Generalized Domain Adaptation

Yu Mitsuzumi Go Irie Daiki Ikami Takashi Shibata
NTT Communication Science Laboratories, NTT Corporation, Japan
{yu.mitsuzumi.ae, daiki.ikami.ef}@hco.ntt.co.jp, {goirie, t.shibata}@ieee.org

Abstract

Many variants of unsupervised domain adaptation (UDA) problems have been proposed and solved individually. Its side effect is that a method that works for one variant is often ineffective for or not even applicable to another, which has prevented practical applications. In this paper, we give a general representation of UDA problems, named Generalized Domain Adaptation (GDA). GDA covers the major variants as special cases, which allows us to organize them in a comprehensive framework. Moreover, this generalization leads to a new challenging setting where existing methods fail, such as when domain labels are unknown, and class labels are only partially given to each domain. We propose a novel approach to the new setting. The key to our approach is self-supervised class-destructive learning, which enables the learning of class-invariant representations and domain-adversarial classifiers without using any domain labels. Extensive experiments using three benchmark datasets demonstrate that our method outperforms the state-of-the-art UDA methods in the new setting and that it is competitive in existing UDA variations as well.

1. Introduction

Deep learning is data-hungry. It performs remarkably well in a domain that has a sufficient amount of labeled data, but its performance suffers significantly in one that does not. Unsupervised domain adaptation (UDA) aims to resolve this problem by transferring a model learned for a label-rich source domain to a label-less target domain.

Besides the standard UDA, which assumes a fully labeled source domain and a completely unlabeled target domain, a number of variants have been proposed to address more complex and practical problems. Major variants are illustrated in Fig. 1. One representative example is multi-source domain adaptation (MSDA) or multi-target domain adaptation (MTDA), an extension to the case where there is more than one source or target domain [45, 26, 12, 12]. Open set domain adaptation (OSDA) and partial domain adaptation (PDA) address the case where the class sets of the source and target do not match, i.e., there exist unknown classes [25, 32, 16, 2, 3]. Some extensions of these variants have also been studied [20, 7, 43].

Figure 1: **Schematic overview of Generalized Domain Adaptation (GDA)**. GDA covers major existing UDA problems as its special cases by imposing some constraints on classes and/or domains (represented in the shapes and colors of symbols, respectively). Moreover, it provides new challenging settings where domain labels are unknown, and class labels are given to only a subset in each domain.

Most of these variants have been proposed independently and solved individually. A negative side of this history is that a method that works for one variant may not work for or even be applicable to another. In reality, it is rarely possible to identify which variant of the problems one is facing, which requires a costly trial and error to figure out the type of problem or find a satisfactory solution. Moreover, a real problem is often a combination of these variants, in which case none of the methods will eventually be applicable.

In this work, we aimed to overcome this problem. Our approach is to first consider giving a new general representation of the UDA problems that covers all these major UDA variants and then implement a method for solving them. Definitions of the most existing UDA problems discussed above assume a clear distinction between the source and target domains, and whether or not class labels are available is determined on a domain-by-domain basis. Instead, in

our generalized representation, which we call Generalized Domain Adaptation (GDA), everything is determined on a sample-by-sample basis; each sample is given a class label, a domain label, and indexes indicating whether or not these labels are available. In Sec. 3, we will show that this slight difference in perspective allows us to represent all the major variants and their combinations as GDA's special cases.

Moreover, GDA brings a brand new challenge as illustrated in Fig. 1, which is not just a straightforward combination of the existing variants. The key properties of this setting are that the domain labels are completely unknown for all the samples, and the class labels are given to only a subset of the classes of each domain. Indeed, such a setting arises in practice, for example, for data coming from multiple institutions where the acquisition processes are unknown. Nonetheless, it has not received much attention so far. As shown later in our experiments (Sec. 5), the state-of-the-art UDA methods applicable to this setting, if forced, suffer from severe performance degradations.

We propose self-supervised class-destructive learning to accurately estimate the domain of each sample, which is essential missing ingredient to solve the new problem. The assumptions behind our approach are that (1) class information of an image is strongly dependent on its local structural information (e.g., shape and part connection); and that (2) domain and class information are independent of each other. Based on these assumptions, our method first transforms an image into a "class-indistinguishable" form by randomly shuffling the positions of its pixel blocks to break its local structure and then performs self-supervised learning to capture class-independent information. This enables learning of class-invariant and domain-variant representations without using any domain labels, making it possible to train a domain-invariant classifier with a simple domain-adversarial learning approach. Our method works even for the case where the class labels are only partially available for each domain. Furthermore, our method can readily cope with open set settings by integrating a joint label-network optimization framework [38]. Thorough experiments on three datasets demonstrate that our method outperforms the state-of-the-art methods in the new setting and is highly competitive in the existing UDA problems.

Our main contributions include: (1) a general representation of UDA problems named Generalized Domain Adaptation (GDA); (2) new UDA settings where existing methods fail; and (3) a novel domain label estimation method based on self-supervised class-destructive learning.

2. Related Work

Given a fully labeled source domain and an unlabeled target domain, the task of (standard) UDA is to train a classifier for the target domain by bridging the gap between the two domains (domain shift). Numerous methods have been proposed [40, 17, 18, 34, 35, 10, 11, 39], but most of them are based on strong assumptions that may not be feasible in practice – the number of domains is always two, which distribution each data sample comes from (i.e., domain labels) is known, and class sets of both domains are perfectly consistent. This has led to a growing stream of various UDA variants and attempts to develop more realistic and flexible methodologies. The major variants are as follows:

Multi-Source/Target UDA. Multi-source domain adaptation (MSDA) [21, 36, 45, 26] and multi-target domain adaptation (MTDA) [44, 12] consider UDA where either the source or target domain consists of multiple subdomains. While these studies assume that the subdomain labels of all samples are available, some others assumed that they are unavailable in the source [20] or target domain (Blending-target domain adaptation: BTDA) [7, 27].

