Improving Unsupervised Domain Adaptation: A Pseudo-Candidate Set Approach

Aveen Dayal[1], Rishabh Lalla[1], Linga Reddy Cenkeramaddi[2], C Krishna Mohan[1], Abhinav Kumar[1], and Vineeth N Balasubramanian[1]

[1] Indian Institute of Technology Hyderabad, Hyderabad 502285, India
{ai21resch11003@, rishabh.lalla@cse., ckm@cse., abhinavkumar@ee., vineethnb@cse.}iith.ac.in
[2] University of Agder, Grimstad 4879, Norway
linga.cenkeramaddi@uia.no

Abstract. Unsupervised domain adaptation (UDA) is a critical challenge in machine learning, aiming to transfer knowledge from a labeled source domain to an unlabeled target domain. In this work, we aim to improve target set accuracy in any existing UDA method by introducing an approach that utilizes pseudo-candidate sets for labeling the target data. These pseudo-candidate sets serve as a proxy for the true labels in the absence of direct supervision. To enhance the accuracy of the target domain, we propose Unsupervised Domain Adaptation refinement using Pseudo-Candidate Sets (UDPCS), a method which effectively learns to disambiguate among classes in the pseudo-candidate set. Our approach is characterized by two distinct loss functions: one that acts on the pseudo-candidate set to refine its predictions and another that operates on the labels outside the pseudo-candidate set. We use a threshold-based strategy to further guide the learning process toward accurate label disambiguation. We validate our novel yet simple approach through extensive experiments on three well-known benchmark datasets: Office-Home, VisDA, and DomainNet. Our experimental results demonstrate the efficacy of our method in achieving consistent gains on target accuracies across these datasets.

Keywords: Unsupervised Domain Adaptation · Disambiguation · Pseudo-Candidate Set

1 Introduction

In recent years, Deep Neural Networks (DNNs) have showcased remarkable achievements in multiple domains such as robotics [19], medical imaging [37], agriculture [17], and beyond. However, in the realm of deep learning, domain shift presents a significant challenge as it occurs when the training set and the test set exhibit different data distributions, leading to a degradation in model performance [1, 29]. Unsupervised Domain Adaptation (UDA) has emerged as a prominent setting to address such challenges. UDA aims to adapt a model trained on a labeled training set (source domain) to perform well on an unlabeled test set (target domain) by minimizing the domain discrepancy and leveraging the intrinsic structure of the target domain [11, 41]. This adaptation is crucial for deploying deep learning models in dynamic environments where obtaining labeled data for every possible domain is impractical [6]. For instance, [16] explains how UDA techniques can be applied to bridge the gap between simulated and real-world domains, a common challenge in robotics.

To address the UDA problem, two prominent categories of methods have emerged: *adversarial* and *self-training* approaches. Adversarial methods, such as Domain-Adversarial Neural Networks (DANN) [11], Maximum Classifier Discrepancy (MCD) [36], and Margin Disparity Discrepancy (MDD) [50], focus on minimizing the domain discrepancy by aligning the feature distributions of the source and target domains through adversarial training. On the other hand, self-training techniques, exemplified by works like Cycle Self-Training (CST) [22], and Minimum Class Confusion (MCC) [15], leverage the model's predictions on the target domain to iteratively refine the model, aiming to improve its generalization to the target domain. Apart from these, hybrid approaches that integrate both adversarial and self-training paradigms for effective adaptation across domains have also been proposed [3,24,43]. These approaches have shown promising results in addressing the challenges of domain shift in UDA.

Despite these existing efforts in the field of UDA, we recognize an opportunity to further enhance performance in the target domain. Inspired by the use of candidate sets in partial-label learning [10, 40], we present a novel perspective to address the UDA problem by analyzing the patterns in the predictions of trained UDA models. As illustrated in Fig. 1, we observe that for multiple target samples, the trained UDA model's predictions are ambiguous, i.e., the model is not very confident about its predictions. Furthermore, we observe that this ambiguity is not uniformly distributed across all classes; instead, it is predominantly concentrated in a few closely related classes that exhibit the highest predicted probabilities, thereby leading to this ambiguity. To tackle this ambiguity, we introduce a novel technique, Unsupervised Domain Adaptation refinement using Pseudo-Candidate Sets **(UDPCS)**, which refines the predictions within the target domain. **UDPCS** starts by creating a pseudo-candidate set for each target sample, consisting of potential true labels based on the UDA model's predictions. Subsequently, inspired by [25], we propose a disambiguation process which is applied to these pseudo-candidate sets. Additionally, our method extends supervision beyond the pseudo-candidate set, penalizing predictions that stray into other classes. As illustrated in Fig. 1, the proposed **UDPCS** method effectively refines the UDA model's predictions by resolving ambiguity. It achieves this by identifying the potentially correct label and increasing its confidence. The efficacy of **UDPCS** is demonstrated through extensive experiments on various benchmark UDA datasets, showcasing its potential to enhance target accuracy in UDA models.

The following are the main contributions of this work:(i) We introduce the concept of a pseudo-candidate set, which is generated from the predictions of the UDA model. To the best of our knowledge, this work is the first to use the idea of pseudo-candidate set in the UDA setting; (ii) We propose a novel refinement technique for UDA, **UDPCS**, by leveraging the identified pseudo-candidate sets; and (iii) We study the proposed method by conducting extensive experiments on multiple UDA benchmark datasets, observing consistent performance gains across various datasets and UDA methods.

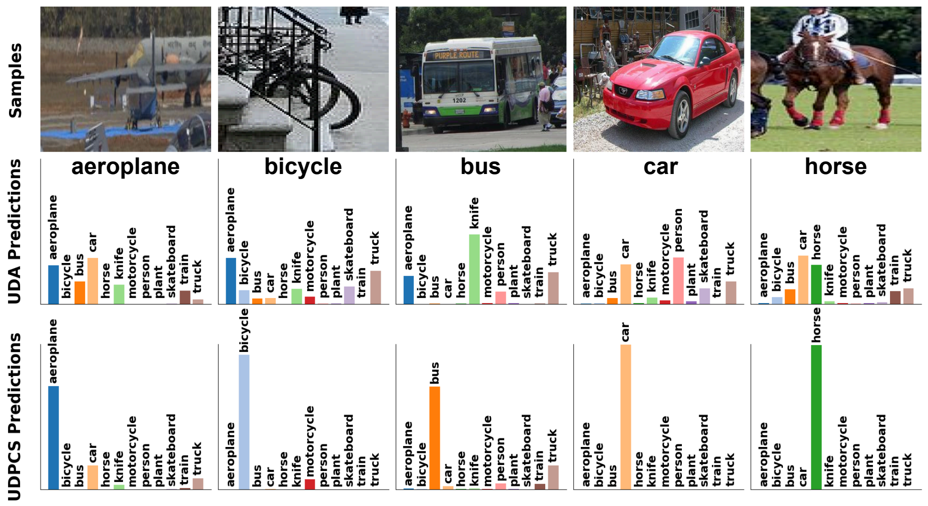

Fig. 1: Comparison of predictions between the trained UDA model (MDD [50]) and the proposed **UDPCS** method after refining the trained UDA model on the VisDA dataset for five target samples.

