorithm 2-3 in [45]. Note in our work, $h$ is a linear classifier with a weight matrix $W \in \mathbb{R}^{m\times C}$. Accordingly, the $v$ is indeed a matrix $V \in \mathbb{R}^{m\times C}$.

Akin to Step 2a, $\frac{\partial}{\partial W}\big|_{W_z,z} M(W,z) = \frac{\partial}{\partial h}\mathcal{D}(h',W,z)$ evaluated at $(h'(W_z,z),W_z,z)$. The matrix inversion in (31) is a major obstacle, and we instantiate Algorithm 2-3 in [45] as follows:

1. Initialize by $V = D = \frac{\partial}{\partial W}\big|_{W_z,z} M(W,z) \in \mathbb{R}^{m\times C}$.

2. **for** $j = 1,\ldots,$ #max-iter **do**

3. $\quad D = D - \alpha \cdot \frac{\partial^2}{\partial W \partial W^\top}\Big|_{W_z,z} \mathcal{R}(W,z)\cdot D$

4. $\quad V = V - D$

So the computational bottleneck is step 3. However, there is a closed form to the directional Hessian if we use the cross-entropy loss, i.e.,

$$\mathcal{R}(W,z) = \mathbb{E}_{z^s\sim\tilde{P}}[-W_{:,y^s}^\top z^s + G(W^\top z^s)]. \tag{32}$$

Indeed, let

$$p^s := \frac{1}{\exp(G(W^\top z^s))}\begin{pmatrix}\exp(W_{:1}^\top z^s)\\ \vdots \\ \exp(W_{:C}^\top z^s)\end{pmatrix}, \quad \text{where} \quad G(u) := \log\sum\nolimits_{c=1}^{C}\exp(u_c). \tag{33}$$

and Appendix D of [49] shows that with $\mathbf{1}_C = (1,\ldots,1)^\top \in \mathbb{R}^C$, $\tilde{P}(x^s) = \frac{1}{b}$, $P = (p^1,\ldots,p^b)$, $Z = (z^1,\ldots,z^b)$,

$$\frac{\partial^2}{\partial W \partial W^\top}\Big|_{W_z,z} \mathcal{R}(W,z)\cdot D = \frac{1}{b} Z\left[Q^\top - P^\top \circ (\mathbf{1}_C^\top \otimes (Q^\top \mathbf{1}_C))\right], \tag{34}$$

$$\text{where} \quad Q = P \circ (D^\top Z). \tag{35}$$

Here $\otimes$ is the Kronecker product, and $\circ$ is the Hadamard product (elementwise). Since only $S$ changes over the iterations on $j$ while $Z$ does not, we can pre-compute $P$ and $Z^\top Z$. Furthermore, we only need to compute the $Q^\top - P^\top \circ (\mathbf{1}^\top \otimes (Q^\top \mathbf{1}))$ as a surrogate for $D$, and then use the pre-computed $Z^\top Z$ in $Q$.

**Step 2c.** Given $v$, we will compute $(b)$ as follows. Since $W$ is a matrix, the derivative can be complicated. So we resort to the vectorization operator $\mathbf{w} := \text{vec}(W)$, and accordingly, we can consider $v$ as the vectorization of a matrix $V \in \mathbb{R}^{m\times C}$. Then the derivative in $\mathbf{w}$ can be written as

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{R}(\mathbf{w},z) = \mathbb{E}_{z^s\sim\tilde{P}}[(p^s - e_{y^s})\otimes z^s], \tag{36}$$

where $e_{y^s}$ is the $y^s$-th canonical vector in $\mathbb{R}^C$. We next compute $v^\top \frac{\partial^2}{\partial \mathbf{w}\partial z^\top}\mathcal{R}(\mathbf{w},z)$.