Open Set and Partial UDA. These problems consider UDAs with "unknown" classes. Open set domain adaptation (OSDA) assumes that private classes exist in the target domain [32, 16, 2] or both [25]. In partial domain adaptation (PDA) [3, 4], the class set of the source is a superset of that of the target. Universal domain adaptation (UniDA) [43] is an integration of OSDA and PDA. Weakly supervised OSDA has also been studied [37]. Multi-source open set domain adaptation (MS-OSDA) is a combination of MSDA and OSDA [28]. A variant of MS-OSDA has also been investigated [42].

In this paper, we propose GDA, a general representation that covers all these major UDA variants. As described in the next section, GDA allows for a systematic discussion of the UDA variants that have been studied independently. It also reveals pivotal settings that have not been considered before and the missing parts essential to solve them.

3. Generalized Domain Adaptation

We first give the formal definition of our GDA.

Problem 1 (GDA) *Suppose we are given a set of tuples*

$$\mathcal{D} = \{(\mathbf{x}, y, d, \delta_y, \delta_d) | \mathbf{x} \in \mathbb{R}^n, y, d \in \mathbb{N}, \delta_y, \delta_d \in \{0, 1\}\},$$

where $\mathbf{x}$ *is a sample (e.g., image) of* n *dimensions,* y *is a class label,* d *is a domain label, and* δ_y *and* δ_d *are flags indicating whether the class and domain labels are available (1) or not (0) for training, respectively,*

The task is to find a class classifier F *that satisfies*

$$F(\mathbf{x}) = \begin{cases} y & (y \in \mathcal{L}) \\ \text{UNK} & (y \in \mathcal{C} \setminus \mathcal{L}), \end{cases}$$

where UNK *is the symbol for unknown classes, and*

$$\mathcal{C} = \{y | (\mathbf{x}, y, d, \delta_y, \delta_d) \in \mathcal{D}\},$$
$$\mathcal{L} = \{y | (\mathbf{x}, y, d, \delta_y, \delta_d) \in \mathcal{D}, \delta_y = 1\}.$$

Table 1: **GDA representations of existing and new UDA problems**. Problems and corresponding constraints are listed.

Problem	d	Class label set condition			δ_d
		Openness	Target domain	Source domain	
UDA	$d \in \{1,2\}$	$\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$	$\mathcal{L}_2 = \emptyset,\ \mathcal{U}_2 = \mathcal{C}_2$	$\mathcal{L}_1 = \mathcal{C}_1 = \mathcal{L},\ \mathcal{U}_1 = \emptyset$	$\forall \delta_d = 1$
MSDA	$d \in \mathbb{N}$	$\forall i,\ \mathcal{C}_i = \mathcal{C}$	$\exists! j,\ \mathcal{L}_j = \emptyset,\ \mathcal{U}_j = \mathcal{C}_j$	$\forall i \neq j,\ \mathcal{L}_i = \mathcal{C}_i = \mathcal{L},\ \mathcal{U}_i = \emptyset$	$\forall \delta_d = 1$
OSDA	$d \in \{1,2\}$	$\mathcal{C}_1 \subset \mathcal{C}_2 = \mathcal{C}$	$\mathcal{L}_2 = \emptyset,\ \mathcal{U}_2 = \mathcal{C}_2$	$\mathcal{L}_1 = \mathcal{C}_1 = \mathcal{L},\ \mathcal{U}_1 = \emptyset$	$\forall \delta_d = 1$
MS-OSDA	$d \in \mathbb{N}$	$\exists! j,\ \mathcal{C}_j = C,$ $\forall i \neq j,\ \mathcal{C}_i \subset \mathcal{C}_j$	$\exists! j,\ \mathcal{L}_j = \emptyset,\ \mathcal{U}_j = \mathcal{C}_j$	$\forall i \neq j,\ \mathcal{L}_i = \mathcal{C}_i = \mathcal{L},\ \mathcal{U}_i = \emptyset$	$\forall \delta_d = 1$
BTDA	$d \in \mathbb{N}$	$\forall i,\ \mathcal{C}_i = \mathcal{C}$	$\forall i \neq j,\ \mathcal{L}_i = \emptyset,\ \mathcal{U}_i = \mathcal{C}_i$	$\exists! j,\ \mathcal{L}_j = \mathcal{C}_j = \mathcal{L},\ \mathcal{U}_j = \emptyset$	$\delta_d = \begin{cases} 1 & (d = j) \\ 0 & (d \neq j) \end{cases}$
GDA1	$d \in \mathbb{N}$	$(\mathcal{L} \subseteq \mathcal{C})$	$\exists\, i, j,\ \mathcal{L}_i \neq \mathcal{L}_j,\ \forall k,\ \mathcal{L}_k \cap \mathcal{U}_k = \emptyset$		$\forall \delta_d = 0$
GDA2	$d \in \mathbb{N}$	$(\mathcal{L} \subseteq \mathcal{C})$	$\exists\, i, j,\ \mathcal{L}_i \neq \mathcal{L}_j,\ \forall k,\ \mathcal{L}_k \subset \mathcal{U}_k$		$\forall \delta_d = 0$

Intuitively, $\mathcal{C}$ is the set of all classes contained in $\mathcal{D}$, and $\mathcal{L}$ is the "known" (i.e., labeled) subset of $\mathcal{C}$. Given $\mathbf{x}$, F is required to output its correct class label y iff it is in $\mathcal{L}$, or to detect it as unknown otherwise. In GDA, the availability of the class and domain label is controlled on a sample-by-sample basis. This makes it possible to encompass new cases, for example, where the class labels are only available for a part of the classes or samples within the same domain, or the domain labels are not available at all.