2 Related Work

This section discusses the earlier literature that was proposed specifically related to UDA and the disambiguation process inspired by partial-label learning.

UDA aims to perform well on the unlabeled target domain by leveraging the labeled source data. Inspired by the theoretical analysis [1], learning domain-invariant features has emerged as a pivotal approach. This concept has given rise to one of the most prevalent methodologies for addressing UDA challenges: adversarial methods. Inspired by GAN, the key principle behind adversarial methods is to minimize the domain shift through a minimax optimization strategy. [11] introduced DANN, a pioneering work that employs a deep learning-based adversarial approach for UDA using a generator and a domain discriminator. Bi-classifier approaches, as proposed by [36] and [20], propose an adversarial approach involving a single feature extractor and two classifiers. [50] introduced a novel disparity discrepancy method, Margin Disparity Discrepancy (MDD), to learn domain invariant features. Various other adversarial methods include [9,12,14,23,35,44,45]. Another significant avenue in UDA is self-training, where the model iteratively enhances its predictions to perform effectively on the target domain. Minimum Class Confusion (MCC), as proposed by [15], introduces a novel loss function that minimizes class confusion, leading to improved transfer gains. Additionally, Cycle Self-Training (CST), presented in [22], advocates for a cyclic self-training approach that refines target pseudo-labels in two steps and by leveraging Tsallis entropy as a regularizer. Various other methodologies in this category include [5, 26, 38]. Beyond these techniques, alternative methods such as optimal transport [18, 28, 32], moment matching [2, 21], and self-supervised techniques [4,48], have been proposed to address UDA challenges. Notably, recent years have witnessed the emergence of several hybrid approaches, which typically amalgamate different categories of techniques [3,24,27,33,43,51]. In this work, we introduce a novel perspective for tackling the UDA problem by analyzing the predictions of any trained UDA model on the target domain. Our proposed method is designed to iteratively refine the UDA model's prediction on the target domain.

Disambiguation is a key aspect of partial label learning (PLL). The PRODEN method [25] is a pivotal work in PLL that employs an iterative strategy for refining candidate sets, directly targeting label disambiguation. Following PRODEN, recent advancements have continued to refine the disambiguation process in PLL. LWC [46] introduces a weighted loss function that prioritizes probable labels within candidate sets, enhancing the disambiguation process. CR [47] introduces consistency regularization in PLL for disambiguating the labels. These developments highlight the importance of refining disambiguation processes for accurate learning outcomes. In this work, inspired from [25], we introduce the concept of pseudo-candidate set and propose a disambiguation process on these pseudo-candidate sets. Unlike [49], which introduces a hybrid setting between UDA and PLL, our work focuses on leveraging the concepts from PLL to refine the predictions on the target domain in UDA.

3 UDPCS: Methodology

Preliminaries. This section describes the basic assumptions and notations for classification problems in UDA. In a typical UDA framework, three key components are involved: a feature extractor ($\mathcal{G}$), a classifier (f), and a domain adaptive component. The source and target domains are denoted by $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively. The term $\mathcal{D}$ refers to any of these domains when the index is irrelevant. Throughout this work, the terms 'domain' and 'distribution' are used interchangeably. Each such domain $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is an input space and $\mathcal{Y}$ is the output (label) space, which is $\{0, 1\}$ in binary classification and $\{1, \cdots, k\}$ in multi-class classification. The term $\hat{D}$ is used to denote a set of samples drawn independently from $\mathcal{D}$, i.e. $\hat{D} = \big\{(x_i, y_i)\big\}_{i=1}^{n}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, $\forall i \in \big\{1, 2, \ldots, n\big\}$. We use (x, y) to refer to a sample (x_i, y_i) when the index is irrelevant. We define the model, $h(x) = f(\mathcal{G}(x))$, where $h : \mathcal{X} \to \mathbb{R}^k$, with each dimension of the output representing the class probability for each class. We also use Z to refer to the pseudo-candidate set where $Z \subset \mathcal{Y}$.

Methodology. We now present the methodology of our proposed approach, UDPCS, designed to enhance the accuracy of the target domain by leveraging pseudo-candidate sets. In the standard UDA setting, we assume that the training data consists of a labeled source domain, $\mathcal{D}_S$, and an unlabeled target domain, $\mathcal{D}_T$. The primary objective of a UDA method is to perform well on unlabeled target data. This is generally accomplished by employing a classification loss and an adaptation loss that minimizes the discrepancy between the source and target features. Over time, various adaptation loss functions have been proposed, depending on the specific UDA method employed [15,43,50]. Our proposed UDPCS method builds upon these standard techniques to further improve UDA performance with minimal overhead. We now describe our methodology and then illustrate it with a toy example.

In order to serve as a refinement strategy over any UDA method, we first train the UDPCS model $h(.)$ using any off-the-shelf UDA algorithm. Once the UDA training is complete, we employ the trained model $h(.)$ to generate a pseudo-candidate set for target data, as below.

$$Z = \{j | h_j(x) > \tau, \forall j \in \{1, \dots, k\}\}, \tau > 0 \tag{1}$$

For each input x, Z represents the pseudo-candidate set consisting of class indices j for which the corresponding class prediction probability, $h_j(x)$, exceeds a pre-defined threshold τ. This approach allows us to include only those classes in the pseudo-candidate set that have a relatively higher likelihood of being the true label, as indicated by the model $h(.)$. We further define a non-candidate set as the complement of the pseudo-candidate set, i.e., all class indices not in Z.

Subsequently, for the refinement of the model's predictions in the target domain, our method is trained by utilizing an objective that consists of two loss functions: *candidate loss* L_c and *non-candidate loss* L_{nc}. These loss functions are each described below.