Obviously its derivative in $z^t$ is 0, and its derivative in $z^s$ is

$$\tilde{P}(x^s)v^\top \tfrac{\partial}{\partial z^s}[(p^s - e_{y^s}) \otimes z^s] = \tilde{P}(x^s)v^\top (\tfrac{\partial}{\partial z^s}[p^s \otimes z^s] - e_{y^s} \otimes I_m) \tag{37}$$

$$= \tilde{P}(x^s)v^\top \tfrac{\partial}{\partial z^s}[p^s \otimes z^s] - \tilde{P}(x^s)V_{:,y^s}^\top, \tag{38}$$

where $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix and $v = \mathrm{vec}(V)$. To compute the first term in (38), we drop the superscript $s$ for simplicity. Notice that for any class $c$ from 1 to $C$, we have

$$\tfrac{\partial p_c}{\partial z} = p_c W_{:c}^\top - p_c \sum_{i=1}^{C} p_i W_{:i}^\top = p_c(e_c - p)^\top W^\top. \tag{39}$$

Therefore

$$\tfrac{\partial}{\partial z}(p_c z) = p_c I_m - z\tfrac{\partial}{\partial z} p_c = p_c(I_m - z(e_c - p)^\top W^\top), \tag{40}$$

which implies that

$$v^\top \tfrac{\partial}{\partial z}[p \otimes z] = \sum_c p_c V_{:c}^\top (I_m - z(e_c - p)^\top W^\top) \tag{41}$$

$$= (Vp)^\top + (z^\top Vp)(Wp)^\top - [p^\top \circ (z^\top V)]W^\top. \tag{42}$$

This can be computed efficiently because it only involves matrix-vector multiplication. In practice, we would like to do it in a batch for all $s$ (recall we have dropped this superscript). Letting $A = VP$, $B = WP$, $F = Z^\top V$, it is not hard to derive that

$$\begin{pmatrix} v^\top \frac{\partial}{\partial z^1}[p^1 \otimes z^1] \\ v^\top \frac{\partial}{\partial z^2}[p^2 \otimes z^2] \\ \vdots \end{pmatrix} = A^\top + [(A^\top \circ Z^\top)\mathbf{1}_m \mathbf{1}_m^\top] \circ B^\top - (P^\top \circ F)W^\top. \tag{43}$$

To construct $(V_{:,y^1}, \ldots, V_{:,y^b})^\top$, we can utilize the infrastructure in the programming language. For example, in MATLAB, it can be easily computed by $V(:, [y^1, \ldots, y^b])'$.

### B.1 Analysis of computational cost

The calculation of the derivatives of the second term in (29) and the part (a) in (30) is straightforward. The computational cost is $\mathcal{O}(bm)$. Recall that the inverse Hessian vector production in (31) is the main computational bottleneck. The approximation algorithm in Step 2b can be solved with $\mathcal{O}(i_{\max} bmC)$, where $i_{\max}$ is the number of maximal iterations. The matrix vector multiplication in Step 2c costs $\mathcal{O}(bmC)$. Therefore, the total computational cost is upper bounded by $\mathcal{O}(i_{\max} bmC)$.

In practice, we used conjugate gradient (CG), where the $i_{\max}$ stands for the maximum number of iterations for CG. We set $m = 1024$, $b = 150$, and $C$ can be at most 65 in our datasets. Instead of limiting the maximum number of iterations, we set the tolerance of convergence to $10^{-5}$. The final time cost for completing CG over the entire mini-batch was less than a second, and the remaining operations in implicit differentiation (30) are much less expensive.

## C Bounding the gap in gradient from cache augmentation

The key advantage of $i$-CDD is the principled optimization. While the cache augmentation in Section 4.2 may appear ad hoc, we point out here that it only introduces a bias in the gradient optimization that can be bounded linearly by the queue length, i.e., staleness.

Suppose our mini-batch size is $b$ and the input samples drawn at iteration $\tau$ are $\{x_i^\tau\}_{i=1}^b$. Note we do not distinguish source or target domain and simply treat them as $x_i^\tau$. Suppose at the beginning of iteration $\tau$, the feature extractor is $\phi_\tau$. Then the latent features are $z_i^\tau = \phi_\tau(x_i^\tau)$. Suppose we store the latent feature of the past $s$ steps, i.e., $\{z_i^{\tau-1}\} \cup \ldots \cup \{z_i^{\tau-s}\}$. That is, $s$ is our staleness factor. To simplify notation, we denote $z^{\tau-1} := \{z_i^{\tau-1}\}$ and $z^{\tau_1:\tau_2} := z^{\tau_1} \cup \ldots \cup z^{\tau_2}$. Suppose the ultimate objective value of $i$-CDD is $J$, then our algorithm with cache augmentation computes the gradient in $\phi$ at iteration $\tau$ as

$$g := \frac{1}{b}\sum_{i=1}^{b} \frac{\partial z_i^\tau}{\partial \phi} \frac{\partial}{\partial z_i^\tau} J(z^{\tau-s:\tau}). \tag{44}$$

Here the average is only on the $z_i^\tau$ of the current iteration $\tau$, although $J$ is computed using stale $z$ features in $\tau-1,\dots,\tau-s$.