3.1. Representing Existing UDA Problems

We show that GDA (Problem 1) turns into various existing UDA problems by imposing proper constraints. Representative examples are listed in Table 1. For brevity, we use $\mathcal{C}_i$, $\mathcal{L}_i$ and $\mathcal{U}_i$ for the set of all classes, known classes, and unknown classes within domain i, respectively.

$$\begin{aligned}
\mathcal{C}_i &= \{y | (\mathbf{x}, y, d, \delta_y, \delta_d) \in \mathcal{D}, d = i\}, \\
\mathcal{L}_i &= \{y | (\mathbf{x}, y, d, \delta_y, \delta_d) \in \mathcal{D}, d = i, \delta_y = 1\}, \\
\mathcal{U}_i &= \{y | (\mathbf{x}, y, d, \delta_y, \delta_d) \in \mathcal{D}, d = i, \delta_y = 0\}.
\end{aligned}$$

The standard UDA can be represented by limiting the number of domains to two ($d \in \{1,2\}$) sharing the same class set ($\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$), with the domain labels available for all the samples ($\forall \delta_d = 1$) and the class labels fully available for only those in one of the two domains ($\mathcal{L}_1 = \mathcal{C}_1 = \mathcal{L}$, $\mathcal{U}_1 = \emptyset$, $\mathcal{L}_2 = \emptyset$, $\mathcal{U}_2 = \mathcal{C}_2$). Discarding the condition on the number of domains ($d \in \mathbb{N}$) leads to multi-domain settings. For example, MSDA is represented by imposing constraints so that the class labels are unavailable in a certain domain ($\exists! j,\ \mathcal{C}_j = \mathcal{C},\ \mathcal{L}_j = \emptyset$) but fully accessible in the others ($\forall i \neq j,\ \mathcal{C}_i = \mathcal{C},\ \mathcal{L}_i = \mathcal{C}_i = \mathcal{L},\ \mathcal{U}_j = \emptyset$). OSDA [32, 16, 2] contains unknown classes in the target domain ($\mathcal{C}_1 \subset \mathcal{C}_2 = \mathcal{C}$). PDA [3, 4] and UniDA [43] can be written in the same way. Combinations and extensions of these problems, such as MS-OSDA [28] and BTDA [7, 27], can also be represented as in Table 1.

3.2. Deriving New UDA Problems

The systematic representations in Table 1 reveal that most of the existing UDA problems impose similar constraints of only limited types, which suggests a great potential to open up new UDA problems. In particular, all the UDA problems assume that the domain labels are visible ($\forall \delta_d = 1$) for at least one domain, which naturally reminds us to consider a "blind" domain adaptation problem where no domain labels are available for any domain, i.e., $\forall \delta_d = 0$. This is extremely challenging, especially when the labeled classes are inconsistent between the domains, i.e., $\exists\ i, j,\ \mathcal{L}_i \neq \mathcal{L}_j$, because no explicit information for disentangling the class and domain features is given.

Depending on whether the class labels are fully assigned to samples within the same class (i.e., $\forall k,\ \mathcal{L}_k \cap \mathcal{U}_k = \emptyset$) or only given to some of them (i.e., $\forall k,\ \mathcal{L}_k \subset \mathcal{U}_k$), we can define the following two versions of GDA:

Example A (GDA1)

$$\begin{aligned}
&\forall \delta_d = 0, \\
&\exists\, i, j,\ \mathcal{L}_i \neq \mathcal{L}_j,\ \forall k,\ \mathcal{L}_k \cap \mathcal{U}_k = \emptyset.
\end{aligned}$$

Example B (GDA2)

$$\begin{aligned}
&\forall \delta_d = 0, \\
&\exists\, i, j,\ \mathcal{L}_i \neq \mathcal{L}_j,\ \forall k,\ \mathcal{L}_k \subset \mathcal{U}_k.
\end{aligned}$$

Note that these problems do not make any assumptions on "openness"; they can be either a case where all the classes are known ($L = C$) or some of them are unknown ($L \subset C$).

None of the existing methods are directly applicable to these problems, as they all assume that the domain labels are known for at least one domain. They can be applied, if forced, by considering the labeled and unlabeled data (across multiple domains) as the "source" and "target", respectively. However, as shown later in Sec. 5, the performance is far from satisfactory.

4. Method

We propose an approach to solving the new problems. An overview is shown in Fig. 2. Our method comprises two major steps: estimating the domain labels for all samples and learning a domain-invariant classifier using the estimated domain labels.