Candidate Loss. This loss guides the model $h(.)$ to align its predictions with the pseudo-candidate set. We introduce a label confidence vector $\boldsymbol{w} \in \mathbb{R}^k$, representing the confidence for all k classes. Each element w_j in this vector signifies the confidence assigned to class j. We ensure that $\sum_{j \in Z} w_j = 1$ and $w_j = 0 \ \forall j \notin Z$, meaning that the confidence values w_j are assigned only to classes present in the pseudo-candidate set, and set to 0 otherwise. Consequently, $\boldsymbol{w}$ serves as a distribution over the classes in the pseudo-candidate set. The candidate loss applied to the pseudo-candidate set is then defined as:

$$L_c = \begin{cases} KL(\boldsymbol{w} || h(\mathcal{A}_1(x)) & \text{if } \max(h(\mathcal{A}_1(x))) < \gamma \\ 0 & \text{otherwise} \end{cases}, \gamma < 1 \tag{2}$$

The candidate loss is applied to samples for which the model's maximum predicted class probability falls below a certain threshold γ. Rather than using the raw input x, we utilize a strongly augmented version, $\mathcal{A}_1(x)$, as the input. Inspired by FixMatch [39], for strong augmentations, we leverage Cutout [8] and

RandAugment [7], which produce heavily distorted versions of the given input x. The rationale behind employing strong augmentation and thresholding is to differentiate between samples based on the model's predictions. For samples where the model's prediction is just above the threshold γ, the strong augmentation is likely to lower the prediction values, thereby including these samples in the candidate loss L_c. Conversely, for samples where the model's predictions are very high, even after strong augmentation, the predictions remain robust and above the threshold. Consequently, these samples are excluded from the candidate loss, ensuring that the loss is primarily applied to samples where the model's predictions are less certain. This loss is calculated as the Kullback-Leibler (KL) divergence between the label confidence vector $\boldsymbol{w}$ and the model's predictions $h(\mathcal{A}_1(x))$. Thus, in this way, the candidate loss aligns the pseudo-candidate set distribution and the model's prediction distribution.

Non-Candidate Loss. This loss is applied to the elements outside the pseudo-candidate set as:

$$L_{nc} = -\sum_{j \notin Z} \log(1 - h_j(\mathcal{A}_2(x))) \tag{3}$$

The motivation for the non-candidate loss is that it penalizes the model for assigning high probabilities to classes that are not in the pseudo-candidate set Z. It is computed as the negative log-likelihood of the complement probabilities for classes outside Z, using the model's predictions $h(\mathcal{A}_2(x))$ after applying a weak augmentation $\mathcal{A}_2$ to the input x. We use a standard flip-and-shift augmentation as our weak $\mathcal{A}_2$. This loss discourages the model from considering classes outside the pseudo-candidate set as potential true labels.

Overall Objective. The overall objective function of the proposed UDPCS model is the sum of the candidate and non-candidate loss terms:,

$$Loss = L_c + \lambda L_{nc} \tag{4}$$

where λ is a hyperparameter that balances the contributions of the two loss functions.

Updating $\boldsymbol{w}$. As in Eqn. (2), the vector $\boldsymbol{w}$ plays a crucial role in guiding the model's predictions. Therefore, updating $\boldsymbol{w}$ after each iteration is essential for disambiguation of the pseudo-candidate set. We set the label confidence $\boldsymbol{w}$ initially to $\frac{1}{|Z|}$ if the class j is in the pseudo-candidate set Z, where $|Z|$ denotes the cardinality of Z. This ensures that each class in the pseudo-candidate set receives an equal initial confidence value. The rationale for assigning an equal initial confidence value to each class in the pseudo-candidate set is to enable the proposed UDPCS method to disambiguate the correct class from a set of classes without bias. This scheme is given below:

$$w_j = \begin{cases} \frac{1}{|Z|} & \text{if } j \in Z \\ 0 & \text{otherwise} \end{cases}, \; \boldsymbol{w} \in \mathbb{R}^k \text{ and } \sum_{j \in Z} w_k = 1 \tag{5}$$

Algorithm 1 Unsupervised Domain Adaptation refinement using pseudo-candidate sets (UDPCS)

```
1: Input: Source domain data $D_S = \{x_i^S, y_i^S\}_{i=1}^{n_S}$, target domain data $D_T = \{x_j^T\}_{j=1}^{n_T}$, learning rate $\eta$,number of iterations $T$, refinement iterations $T_{\text{refine}}$, strong augmentation $\mathcal{A}_1$, weak augmentation $\mathcal{A}_2$.
2: Output: Refined model $h$
3: Initialize model $h$ with random weights
4: for $t = 1$ to $T$ do
5:     $(x^S, y^S) \sim D_S$ and $(x^T) \sim D_T$
6:     Update the parameters of $h$ utilizing $(x^S, y^S)$ and $(x^T)$ in accordance with any standard unsupervised domain adaptation technique.
7: end for
8: Generate the pseudo-candidate set '$Z$' for each sample in $D_T$ using Eqn. (1)
9: Initialize the label confidence vector '$\boldsymbol{w}$' using Eqn. (5)
10: for $t = 1$ to $T_{\text{refine}}$ do
11:     $(x^T) \sim D_T$, with augmentations $\mathcal{A}_1(x^T)$ and $\mathcal{A}_2(x^T)$
12:     Compute the candidate loss $L_c$ using Eqn. (2)
13:     Compute the non-candidate loss $L_{nc}$ using Eqn. (3)
14:     Compute the final loss, $Loss = L_c + \lambda L_{nc}$
15:     Update the label confidence vector using Eqn. (6)
16:     Update model $h$ using gradient descent: $h \leftarrow h - \eta \nabla Loss$
17: end for
18: return $h$
```

After each iteration, the vector $\boldsymbol{w}$ is dynamically updated based on the model's predictions $h(\mathcal{A}_1(x))$, as below:

$$w_j = \begin{cases} \frac{h_j(\mathcal{A}_1(x))}{\sum_{m \in Z} h_m(\mathcal{A}_1(x))} & \text{if } j \in Z \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

As in Eqn. (6), the confidence w_j for each class j in the pseudo-candidate set Z is updated to be proportional to the model's prediction probability for that class, normalized by the sum of prediction probabilities for all classes in Z. This update ensures that w_j reflects the model's current confidence in each class within the pseudo-candidate set, facilitating the disambiguation process.

Therefore, by iteratively updating the model $h(.)$ using the aforementioned components, the proposed UDPCS improves the model's ability to disambiguate the correct class from the pseudo-candidate set, leading to more accurate and confident predictions on the target domain. Our overall method is summarized in Algorithm 1. We begin by training the proposed model $h(.)$ using a standard UDA technique. We then generate the pseudo-candidate set and initialize the label confidence vector. Subsequently, the method undergoes iterative training, incorporating both candidate and non-candidate loss in each iteration. We now illustrate our approach using a toy example.