Our goal is to bound the distance between $g$ and the "correctly" computed gradient. It is important to note that $z_i^{\tau-1}$ is computed by the *past* features $\phi_{\tau-1}$, not the current $\phi_\tau$. Hypothetically, if we could compute them by using the latest $\phi_\tau$, then let us denote such fictitious $z$ as $\hat{z}_i^{\tau-1} := \phi_\tau(x_i^{\tau-1})$ and define a syntactic sugar $\hat{z}_i^\tau = z_i^\tau$. Then the "correct" gradient from a principled stochastic gradient can be computed by

$$g^* := \frac{1}{b(s+1)} \sum_{\pi=\tau-s}^{\tau} \sum_{i=1}^{b} \frac{\partial \hat{z}_i^\pi}{\partial \phi} \frac{\partial}{\partial \hat{z}_i^\pi} J(\hat{z}^{\tau-s:\tau}). \tag{45}$$

So we can bound the bias by

$$\|g-g^*\| \le \|g-\hat{g}\| + \|\hat{g}-g^*\|, \quad \text{where} \quad \hat{g} := \frac{1}{b}\sum_{i=1}^{b} \frac{\partial z_i^\tau}{\partial \phi} \frac{\partial}{\partial z_i^\tau} J(\hat{z}^{\tau-s:\tau}). \tag{46}$$

Firstly, $g^*$ and $\hat{g}$ both evaluate $J$ based on the augmented sample $\hat{z}^{\tau-s:\tau}$ that is computed hypothetically through the latest $\phi_t$. The former then averages the partial derivative over all the $b(s+1)$ samples while the latter only averages over the latest $b$ samples. This deviation does *not* involve any staleness, and can be bounded by standard concentration bounds such as Hoeffding's inequality.

Secondly, $g$ and $\hat{g}$ differ only in how $J$ is computed. The former uses the stale samples $z^{\tau-s:\tau}$, while the latter uses the fictitious samples $\hat{z}^{\tau-s:\tau}$. Since $J$ is a smooth function,

$$\frac{\partial}{\partial z_i^\tau} J(z^{\tau-s:\tau}) - \frac{\partial}{\partial z_i^\tau} J(\hat{z}^{\tau-s:\tau}) \tag{47}$$

can be bounded by the difference in the input arguments of $J$. Since the gradient in $\phi$ is bounded, so $\|\phi_\tau - \phi_{\tau-s}\| \le \mathcal{O}(s)$. Therefore, $\left\|z_i^{\tau-s} - \hat{z}_i^{\tau-s}\right\| \le \mathcal{O}(s)$, and the mean averaging inside $J$ implies

$$\left\| \frac{\partial}{\partial z_i^\tau} J(z^{\tau-s:\tau}) - \frac{\partial}{\partial z_i^\tau} J(\hat{z}^{\tau-s:\tau}) \right\| \le \mathcal{O}(s) \qquad \text{and hence} \qquad \|g-\hat{g}\| \le \mathcal{O}(s). \tag{48}$$

To summarize, the error in the gradient consists of the standard stochastic gradient noise, along with a term that is bounded linearly by the staleness $s$, which is in turn linear in the cache/queue size. So as long as we do not keep many past features, the optimization will work well. An empirical study has been shown in Section 6.2, and the values of $b$ and cache size have been provided there.

# D Experiment Details

## D.1 Implementation details

We used the official code of CDD, MDD, and MDD+IA to produce the results for Office-Home and Image-CLEF datasets. For other baselines, since the experimental configurations are the same, we quoted the highest results in the corresponding literature. Our PyTorch implementaion is available at `https://www.dropbox.com/sh/8e2enu3mwl7oxwk/AAAT8_xqkyLzLMqxqFH6tTjWa?dl=0`.