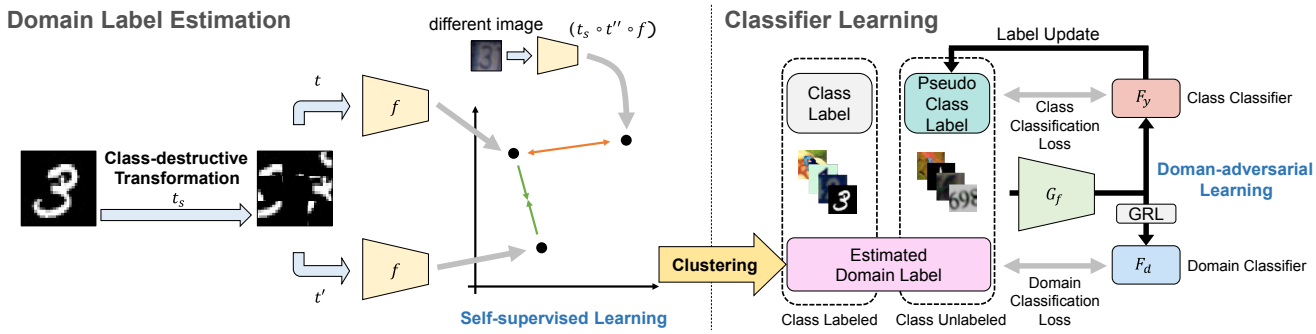


Figure 2: **Overview of our approach to GDA**. We first apply class-destructive transformation to images of unknown domains and then perform self-supervised learning on them to estimate their domain labels. We then perform domain-adversarial learning with the estimated domain labels to learn a domain-invariant classifier.

4.1. Domain Label Estimation

One primary source of the difficulty of these problems is that the domain labels are unknown for all images. The first step, which is the core of our proposed method, estimates the domain labels in a self-supervised learning manner.

Self-supervised Class-destructive Learning. A straightforward approach to estimating the domain labels would be to assume that images belonging to the same domain form a tight cluster in their feature space and to estimate their domain labels by feature clustering. In fact, an existing approach to BTDA [7] is based on a similar assumption to estimate the (sub)domain labels hidden in the target domain. Unfortunately, the results are not as expected; the classes and domains are easily confused in the new problems where the class labels are only given to a subset of images, resulting in the formation of class-dependent clusters.

We aim to prevent the formation of such class-dependent clusters by applying image transformation that aggressively disrupts the class-dependent information of images and using the transformed images for self-supervised feature learning. Specifically, we focus on local structural information of images which is an important clue representing the class of images, such as the shape of an object or the connections between its parts, and destroy it by using a transformation that divides the original image into several pixel blocks and randomly shuffles their positions. Fig. 3 shows examples of transformed images at different pixel block sizes. When the partition is coarse, the shape of the object and its parts are still recognizable. However, as the number of partitions is increased, even the parts are fragmented and become unrecognizable; meanwhile, it is relatively easy to identify which domain each image comes from owing to the remaining global information of the pixels.

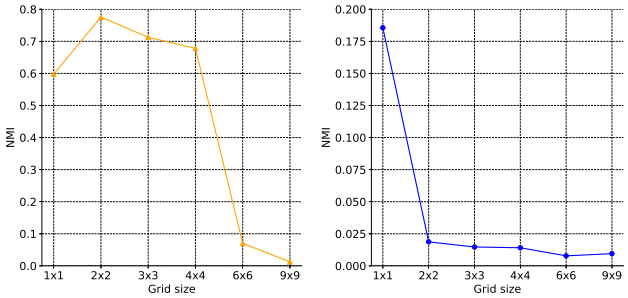
This idea is supported by a quick analysis. We measure the consistency between the clusters of the features learned on the transformed images (by the self-supervised learning algorithm described in the next paragraph) and the ground truth class/domain labels by normalized mutual information (NMI). The results are shown in Fig 4. As the number of grid partitions increases, NMI to the domain labels becomes high within a certain range, while that to the class labels sharply decreases. These results show that by taking an appropriate number of grid partitions, domain-variant features that are invariant to the class information can be learned.

Algorithm. Our algorithm is simple. We first apply the class-destructive transformation to the training images to obtain the transformed images and then perform self-supervised learning on these images to obtain class-invariant features. Finally, we apply clustering to the learned features to assign a domain label to each cluster.

We use [6] for self-supervised learning. A feature extractor f is learned by minimizing the normalized temperature-scaled cross-entropy loss [6] (with the temperature of 0.5) between the features of two images augmented from the same image. The two images are generated as $t(t_s(\mathbf{x}))$ and $t'(t_s(\mathbf{x}))$, where t and t' are two random augmentation operations (e.g., random crop) and t_s is the class-destructive transformation. After training, we estimate the domain labels for all the images by applying clustering to their features. We use a Gaussian mixture for clustering.

Discussion. A relevant idea to our approach is "jigsaw", a popular pretext task for self-supervised learning [24], which has also been applied to UDA and domain generalization [1, 5]. Our work differs from them in both method and purpose. These methods aim to learn *class-dependent* features like object shape and part connections by letting the network solve the task of undoing shuffled images in pixel blocks. Our method, in contrast, aims to learn *class-independent* features and therefore does not solve the jigsaw task. Technically, this is a slight difference, but it has the exact opposite effect. More recently, a self-supervised learning method for learning pretext invariant features has been proposed [22]. Although it does not address UDA, our approach is somewhat similar to it. In this work,

Figure 3: **Examples of transformed images**. Two "mug" images of different domains in Office-31. The object class is harder to recognize as the number of grid partitions is increased, but their domains are still identifiable.



(a) Domain NMI (b) Class NMI

Figure 4: **Impact of class-destructive transformation**. NMI between the clusters of the features learned on the transformed images at different grid partitions and the ground truth domain labels (a) and class labels (b). SVHN and SynDigits are used.

we show that the class-dependent information is corrupted more quickly than the domain-dependent information as the pixel block size is decreased, and, based on this finding, we propose a simple self-supervised approach to learning class-independent and domain-sensitive features to solve UDA with unknown domain labels. To the best of our knowledge, these points have never been explored before. Another recent work uses self-supervised clustering for UniDA [30]. Unlike ours, it assumes that the domain labels are known.