Toy Example. To illustrate the working of our method, the toy example in Fig. 2 shows how the proposed UDPCS technique refines a model's predictions. In particular, we demonstrate a single iteration after UDA training. As shown in the figure, we consider a 5-class classification problem with labeled source data, unlabeled target data consisting of 5 samples from $S1$ to $S5$ and a model $h_{UDA}(.)$ trained using any off-the-shelf UDA technique. The unseen true labels

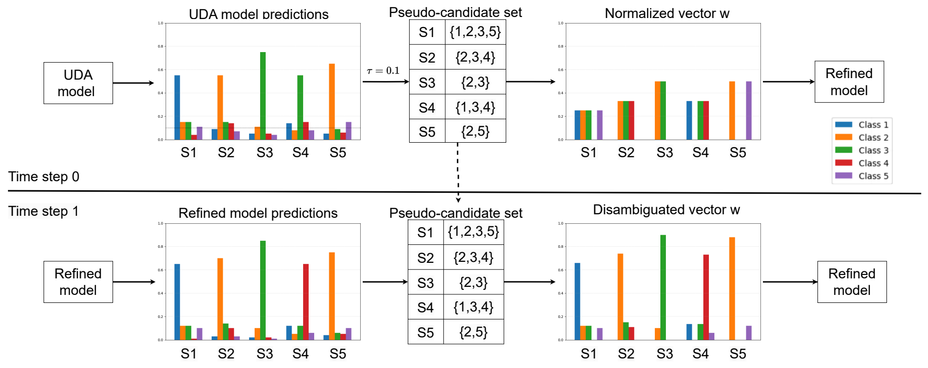

Fig. 2: Illustration of the toy example for demonstrating our UDPCS method. The figure visualizes the process of refining the predictions of a base UDA model by leveraging pseudo-candidate sets.

for these 5 target samples are assigned such that sample j is labeled as class j, $\forall j \in \{1, 2, \ldots, 5\}$.

As mentioned earlier in this section, after UDA training, our UDPCS method generates the pseudo-candidate set Z using model predictions $h_{UDA}(.)$ and with $\tau = 0.1$, according to Eqn. (1), for each target sample. The cardinality of these pseudo-candidate sets for this example ranges from 2 to 4 in Fig. 2. These sets are then utilized to generate the label confidence vector $\boldsymbol{w}$, which, as previously discussed, is initialized with uniform confidence values according to Eqn. (5) (to avoid error bias from the base model). This $\boldsymbol{w}$, along with the pseudo-candidate sets, is then used to refine the model's prediction by updating its parameters using Eqn. (4). In the next iteration, 'Step = 1' in the figure, this refined model generates predictions on the target set, which is then used to get the disambiguated vector $\boldsymbol{w}$ according to Eqn. (6).

As in the figure, the refined model's predictions for each sample definitively impacts model accuracy. After initial UDA model training, the model correctly predicted *S1*, *S2*, and *S3*, but incorrectly predicted *S4* and *S5*, resulting in 60% accuracy. Whereas after the refinement, *S4* is correctly disambiguated, while the confidence in predictions for *S1*, *S2*, and *S3* has increased. For sample *S5*, the disambiguation could not identify the correct label in the first iteration. These changes in predictions increased the model's accuracy from 60% to 80%. It is also noticeable that the prediction probabilities of classes outside the candidate set have reduced, which can be attributed to the non-candidate set loss. Thus, the proposed method improves UDA performance by refining the model's predictions. We show later in our experiments that with this little overhead, we outperform the base UDA model across different baselines, even when the baseline is allowed the same additional training time.

4 Experiments

Datasets: We conduct a comprehensive evaluation of our proposed UDPCS method on several benchmark UDA datasets, including VisDA [31], DomainNet [30], and OfficeHome [42]. Each dataset poses unique challenges for domain adaptation. The VisDA dataset, for instance, exhibits a significant domain gap between the source and target domains, encompassing 12 classes. The DomainNet dataset, on the other hand, offers a broader label space with 365 classes, presenting a different set of challenges. Lastly, the OfficeHome dataset, with its relatively limited number of samples across all domains, necessitates efficient adaptation strategies. Further details about these datasets are provided in the Appendix.

Implementation Details and Baselines: Consistent with recent benchmarks [24, 51], we employ the ResNet50 model as the feature extractor for the OfficeHome dataset and the ResNet101 model for the VisDA and DomainNet datasets. Our method is trained using stochastic gradient descent with momentum [34]. Additional implementation details, such as hyperparameters including threshold values, learning rates, etc., are detailed in the Appendix. Our UDPCS method enhances off-the-shelf UDA techniques; hence, we evaluate its effectiveness against a range of UDA methods such as MDD [50], MCC [15], LeCo [43], SDAT [35], and NWD [3], which encompass diverse approaches like adversarial and self-training. We benchmark our model by comparing the performance of these UDA methods with and without the application of our UDPCS technique. Additionally, we contrast our method with other recent benchmarks in Appendix A1. In our result tables, (Tab. 1 to Tab. 3), the performance of our UDPCS method is denoted as 'UDA method+Ours,' where 'UDA method' indicates the base method upon which UDPCS is applied.

Results on VisDA: The VisDA dataset presents a single adaptation scenario, from synthetic (source) to real-world (target) domains. Class-wise target accuracy results on the VisDA dataset are detailed in Tab. 1. As seen in Tab. 1, the UDPCS method enhances mean accuracy across five different UDA methods. Notably, the maximum improvement, a 2.4% gain, is observed when UDPCS is applied to both the MCC and NWD models. The VisDA dataset is known for its large domain gap between the source and target domains. Despite this challenge, UDPCS achieves an average gain of 2% in target accuracy, as demonstrated in Tab. 1, highlighting its efficacy in refining model predictions.

Results on DomainNet: In line with the settings from [13, 43], we carry out experiments on 12 tasks involving clipart (c), painting (p), real (r), and sketch (s) domains from the original DomainNet dataset. The detailed results for all tasks are showcased in Tab. 2. As indicated in Tab. 2, the UDPCS method surpasses the performance of each underlying UDA method, with a maximum gain of 3.5% in average target accuracy. As seen from Tab. 2, our method, 'MDD+Ours,' achieves the best performance compared to all the other methods. Despite DomainNet's challenge of a vast label space with 365 classes, the UDPCS method achieves an average gain of 1.6% in target accuracy, showcasing the robustness and effectiveness of our approach.

Table 1: Accuracy(%) on VisDA dataset. Values in **red** are the gains achieved by the proposed UDPCS method. Values underlined are the best-performing models in a column. Mean denotes Mean Accuracy. Appendix A1 presents additional results.