We first implemented a variant of CDD, named vCDD, where $\mu_c^s - \mu_{c'}^t$ was replaced by $\mu_c^s - \mu_{c'}^s$ in source domain *only*, and the class-aware sampling in [24] was replaced by cache augmentation. This allowed us to compare $i$-CDD with the exact counterpart that does not use bi-level optimization. We used ResNet-50 pre-trained on ImageNet as the feature extractor of vCDD model. The last FC layer of ResNet-50 was replaced by a 2-layer bottleneck neural network, where each layer has 1024 hidden units and batch normalization and sigmoid activation were applied to the hidden outputs. The bottleneck was immediately followed by a 1-layer classifier with multiple softmax units, each of which corresponds to an output class. $i$-CDD model used the same network architecture.

For $i$-MDD, to make a fair comparison, we followed MDD [23] to implement the network. ResNet-50 was adopted as the feature extractor with parameters pre-trained on ImageNet. The last FC layer of ResNet-50 was replaced by a 1-layer bottleneck network, where batch normalization, ReLU activation, and Dropout were applied to the outputs of the 1024 hidden units. Since we expected that a simple linear classifier could achieve high accuracy on the latent representations, instead of using 2-layer neural network, the main classifier $h$ and auxiliary classifier $h'$ were 1-layer neural network with width 1024.

### D.2 Hyper-parameter selection

Each method has hyper-parameters that are selected using the validation set which is comprised of labeled source examples and unlabeled target examples. The dimensionality of latent representations that are used for computing disparity discrepancy objectives, e.g. $d_{i\text{-MDD}}, d_{i\text{-CDD}}$, was selected from $\{128, 256, 512, 1024, 2048\}$. The size of the circular queue (cache) for each class was selected from $\{10, 30, 50, 100, 200\}$. For $i$-MDD method, the trade-off parameter $\alpha$ in (11) was selected from $\{0.01, 0.1, 1, 10, 100\}$; the trade-off parameter $\gamma$ in (13) was selected from $\{2, 3, 4, 5, 10\}$. For CDD and $i$-CDD methods, the trade-off parameter $\beta$ in (14) and (20) was selected from $\{0.001, 0.01, 0.1, 1\}$.

The hyper-parameters that were used for producing the results are summarized here:

Table 4: Hyper-parameters for all algorithms

| Dataset | Algorithm | latent dimension | cache size | $\alpha$ | $\beta$ | $\gamma$ |
|---|---|---|---|---|---|---|
| Office-31 | CDD | 1024 | 30 | - | 0.001 | - |
| | $i$-CDD | 1024 | 30 | - | 0.001 | - |
| | $i$-MDD | 1024 | - | 10 | - | 4 |
| Office-Home | CDD | 1024 | 50 | - | 0.01 | - |
| | $i$-CDD | 1024 | 50 | - | 0.01 | - |
| | $i$-MDD | 1024 | - | 10 | - | 4 |
| Image-CLEF | CDD | 2048 | 30 | - | 0.001 | - |
| | $i$-CDD | 2048 | 30 | - | 0.001 | - |
| | $i$-MDD | 2048 | - | 10 | - | 4 |

### D.3 Additional comparison with methods not based on feature adaptation

We also compared with three state-of-the-art methods for unsupervised domain adaptation that are not based on feature adaptation. These include [72], [73], and [74]. The performance on all the datasets is summarized in Table 5, in comparison with $i$-CDD:

Table 5: Accuracy on target domain

| Method | Office-31 | Office-Home | ImageCLEF |
|---|---|---|---|
| [72] | 88.6 | 71.8 | 88.5 |
| [73] | 89.6 | 71.0 | 90.3 |
| [74] | 88.8 | 69.2 | 90.2 |
| $i$-CDD | 90.9 | 70.8 | 89.4 |

In Table 5, we conducted the experiment for [72] on ImageCLEF, and the results for each domain are as follows:

| I -> P | P -> I | I -> C | C -> I | C -> P | P -> C |
|---|---|---|---|---|---|
| $77.4 \pm 0.5$ | $92.2 \pm 0.6$ | $96.1 \pm 0.2$ | $91.7 \pm 0.4$ | $77.6 \pm 0.6$ | $95.8 \pm 0.4$ |

The rest of the results in the table are quoted from the original paper, after checking manually on the data and their code.