4.2. Classifier Learning

The second step trains our class classifier network. Once the domain labels are estimated, we can train the classifier through domain-adversarial classifier learning.

Domain-adversarial Classifier Learning. We follow a standard domain-adversarial learning approach [10] to train our classifier network. The network consists of three major parts: a (shared) feature extractor G_f, a class label predictor F_y, and a domain classifier F_d. Our class classifier is defined as $F = G_f \circ F_y$. An image $\mathbf{x}$ is fed to G_f to extract the feature and then mapped by F_y to the class label y and by F_d to the domain label d. With the aim of learning features discriminative to the classes but domain-invariant, the entire network is trained by solving the following problems:

$$\begin{aligned} &\min_{G_f, F_y} \ \mathcal{L}_y - \lambda \mathcal{L}_d, \\ &\min_{F_d} \ \mathcal{L}_d, \end{aligned} \tag{1}$$

where $\mathcal{L}_y$ is a class classification loss and $\mathcal{L}_d$ is a domain classification loss. We use softmax cross-entropy for both, with the ground truth class labels and the domain labels estimated by our domain label estimation method. The two problems can be efficiently minimized at the same time by using the gradient reversal layer (GRL) [10].

Handling Unknown Classes and Unlabeled Samples. So far, we considered the case where neither unknown classes nor unlabeled samples of known classes exist. In our new problems, however, both exist. Let us assume that all the unlabeled samples are of unknown classes, and 'UNK' (the class label for unknown classes) is assigned to all. In reality, some of them are truly of unknown classes, while others are actually of known ones. We treat this problem as "learning with noisy labels" and solve it by estimating the true class labels of all the unlabeled samples and optimizing the network parameters simultaneously.

Our approach is based on the idea of joint label-network optimization [38]. We first train an initial class classifier F by solving the problem Eq. (1) using only the labeled samples and then assign an initial pseudo class label to each unlabeled sample $\mathbf{x}$ using the initial classifier. Our assumption is that if an unlabeled sample $\mathbf{x}$ belongs to a known class $y \in \mathcal{L}$, the class probability of the corresponding class predicted by the initial classifier is high, and if it belongs to an unknown class $y \in \mathcal{C} \setminus \mathcal{L}$, the probabilities are comparable between the classes. With this assumption, we determine the initial pseudo class label y of $\mathbf{x}$ based on the entropy of the class probability. Specifically,

$$y = \begin{cases} \text{UNK} & (H(y|\mathbf{x}) > \sigma), \\ \underset{k}{\operatorname{argmax}}\, F(\mathbf{x})[k] & (\text{otherwise}), \end{cases} \tag{2}$$

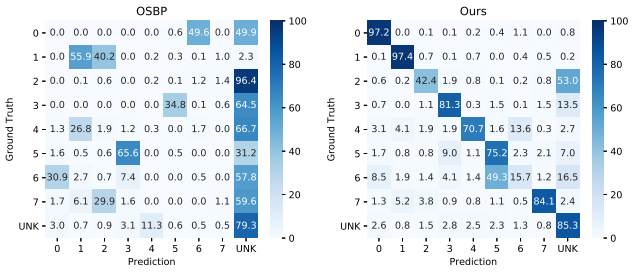
where $F(\mathbf{x})[k]$ gives the class probability of the k-th class. $H(y|\mathbf{x})$ is the entropy, i.e., $H(y|\mathbf{x}) = \sum_k F(\mathbf{x})[k] \log F(\mathbf{x})[k]$, and σ is a threshold. Given the initial pseudo class labels, we update the network parameters on both labeled and unlabeled samples by solving Eq. (1) with the additional regularization term $\mathcal{L}_p$ in [38] used to prevent the assignment of all labels to a single class. Note that we add one extra output node so that the network can directly output the probability of UNK. Following [38], we update the estimated class labels for unlabeled samples as $y = \operatorname{argmax}_k F(\mathbf{x})[k]$ during training.

Table 2: **Results for Digits in GDA1**. The best and second-best are highlighted in bold and underlined, respectively.

	sv(0-3), sy(4-7)			sv(0-3), mt(4-7)			sv(0-2), sy(3-5), mt(6-8)			sv(0,1), sy(2,3), mt(4,5), mm(6,7)			sv(0-5), sy(2-7)			sv(0-5), mt(2-7)			Avg.		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
Labeled Only	69.50	-	-	20.82	-	-	42.95	-	-	42.31	-	-	71.32	-	-	25.71	-	-	45.44	-	-
MCD [31]	84.12	-	-	23.17	-	-	48.07	-	-	41.07	-	-	79.61	-	-	24.77	-	-	50.14	-	-
OSBP [32]	64.33	84.91	73.20	7.37	79.32	13.49	33.95	60.84	43.58	19.48	66.47	30.13	33.15	70.53	45.10	10.93	80.33	19.24	28.20	73.73	37.46
ROS [2]	43.62	86.98	58.10	5.10	60.23	9.40	18.15	67.07	28.57	15.36	79.40	25.74	50.90	80.37	62.33	0.67	56.04	1.32	22.30	71.68	30.91
UAN [43]	61.50	0.00	0.00	14.50	0.70	1.34	39.96	0.92	1.80	33.89	0.00	0.00	65.72	6.55	11.91	24.90	2.21	4.06	40.08	1.73	3.18
Ours	86.18	91.19	**88.61**	70.50	85.31	**77.20**	79.76	84.08	**81.86**	66.02	79.97	**72.33**	84.83	85.75	**85.29**	73.46	76.92	**75.15**	76.79	83.87	**80.07**

Table 3: **Results for Office-31 in GDA1**. The best and second-best are highlighted in bold and underlined, respectively.