Model	plane	bcybl	bus	car	horse	knife	mcycle	persn	plant	sktb	train	truck	**Mean**	**Gains**
MDD(ICML'19)	94.6	75.0	68.5	70.2	91.4	94.3	91.5	79.7	91.6	88.4	84.3	36.8	80.5	
+**Ours**	96.1	76.8	69.3	72.4	93.3	96.5	92.5	80.5	92.6	89.7	86.8	38.6	82.1	**1.6**
MCC(ECCV'20)	94.5	85.9	74.5	71.6	93.9	94.9	85.9	82.3	91.4	91.7	82.4	57.4	83.8	
+**Ours**	96.2	88.4	77.6	73.7	95.1	97.7	89.5	83.7	94.5	93.4	85.2	59.1	86.2	**2.4**
LeCo(ACCV'22)	96.1	87.5	85.8	89.6	95.6	96.2	90.8	82.7	94.4	91.1	84.4	41.9	86.3	
+**Ours**	97.5	88.6	85.5	89.9	97.3	97.1	93.3	84.8	96.4	93.7	87.2	45.2	88.0	**1.7**
SDAT(ICML'22)	94.6	82.9	77.0	67.4	93.6	97.1	92.4	82.6	92.8	87.0	85.6	52.2	83.8	
+**Ours**	95.6	84.6	79.3	76.0	94.9	98.1	92.7	83.9	94.4	90.6	88.1	48.7	85.6	**1.8**
NWD (CVPR'22)	93.9	83.9	76.6	73.2	93.1	91.0	85.7	80.9	91.8	90.0	82.0	51.2	82.8	
+**Ours**	96.0	87.4	78.6	77.2	94.5	93.8	89.1	83.2	93.8	94.3	83.6	51.5	85.2	**2.4**
													Average Gains	**2**

Table 2: Accuracy(%) on DomainNet dataset. Values in **red** are the gains achieved by the proposed UDPCS method. Values underlined are the best-performing models in a column. (Avg. = Average Accuracy). Appendix A1 presents additional results.

Model	c-p	c-r	c-s	p-c	p-r	p-s	r-c	r-p	r-s	s-c	s-p	s-r	**Avg.**	**Gains**
CDAN (NeurIPS'18)	40.4	56.8	46.1	45.1	58.4	40.5	55.6	53.6	43.0	57.2	46.4	55.7	49.9	
+**Ours**	43.7	59.1	47.7	45.9	59.2	42	58.2	53.7	45.7	59	48.2	57.8	51.7	**1.8**
MDD (ICML'19)	42.8	59.2	48.3	49.2	59.6	43.6	58.8	54.1	46.6	59.1	46.7	57.8	52.2	
+**Ours**	46.7	63.5	51.4	51.1	62.3	46.3	63.1	57.7	50.7	63.4	50.3	61.9	55.7	**3.5**
MCC (ECCV'20)	37.1	55.6	41.4	45.5	59.9	39.7	54.3	53.1	36.8	58.2	46.5	56.1	48.7	
+**Ours**	39.1	56.6	43.0	47.2	60.6	41.1	56.4	55.0	38.8	59.1	47.0	57.7	50.1	**1.4**
SDAT (ICML'22)	41.9	57.2	47.2	47.8	60.2	42.3	56.5	53.9	42.7	60.4	48.0	57.5	51.3	
+**Ours**	43.7	58.1	47.9	48.1	59.4	43.0	56.0	53.1	43.8	58.7	47.6	57.7	51.4	**0.1**
													Average Gains	**1.7**

Results on OfficeHome: The OfficeHome dataset encompasses 12 tasks across the Art (A), Clipart (C), Product (P), and Real World (R) domains. Target accuracy results for all tasks are presented in Tab. 3. As evident in Tab. 3, the UDPCS method surpasses the performance of the underlying UDA methods, with a maximum gain of 1.8% in average target accuracy. As seen from Tab. 3, our methods, 'MCC+Ours' and 'NWD+Ours,' achieves the best performance compared to all the other methods. Although OfficeHome presents a challenge with its relatively small sample size per domain, the UDPCS technique still manages to achieve an average gain of 0.9%, demonstrating its effectiveness.

5 More Empirical Analysis and Ablation Studies

Ablation Studies: To assess the impact of various components of the proposed UDPCS method, we conduct ablation studies and present the results in Tab. 4. The table outlines experiments on three key components: candidate loss (L_c),

Table 3: Accuracy(%) on OfficeHome dataset. Values in **red** are the gains achieved by the proposed UDPCS method. Values underlined are the best-performing models in a column. (Avg. = Average Accuracy). Appendix A1 presents additional results.

Model	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**	**Gains**
MDD (ICML'19)	75.5	56.2	79	63.1	72.5	72.6	63.1	55.1	79.7	73.3	60.1	83.6	69.5	-
+Ours	77.6	57.8	80.5	65.4	75.1	75.4	65.1	56.5	81.6	74.1	61.3	84.7	71.3	**1.8**
MCC(ECCV'20)	79.8	58.1	83.2	68.6	76.6	78.6	67.8	55.5	82.2	74.4	60.8	85.8	72.6	
+ Ours	81.1	59.3	84.1	69.2	78	79.6	69.1	56.4	82.8	74.7	62.1	86.4	73.6	**1**
Leco (ACCV'22)	78.7	58.3	82.4	68.6	79.1	79.3	67.5	56.2	82.6	75	60.9	86	72.9	
+Ours	79.1	59.2	82.7	68.8	79.4	79.3	67.7	56.9	82.6	75.2	61.1	86.4	73.2	**0.3**
SDAT(ICML'22)	77.2	58.9	81.2	66.6	75.8	76.8	63.9	57	81.8	75.1	64.8	85.8	72.1	
+Ours	78.1	59.7	82.1	67	76.3	77	64.4	57.6	82.6	75.2	65.2	86	72.6	**0.5**
NWD(CVPR'22)	80.1	58.1	83.4	67.9	77.2	78.6	67.9	55.7	82	74.1	60.7	85.7	72.6	-
+Ours	80.9	59.1	83.9	68.6	79	80	68.5	56.7	83.2	74.9	62.1	86.7	73.6	**1**
													Average Gains	**0.9**

non-candidate loss (L_{nc}), and a threshold (γ), with a tick mark indicating the inclusion of a component in a particular experiment. The absence of all ticks denotes the use of the base UDA method without our refinement. We evaluate two base UDA methods, MDD and MCC, on the OfficeHome dataset, as detailed in Tab. 4. For the MDD+Ours model, introducing candidate loss (L_c) yields a 1.2% increase in average accuracy, while adding non-candidate loss (L_{nc}) brings a similar enhancement of 1.1%. The inclusion of thresholding (γ), particularly with candidate loss, leads to a 1.6% improvement, emphasizing the value of focusing on ambiguous samples. The combination of both losses, without the threshold, results in a 1.4% uptick, and the full UDPCS model, integrating all components, achieves the highest gain of 1.8% over the base MDD model. In the MCC+Ours model, the addition of candidate loss and non-candidate loss individually contributes to improvements of 0.4% and 0.7%, respectively. The integration of thresholding, either alone or with other components, consistently enhances performance, with the full UDPCS model outperforming the base MCC model by 1%. These findings highlight the synergistic effect of combining candidate loss, non-candidate loss, and thresholding, underscoring the comprehensive effectiveness of the UDPCS method in boosting average accuracy.