Our $i$-CDD outperforms all these methods on Office-31. In addition, [72] is inferior to $i$-CDD on ImageCLEF, and [74] is inferior on Office-Home. [73] is almost the same as $i$-CDD on Office-Home. In addition, [73] requires solving a large generalized eigenvalue systems in their Eq 7. According to their Section "Computational Complexity", the cost is $O(d_1(d_1^2 + n^2))$ for $n$ images in the source and target domains combined, and $d_1$ can be as large as 1024. So it is highly intensive in computation for large $n$. Although stochastic PCA could be applied, its impact on the performance remains unclear.

To conclude, our $i$-CDD performs very competitively overall, and it could be overly demanding to require a method outperform state of the art on *all* datasets.

### D.4 Additional ablation studies

**Impact of Batch Size**

In our methods, random sampling was used to produce mini-batch data. Obviously, the mini-batch size determines the sampling distribution of the label space. For instance, when the mini-batch size is small, it may happen that within a given batch of samples, all source samples were drawn from 10 classes among 65 classes and all target samples were drawn from another 10 classes. The class-wise alignment objectives would suffer from this between-domain class distribution shift in the form of misalignment. Therefore, we investigated the impact of batch size.

Table 6: Impact of mini-batch size on target domain accuracy (Ar $\rightarrow$ Cl, Office-Home)

| batch size | vCDD | $i$-CDD |
|---|---|---|
| 16 | 28.4 | 29.5 |
| 32 | 39.3 | 38.8 |
| 64 | 55.9 | 57.3 |
| 128 | 56.9 | 59.4 |
| 256 | 56.7 | 59.2 |

As shown in Table 6, both vCDD and $i$-CDD enjoyed performance improvement with increased mini-batch size. Both methods worked better with a larger mini-batch size. This is because large mini-batch increases the empirical class diversity in each batch. This result suggests that class-conditioned domain adaptation approaches work well when the class diversity is high, e.g., when each mini-batch covers the whole label space.

**Standard deviations of Office-Home**

To complement Table 2, we next present the mean and standard deviation of target domain accuracy for vCDD, $i$-CDD, and $i$-MDD on the Office-Home dataset. Most existing literature does not report standard deviation on this dataset, so it was not reported in Table 2.

Table 7: Accuracy (%) on Office-Home for unsupervised domain adaptation

| Method | vCDD | $i$-CDD | $i$-MDD |
|---|---|---|---|
| Ar $\rightarrow$ Cl | 56.2 $\pm$ 0.6 | 60.8 $\pm$ 0.7 | 56.5 $\pm$ 0.5 |
| Ar $\rightarrow$ Pr | 74.2 $\pm$ 0.4 | 77.5 $\pm$ 0.7 | 74.7 $\pm$ 0.6 |
| Ar $\rightarrow$ Rw | 77.0 $\pm$ 0.6 | 78.8 $\pm$ 0.5 | 78.3 $\pm$ 0.3 |
| Cl $\rightarrow$ Ar | 62.4 $\pm$ 0.4 | 64.3 $\pm$ 0.5 | 61.9 $\pm$ 0.4 |
| Cl $\rightarrow$ Pr | 72.3 $\pm$ 0.5 | 74.3 $\pm$ 0.6 | 72.4 $\pm$ 0.4 |
| Cl $\rightarrow$ Rw | 71.4 $\pm$ 0.4 | 73.4 $\pm$ 0.5 | 72.3 $\pm$ 0.6 |
| Pr $\rightarrow$ Ar | 61.7 $\pm$ 0.7 | 65.3 $\pm$ 0.8 | 63.2 $\pm$ 0.7 |
| Pr $\rightarrow$ Cl | 61.4 $\pm$ 0.6 | 61.9 $\pm$ 0.6 | 55.6 $\pm$ 0.5 |
| Pr $\rightarrow$ Rw | 78.7 $\pm$ 0.6 | 78.7 $\pm$ 0.5 | 78.4 $\pm$ 0.3 |
| Rw $\rightarrow$ Ar | 71.3 $\pm$ 0.4 | 72.1 $\pm$ 0.5 | 71.4 $\pm$ 0.4 |
| Rw $\rightarrow$ Pr | 60.6 $\pm$ 0.5 | 61.8 $\pm$ 0.4 | 59.7 $\pm$ 0.2 |
| Rw $\rightarrow$ Cl | 81.7 $\pm$ 0.4 | 81.8 $\pm$ 0.6 | 81.7 $\pm$ 0.5 |
| Avg | 69.3 | 70.8 | 68.8 |

## 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]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
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]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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]
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]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (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]