	D(0-9), W(10-19)			A(0-9), D(10-19)			W(0-9), A(10-19)			Avg.		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
Labeled Only	93.25	-	-	30.91	-	-	41.35	-	-	55.17	-	-
MCD [31]	86.15	-	-	39.55	-	-	23.43	-	-	49.71	-	-
OSBP [32]	95.56	87.91	91.58	0.42	99.91	0.84	3.56	99.84	6.87	33.18	95.89	33.10
ROS [2]	93.95	91.65	**92.79**	38.57	74.92	50.92	36.82	74.20	49.22	56.45	80.26	64.31
UAN [43]	98.03	42.64	59.43	6.42	28.17	10.46	51.72	32.13	39.64	52.06	34.31	36.51
Ours	91.71	82.19	86.69	76.64	75.33	**75.98**	76.87	84.28	**80.40**	81.74	80.60	**81.02**



(a) OSBP (b) Ours

Figure 5: **Confusion matrices**. The results of OSBP and Ours in "sv(0-3), mt(4-7)" are reported.

5. Experiments

We first show the results on the new problems, GDA1 and GDA2, in Sec. 5.1 and then those on the three existing problems covered by GDA, i.e., OSDA, MS-OSDA and BTDA, in Sec. 5.2.

5.1. Experiments on New UDA Problems

We show the results in GDA1 and GDA2. In summary, our method outperforms existing methods.

Datasets. We use Digits and Office-31. Digits is composed of four digit datasets: SVHN (sv) [23], SynDigits (sy) [10], MNIST (mt) [15], and MNIST-M (mm) [10]. Following [7], we use 25,000 samples in sv, sy, and mt for training and 9,000 samples for testing. For mm, we use 21,661 samples for training and 7,854 samples for testing. Office-31 [29] is composed of three domains: Amazon (A), DSLR (D), and Webcam (W) having 4,110 samples of 31 object classes.

We use several different settings of the combinations of domains and splits of labeled, unlabeled (known), and unknown classes. An example is "sv(0-3), sy(4-7)" which means that only the sv and sy domains are used, where {"0", "1", "2", "3"} in sv and {"4", "5", "6", "7"} in sy are labeled and the rest remain unlabeled; {"8", "9"}, which are not labeled in either domain, are treated as unknown. For GDA2, we follow the same settings as GDA1 but leave half of the samples in each labeled class unlabeled.

Table 4: **Results for Digits in GDA2**. The best and second-best are highlighted in bold and underlined, respectively.

	sv(0-3), mt(4-7)			sv(0-2), sy(3-5), mt(6-8)			sv(0,1), sy(2,3), mt(4,5), mm(6,7)			Avg.		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [32]	52.45	86.32	65.25	55.29	26.24	35.59	41.33	79.41	54.37	49.69	63.99	51.74
ROS [2]	0.20	68.95	0.40	12.45	89.34	21.85	8.15	81.15	14.81	6.93	79.81	12.36
UAN [43]	56.96	0.17	0.34	59.86	0.96	1.89	48.99	0.00	0.00	55.27	0.38	0.74
Ours	76.95	84.54	**80.57**	77.26	74.51	**75.86**	69.67	78.18	**73.68**	74.63	79.08	**76.70**

Table 5: **Results for Office-31 in GDA2**. The best and second-best are highlighted in bold and underlined, respectively.

	D(0-9), W(10-19)			A(0-9), D(10-19)			W(0-9), A(10-19)			Avg.		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [32]	93.57	92.96	**93.26**	46.20	77.75	57.96	51.63	78.66	62.34	63.80	83.12	71.19
ROS [2]	92.35	90.77	91.55	40.32	71.16	51.47	27.69	76.85	40.71	53.45	79.59	61.25
UAN [43]	97.24	46.81	63.20	65.96	23.58	34.74	65.83	28.15	39.44	76.34	32.85	45.79
Ours	90.33	93.41	91.84	72.44	65.08	**68.56**	72.99	83.89	**78.06**	78.59	80.79	**79.49**

Evaluation Metrics and Baselines. We use three evaluation metrics: OS* and UNK, which are the average accuracy on the known and unknown classes, respectively; and HOS, which is the harmonic mean of OS* and UNK. Although no existing methods are directly applicable to GDA1 and GDA2, we forcibly applied the following methods as baselines by considering the labeled data as the source domain and the unlabeled data as the target domain: Labeled Only (trained with the labeled data only), MCD [31], OSBP [32], ROS [2], and UAN [43].

Implementation Details. For domain label estimation, we use a two-layer MLP followed by a four-layer CNN for our feature extractor f (see supplementary material for details). We use the 3×3 grid for our class-destructive transformation unless otherwise noted. We use random crop and grayscale for our data augmentation. We also use Gaussian blur and the operations used in [8] for Digits. The network is optimized by using Adam [14] with the learning rate of 1.0×10^{-3} for 300 epochs on Office-31 and 80 epochs on Digits with the batch size of 512.

We use a different classifier for each dataset. For Digits, we use a four-layer CNN for our feature extractor G_f and two-layer MLPs for our class label predictor F_y and domain classifier F_d (see supplementary material for detail). For Office-31, we use ResNet-50 [13] pretrained on ImageNet with the top classification layers discarded for G_f, one fully connected layer for F_y, and a three-layer MLP ($1024 \rightarrow 1024 \rightarrow$ #domains) for F_d. The network is trained

Table 6: **Results for Office-31 in OSDA**. The best and second-best are highlighted in bold and underlined, respectively.