Complexity Analysis: The complexity analysis of the UDPCS method, as presented in Table A1 in the Appendix, illustrates its efficiency in terms of training time while achieving improvements in average accuracy. The UDPCS method achieves higher average accuracy without incurring any significant training time. Please see the Appendix for the complete discussion.

Hyperparameter λ : The exploration of the hyperparameter λ, which balances the candidate loss (L_c) and the non-candidate loss (L_{nc}) in the total loss equation ($L_c + \lambda L_{nc}$), reveals its impact on the performance of the UDPCS method, as shown in Table 5. We run each of the experiment three times are report the average values. For the MDD+Ours model, setting $\lambda = 0.5$ results in an aver-

Table 4: Ablation results (%) on OfficeHome dataset assessing the impact of UDPCS components.

Model	L_c	L_{nc}	γ	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**
MDD+Ours				75.5	56.2	79	63.1	72.5	72.6	63.1	55.1	79.7	73.3	60.1	83.6	69.5
	✓			76.9	58	80.2	64.8	74.7	74.8	63.6	55.9	80.9	73.5	61.1	84	70.7
		✓		76.8	57.3	79.7	65.4	74.5	75.1	64.5	55.7	80.2	73.4	60.5	83.9	70.6
	✓		✓	77.4	58.1	80.6	64.4	75.2	74.6	65	56.5	81.4	73.8	61.3	84.4	71.1
	✓	✓		77.1	57.6	80.5	64.5	74.8	75.1	64.5	56.4	81.2	74	61	84.3	70.9
	✓	✓	✓	77.6	57.8	80.5	65.4	75.1	75.4	65.1	56.5	81.6	74.1	61.3	84.7	**71.3**
MCC+Ours				79.8	58.1	83.2	68.6	76.6	78.6	67.8	55.5	82.2	74.4	60.8	85.8	72.6
	✓			80.4	59	83.8	69.1	77.3	78.7	68	56.5	82	74.2	61.4	85.9	73
		✓		80.3	58.8	84	69.2	77.2	79.7	68.5	55.8	82.9	75.1	61.5	86.2	73.3
	✓		✓	80.7	58.7	83.5	69	77.1	78.4	68.6	56.7	82.3	74.6	62	86	73.1
	✓	✓		80.4	58.8	84	68.8	77.5	78.9	68	56.2	82.9	74.1	61.5	86	73.1
	✓	✓	✓	81.1	59.3	84.1	69.2	78	79.6	69.1	56.4	82.8	74.7	62.1	86.4	**73.6**

age accuracy of 71.1%, which slightly improves to 71.2% when $\lambda = 1$. Further increasing λ to 2 maintains the average accuracy of 71.2%, suggesting a robust performance across different values of λ. Similarly, for the MCC+Ours model, the average accuracy is 73.6% for all the λ values with slight variations in the standard deviation values. These findings indicate that the UDPCS method is relatively stable across a range of λ values, with minor fluctuations in average accuracy. This stability underscores the method's robustness and the effectiveness of the balanced approach to handling candidate and non-candidate losses in refining predictions for domain adaptation tasks. Please see the Appendix for the complete results of this analysis.

Table 5: Accuracy(%) of UDPCS for different λ values on OfficeHome dataset.

λ	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**
							MDD+Ours						
$\lambda = 0.5$	77.2	58.2	80.6	64.8	75.3	74.9	65.1	56.5	81.0	73.9	61.4	84.5	71.1 ± 0.1
$\lambda = 1$	77.5	58.0	80.6	65.3	75.2	75.1	65.0	56.5	81.6	74.1	61.4	84.5	**71.2** ± 0.02
$\lambda = 2$	77.3	57.9	80.8	65.1	75.0	75.3	65.1	56.6	81.5	74.0	61.3	84.4	**71.2** ± 0.01
							MCC+Ours						
$\lambda = 0.5$	80.9	59.1	83.8	69.6	77.8	79.6	68.9	56.6	83.2	74.8	62.1	86.6	**73.6** ± 0.05
$\lambda = 1$	81.0	59.1	84.3	69.3	77.9	79.6	68.9	56.6	83.2	74.7	62.0	86.5	**73.6** ± 0.06
$\lambda = 2$	80.9	59.2	84.2	69.2	78.0	79.6	68.9	56.7	82.9	74.8	62.4	86.6	**73.6** ± 0.1

Weak Augmentation: The exploration of weak augmentation $\mathcal{A}_2$ in the candidate set loss for the UDPCS method, as compared to the main results obtained using strong augmentation $\mathcal{A}_1$, reveals interesting insights into the method's performance, as shown in Table 6. For the MDD+Ours model, employing weak augmentation results in an average accuracy of 69.6%, which is lower than the 71.3% achieved with strong augmentation. Similarly, the MCC+Ours model exhibits a decrease in average accuracy from 73.6% with strong augmentation to 72.8% with weak augmentation. These findings suggest that strong augmentation plays a crucial role in the effectiveness of the UDPCS method, particularly in refining predictions for domain adaptation tasks. Further augmentation related experiments are reported in the Appendix.

Table 6: Accuracy(%) of UDPCS for different augmentation used on OfficeHome dataset. (Aug. = Augmentation)

Aug.	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**
							MDD+Ours						
Weak	76.3	56.4	79	63.3	73.5	73.6	63.1	54.7	79.3	73	59.7	83.1	69.6
Strong	77.6	57.8	80.5	65.4	75.1	75.4	65.1	56.5	81.6	74.1	61.3	84.7	**71.3**
							MCC+Ours						
Weak	79.9	58.2	83.2	69.1	76.8	78.5	67.6	56.7	81.9	74.1	61.2	86	72.8
Strong	81.1	59.3	84.1	69.2	78	79.6	69.1	56.4	82.8	74.7	62.1	86.4	**73.6**

Threshold γ: The examination of different threshold values (γ) used in the candidate loss, Eqn. (2), of the UDPCS method provides insights into the method's sensitivity to this hyperparameter, as shown in Table 7. For the MDD+Ours model, the average accuracy remains consistent at 71.1% for both $\gamma = 0.8$ and $\gamma = 0.9$, with a slight increase to 71.3% when $\gamma = 0.85$. Similarly, for the MCC+Ours model, the average accuracy is stable at 73.5% for $\gamma = 0.8$ and $\gamma = 0.9$, with a marginal increase to 73.6% at $\gamma = 0.85$. The slight variations in average accuracy suggest that while the choice of threshold can influence the method's performance, the UDPCS method is not overly sensitive to this hyperparameter.

Table 7: Accuracy(%) of UDPCS for different threshold (γ) values on OfficeHome dataset.

	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**
							MDD+ours						
$\gamma = 0.8$	77.1	58.5	80.8	64.9	75.1	74.9	64.7	56.5	81.6	73.8	61.3	84.5	71.1
$\gamma = 0.85$	77.6	57.8	80.5	65.4	75.1	75.4	65.1	56.5	81.6	74.1	61.3	84.7	**71.3**
$\gamma = 0.9$	77.3	58.1	81.1	64.9	75	75	64.5	56.6	81.4	74	61.1	84.6	71.1
							MCC+Ours						
$\gamma = 0.8$	81.1	59.3	83.6	69.2	78	79.6	69.1	56.4	82.8	74.7	61.9	86.4	73.5
$\gamma = 0.85$	81.1	59.3	84.1	69.2	78	79.6	69.1	56.4	82.8	74.7	62.1	86.4	**73.6**
$\gamma = 0.9$	80.7	59.3	83.8	69.4	77.5	79.3	68.9	56.5	82.9	74.7	62.4	86.6	73.5

Cross entropy: The investigation of using cross entropy as an alternative to KL divergence for the candidate loss, Eqn.(2) in the UDPCS method reveals the method's adaptability and effectiveness across different loss functions, as demonstrated in Table 8. For the MDD+Ours model, employing cross entropy for the candidate loss yields an average accuracy of 71.3%, which is identical to the average accuracy achieved using KL divergence. Similarly, the MCC+Ours model maintains an average accuracy of 73.6% with both cross entropy and KL divergence. The consistency in average accuracy across different loss functions suggests that the method's ability to refine predictions for domain adaptation tasks is robust to the choice of loss function.

Table 8: Accuracy(%) of UDPCS for different loss functions on OfficeHome dataset. (CE= Cross Entropy, KL=KL Divergence)

Loss	A-P	A-C	A-R	C-A	C-P	C-R	P-A	P-C	P-R	R-A	R-C	R-P	**Avg.**
							MDD+Ours						
CE	77.4	58.1	80.7	64.9	75.2	75.3	65.3	56.3	81.5	73.9	61.7	84.8	**71.3**
KL	77.6	57.8	80.5	65.4	75.1	75.4	65.1	56.5	81.6	74.1	61.3	84.7	**71.3**
							MCC+Ours						
CE	81.1	59.3	84.1	69.2	78	79.6	69.1	56.4	82.8	74.7	62.1	86.4	**73.6**
KL	80.8	59.2	84.5	69.4	77.9	79.7	68.4	56.5	83.2	75.1	62.1	86.4	**73.6**

6 Conclusion

In conclusion, UDPCS presents a novel approach to UDA by leveraging the concept of disambiguation and introducing pseudo-candidate sets to approximate the true labels of the target set. Our method is distinguished by the use of two loss functions to refine predictions within the pseudo-candidate set and to penalize incorrect labels outside of it, combined with a threshold-based strategy to focus on ambiguous samples. Through comprehensive experiments on the benchmark datasets, our approach has demonstrated its effectiveness in consistently improving target set accuracy across diverse domains. We also conduct thorough analysis and ablation studies. The results highlight the potential of our method to serve as a robust solution for UDA challenges.

Acknowledgements

The research of AD is supported in part by the Prime Minister Research Fellowship, Ministry of Education, Government of India. VNB would like to acknowledge the support through the Govt of India SERB IMPRINT and DST ICPS funding programs for this work. The research of CKM is supported by the LiDAR and Camera Sensors Data based Deep Learning Algorithms for Autonomous Driving System project, funded by Govt. of India SERB program. The research is also partly supported by the Indo-Norwegian Collaboration in

Autonomous Cyber-Physical Systems (INCAPS) project: 287918 of the International Partnerships for Excellent Education, Research and Innovation (INTPART) program and the Low-Altitude UAV Communication and Tracking (LUCAT) project: 280835 of the IKTPLUSS program from the Research Council of Norway. We are grateful to the anonymous reviewers for the feedback that helped improved the presentation of this paper.

References

1. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine Learning **79**(1-2), 151–175 (2010)
2. Chen, C., Fu, Z., Chen, Z., Jin, S., Cheng, Z., Jin, X., Hua, X.S.: Homm: Higher-order moment matching for unsupervised domain adaptation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3422–3429 (2020)
3. Chen, L., Chen, H., Wei, Z., Jin, X., Tan, X., Jin, Y., Chen, E.: Reusing the task-specific classifier as a discriminator: Discriminator-free adversarial domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7181–7190 (2022)
4. Chen, W., Lin, L., Yang, S., Xie, D., Pu, S., Zhuang, Y.: Self-supervised noisy label learning for source-free unsupervised domain adaptation. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 10185–10192. IEEE (2022)
5. Chen, Y., Wei, C., Kumar, A., Ma, T.: Self-training avoids using spurious features under domain shift. Advances in Neural Information Processing Systems **33**, 21061–21071 (2020)
6. Csurka, G.: Domain adaptation for visual applications: A comprehensive survey, pp. 1–35. Springer (2017)
7. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space (2019)
8. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout (2017)
9. Du, Z., Li, J., Su, H., Zhu, L., Lu, K.: Cross-domain gradient discrepancy minimization for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3937–3946 (2021)
10. Feng, L., Lv, J., Han, B., Xu, M., Niu, G., Geng, X., An, B., Sugiyama, M.: Provably consistent partial-label learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 10948–10960. Curran Associates, Inc. (2020), `https://proceedings.neurips.cc/paper_files/paper/2020/file/7bd28f15a49d5e5848d6ec70e584e625-Paper.pdf`
11. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Proceedings of the 32nd International Conference on Machine Learning (ICML) (2015)
12. Gao, Z., Zhang, S., Huang, K., Wang, Q., Zhong, C.: Gradient distribution alignment certificates better adversarial domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8937–8946 (2021)
13. Jiang, J., Chen, B., Fu, B., Long, M.: Transfer-learning-library. `https://github.com/thuml/Transfer-Learning-Library` (2020)