	A→W			A→D			D→W			W→D			D→A			W→A			**Avg.**		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
STA_{sum} [16]	92.1	58.0	71.0	95.4	45.5	61.6	97.1	49.7	65.5	96.6	48.5	64.4	94.1	55.0	69.4	92.1	46.2	60.9	94.6	50.5	65.5
STA_{max} [16]	86.7	67.6	75.9	91.0	63.9	75.0	94.1	55.5	69.8	84.9	67.8	75.2	83.1	65.9	73.2	66.2	68.0	66.1	84.3	64.8	72.5
OSBP [32]	86.8	79.2	82.7	90.5	75.5	**82.4**	97.7	96.7	97.2	99.1	84.2	91.1	76.1	72.3	75.1	73.0	74.4	73.7	87.2	80.4	83.7
UAN [43]	95.5	31.0	46.8	95.6	24.4	38.9	99.8	52.5	68.8	81.5	41.4	53.0	93.5	53.4	68.0	94.1	38.8	54.9	93.4	40.3	55.1
ROS [2]	88.4	76.7	82.1	87.5	77.8	**82.4**	99.3	93.0	96.0	100.0	99.4	**99.7**	74.8	81.2	77.9	69.7	86.6	77.2	86.6	85.8	**85.9**
Ours	82.5	84.0	**83.2**	73.8	83.2	77.9	96.7	98.0	**97.4**	99.1	96.1	97.6	67.8	92.4	**78.2**	70.8	85.1	**77.3**	81.8	89.8	85.3

Table 7: **Results for Office-Home in OSDA**. The best and second-best are highlighted in bold and underlined, respectively.

	Pr→Rw			Pr→Cl			Pr→Ar			Ar→Pr			Ar→Rw			Ar→Cl		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
STA_{sum} [16]	78.1	66.3	69.7	44.7	71.5	**55.0**	55.4	73.7	63.1	68.7	59.7	63.7	81.1	50.5	62.1	50.8	63.4	56.3
STA_{max} [16]	76.2	64.3	69.5	44.2	67.1	53.2	54.2	72.4	61.9	68.0	48.4	54.0	78.6	60.4	68.3	46.0	72.3	55.8
OSBP [32]	76.2	71.7	73.9	44.5	66.3	53.2	59.1	68.1	**63.2**	71.8	59.8	65.2	79.3	67.5	72.9	50.2	61.1	55.1
UAN [43]	84.0	0.1	0.2	59.1	0.0	0.0	73.7	0.0	0.0	81.1	0.0	0.0	88.2	0.1	0.2	62.4	0.0	0.0
ROS [2]	70.8	78.4	**74.4**	46.5	71.2	56.3	57.3	64.3	60.6	68.4	70.3	**69.3**	75.8	77.2	**76.5**	50.6	74.1	**60.1**
Ours	65.0	78.1	70.9	49.3	71.7	**58.4**	49.6	78.2	60.7	61.5	74.5	67.4	69.5	80.3	74.5	50.1	74.7	59.9

	Rw→Ar			Rw→Pr			Rw→Cl			Cl→Rw			Cl→Ar			Cl→Pr			**Avg.**		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
STA_{sum} [16]	67.9	62.3	65.0	77.9	58.0	66.4	51.4	57.9	54.2	69.8	63.2	66.3	53.0	63.9	57.9	61.4	63.5	62.5	63.4	62.6	61.9
STA_{max} [16]	67.5	66.7	67.1	77.1	55.4	64.5	49.9	61.1	54.5	67.0	66.7	66.8	51.4	65.0	57.4	61.8	59.1	60.4	61.8	63.3	61.1
OSBP [32]	66.1	67.3	66.7	76.3	68.6	72.3	48.0	63.0	54.5	72.0	69.2	70.6	59.4	70.3	**64.3**	67.0	62.7	64.7	64.1	66.3	64.7
UAN [43]	77.5	0.0	0.2	85.0	0.1	0.1	66.2	0.0	0.0	80.6	0.1	0.2	70.5	0.0	0.0	74.0	0.1	0.2	75.2	0.0	0.1
ROS [2]	67.0	70.8	**68.8**	72.0	80.0	**75.7**	51.5	73.0	**60.4**	65.3	72.2	68.6	53.6	65.5	58.9	59.8	71.6	65.2	61.6	72.4	**66.2**
Ours	54.8	81.6	65.6	69.5	78.9	73.8	53.0	72.7	61.3	64.3	78.9	**70.8**	47.1	80.8	59.5	60.5	74.5	**66.8**	57.8	77.1	65.8

Table 8: **Results for Office-31 in MS-OSDA**. The best and second-best are highlighted in bold and underlined, respectively.

	A, D → W		A, W → D		W, D → A		Avg.	
	OS*	OS	OS*	OS	OS*	OS	OS*	OS
OSVM [33]	71.2	51.2	84.9	56.2	58.2	61.4	71.4	56.3
OSVM+DANN [10]	65.0	83.3	68.0	91.9	51.2	37.5	61.4	70.9
OSBP [32]	94.0	90.0	93.0	89.0	79.0	75.0	88.7	84.7
IOSBP [9]	91.1	88.0	87.8	87.1	75.0	74.5	84.6	83.2
MOSDANET [28]	**99.0**	**98.2**	**99.4**	**98.3**	**81.0**	**79.3**	**93.1**	**91.9**
Ours	94.7	94.1	94.1	94.5	76.1	75.6	88.3	88.1

Table 9: **Results for Office-31 in BTDA**. The best and second-best are highlighted in bold and underlined, respectively.