14. Jin, X., Lan, C., Zeng, W., Chen, Z.: Re-energizing domain discriminator with sample relabeling for adversarial domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9174–9183 (2021)
15. Jin, X., Lan, C., Zeng, W., Chen, Z.: Minimum class confusion for versatile domain adaptation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
16. Jing, X., Qian, K., Jianu, T., Luo, S.: Unsupervised adversarial domain adaptation for sim-to-real transfer of tactile images. IEEE Transactions on Instrumentation and Measurement **72**, 1–11 (2023). https://doi.org/10.1109/TIM.2023.3268458
17. Kamilaris, A., Prenafeta-Boldú, F.X.: Deep learning in agriculture: A survey. Computers and Electronics in Agriculture **147**, 70–90 (2018)
18. Le, T., Nguyen, T., Ho, N., Bui, H., Phung, D.: Lamda: Label matching deep domain adaptation. In: International Conference on Machine Learning. pp. 6043–6054. PMLR (2021)
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
20. Lee, C.Y., Batra, T., Baig, M.H., Ulbricht, D.: Sliced wasserstein discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10285–10295 (2019)
21. Li, S., Liu, C., Lin, Q., Xie, B., Ding, Z., Huang, G., Tang, J.: Domain conditioned adaptation network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11386–11393 (2020)
22. Liu, H., Wang, J., Long, M.: Cycle self-training for domain adaptation. Advances in Neural Information Processing Systems **34**, 22968–22981 (2021)
23. Liu, X., Guo, Z., Li, S., Xing, F., You, J., Kuo, C.C.J., El Fakhri, G., Woo, J.: Adversarial unsupervised domain adaptation with conditional and label shift: Infer, align and iterate. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10367–10376 (2021)
24. Liu, Y., Zhou, Z., Sun, B.: Cot: Unsupervised domain adaptation with clustering and optimal transport. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19998–20007 (2023)
25. Lv, J., Xu, M., Feng, L., Niu, G., Geng, X., Sugiyama, M.: Progressive identification of true labels for partial-label learning. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 6500–6510. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/lv20a.html
26. Mei, K., Zhu, C., Zou, J., Zhang, S.: Instance adaptive self-training for unsupervised domain adaptation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16. pp. 415–430. Springer (2020)
27. Na, J., Jung, H., Chang, H.J., Hwang, W.: Fixbi: Bridging domain spaces for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1094–1103 (2021)
28. Nguyen, T., Le, T., Dam, N., Tran, Q.H., Nguyen, T., Phung, D.Q.: Tidot: A teacher imitation learning approach for domain adaptation with optimal transport. In: IJCAI. pp. 2862–2868 (2021)
29. Patel, V.M., Gopalan, R., Li, R., Chellappa, R.: Visual domain adaptation: A survey of recent advances. IEEE Signal Processing Magazine **32**(3), 53–69 (2015)
30. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1406–1415 (2019)

31. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge (2017)
32. Phan, H., Le, T., Phung, T., Bui, A.T., Ho, N., Phung, D.: Global-local regularization via distributional robustness. In: International Conference on Artificial Intelligence and Statistics. pp. 7644–7664. PMLR (2023)
33. Prabhu, V., Khare, S., Kartik, D., Hoffman, J.: Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8558–8567 (2021)
34. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural networks **12**(1), 145–151 (1999)
35. Rangwani, H., Aithal, S.K., Mishra, M., Jain, A., Radhakrishnan, V.B.: A closer look at smoothness in domain adversarial training. In: International Conference on Machine Learning. pp. 18378–18399. PMLR (2022)
36. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
37. Shen, D., Wu, G., Suk, H.I.: Deep learning in medical image analysis. Annual Review of Biomedical Engineering **19**, 221–248 (2017)
38. Shin, I., Woo, S., Pan, F., Kweon, I.S.: Two-phase pseudo label densification for self-training based domain adaptation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16. pp. 532–548. Springer (2020)
39. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 596–608. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/06964dce9addb1c5cb5d6e3d9838f733-Paper.pdf
40. Tian, Y., Yu, X., Fu, S.: Partial label learning: Taxonomy, analysis and outlook. Neural Networks **161**, 708–734 (2023). https://doi.org/https://doi.org/10.1016/j.neunet.2023.02.019, https://www.sciencedirect.com/science/article/pii/S0893608023000825
41. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
42. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5018–5027 (2017)
43. Wang, X., Zhuo, J., Zhang, M., Wang, S., Fang, Y.: Revisiting unsupervised domain adaptation models: A smoothness perspective. In: Proceedings of the Asian Conference on Computer Vision. pp. 1504–1521 (2022)
44. Wei, G., Lan, C., Zeng, W., Chen, Z.: Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16643–16653 (2021)
45. Wei, G., Lan, C., Zeng, W., Zhang, Z., Chen, Z.: Toalign: task-oriented alignment for unsupervised domain adaptation. Advances in Neural Information Processing Systems **34**, 13834–13846 (2021)

46. Wen, H., Cui, J., Hang, H., Liu, J., Wang, Y., Lin, Z.: Leveraged weighted loss for partial label learning. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 11091–11100. PMLR (18–24 Jul 2021), `https://proceedings.mlr.press/v139/wen21a.html`
47. Wu, D.D., Wang, D.B., Zhang, M.L.: Revisiting consistency regularization for deep partial label learning. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 24212–24225. PMLR (17–23 Jul 2022), `https://proceedings.mlr.press/v162/wu22l.html`
48. Xie, X., Chen, J., Li, Y., Shen, L., Ma, K., Zheng, Y.: Self-supervised cyclegan for object-preserving image-to-image domain adaptation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16. pp. 498–513. Springer (2020)
49. Yan, Y., Guo, Y.: Partial label unsupervised domain adaptation with class-prototype alignment. In: The Eleventh International Conference on Learning Representations (2022)
50. Zhang, Y., Liu, T., Long, M., Jordan, M.: Bridging theory and algorithm for domain adaptation. In: Proceedings of the 36th International Conference on Machine Learning (ICML) (2019)
51. Zhou, L., Ye, M., Zhu, X., Xiao, S., Fan, X.Q., Neri, F.: Homeomorphism alignment for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18699–18710 (2023)