Models	A → D, W	D → W, A	W → A, D	Avg.
Labeled only	68.6	70.0	66.5	68.4
DAN [17]	78.0	64.4	66.7	69.7
RTN [18]	84.3	67.5	64.8	72.2
JAN [19]	84.2	74.4	72.0	76.9
RevGrad [10]	78.2	72.2	69.8	73.4
AMEAN [7]	**90.1**	**77.0**	**73.4**	**80.2**
Ours	88.8	74.5	73.2	78.8

for 1000 epochs using SGD with the initial learning rate of 1.0×10^{-3}, a momentum of 0.9, and a weight decay of 5.0×10^{-4}. The batch size is 256 for Digits and 16 for Office-31. The learning rate is changed to 1.0×10^{-1} at the 100th epoch for Digits. The pseudo class labels are initialized at the 100th epoch and updated at the 200th epoch (see Sec. 4.2). As in [11], λ in Eq. 1 is scheduled as $\lambda = \frac{2}{1+\exp[-\gamma \dot{p}]} - 1$, where p increases linearly from 0 to 1 during training. We set $\gamma = 1000$ for Digits and $\gamma = 10$ for Office-31. We set σ in Eq. (2) to the median of the entropy within the mini-batch. We use the true class distribution as the prior for the regularization term $\mathcal{L}_p$ as done in [38]. Note that performance degradation is not severe if the true distribution is unknown (see supplementary material).

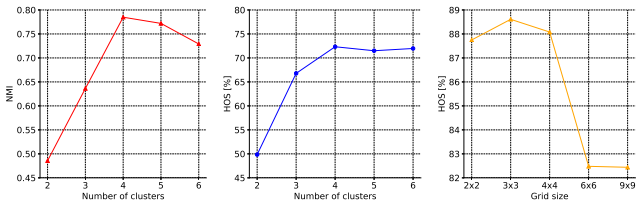
Results. Tables 2 and 3 show the results in GDA1. First, all the baselines fail to achieve adequate performance in many cases. We found noticeable decreases in performance for the domains with large domain gaps. For example, the HOS values of all the baselines cannot be better than 15% in "sv(0-3), mt(4-7)". In "W(0-9), A(10-19)", even the best-performing method, ROS, achieves only 49% in HOS. These results suggest the difficulty of this new problem. Our method yields adequate performance in all the cases and outperforms all the baselines in most cases; the HOS values of our method exceed 77% in "sv(0-3), mt(4-7)" and 80% in "W(0-9), A(10-19)". These results show the exceptional effectiveness of our method in this problem.

Fig. 5 shows confusion matrices for "sv(0-3), mt(4-7)" of OSBP (the best competitor in this case) and Ours. OSBP tends to misclassify {"0", "1", "2", "3"} to {"4", "5", "6", "7"}, and vice versa (other baselines also show the same tendency). This implies that the baselines fail to disentangle the class and domain information, which is the reason for their poor performance. Our method does not suffer from this problem, which emphasizes its effectiveness.

Tables 4 and 5 show the results for GDA2. Similar to GDA1, the performance of the baselines is unsatisfactory. Ours outperforms all of them in all the settings except for "D(0-9), W(10-19)", which proves that Ours is effective even when the class labels are given to only a subset of the samples in the same class.

5.2. Experiments on Existing UDA Problems

We show the results in OSDA, MS-OSDA, and BTDA for the protocols used in [2, 28, 7], respectively. Overall, ours is competitive with the state-of-the-art methods, which confirms that it is applicable even when the type of UDA



(a) (b) (c)

Figure 6: **Sensitivity to hyperparameters**. (a) NMI and (b) HOS vs. the number of clusters in "sv(0,1), sy(2,3), mt(4,5), mm(6,7)". (c) HOS vs. the grid size of class-destractive transformations in "sv(0-4), sy(4-7)".

problem cannot be identified in advance.

Datasets. We use Office-31 and Office-Home. The details of Office-31 are given in Sec. 5.1. Office-Home [41] contains 15,500 samples of 65 object classes of four domains: Art (Ar), Clipart (Cl), Product (Pr), and Real-World (Rw).

For OSDA, we follow the protocol used in [2]. For Office-31, the first 10 and the last 11 classes in alphabetical order are used as known and unknown classes, respectively. For Office-Home, the first 25 and the other 40 are used as known and unknown classes, respectively. We use Office-31 for MS-OSDA [28] and BTDA [7]. For MS-OSDA, we consider all possible combinations of two source domains and one target domain. The first 20 and the other 11 classes in alphabetical order are used as shared and open set classes, respectively. For BTDA, we consider all possible combinations of one source domain and two target domains.

Evaluation Metrics and Baselines. We use OS*, UNK, and HOS for OSDA as done in [2]. For MS-OSDA, we use OS* and OS [28]; OS is the average accuracy both on known and unknown classes. For BTDA, we use classification accuracy [7]. Here, we compare our method with the methods used in the corresponding papers [2, 28, 7].

Implementation Details. We use the same networks as in Sec 5.1 for both domain label estimation and classification. We train the classifier for 200 epochs using SGD with the mini-batch size of 32, momentum of 0.9, and weight decay of 5.0×10^{-4}. The learning rate is initialized to 1.0×10^{-2} for F_y, F_d and the last layer of G_f and to 1.0×10^{-3} for the rest of G_f. It is updated at every iteration as $\mathrm{lr} \leftarrow \mathrm{lr}_{\mathrm{init}} \times (1 + 0.001 \times i)^{-0.75}$, where i is the number of iterations. The pseudo class labels are initialized at the 40th epoch and updated at the 80th epoch. We use the same scheduling policy for λ as in Sec. 5.1 but increase p linearly from 0 to 1 until the 1000th iteration with $\gamma = 1.0$.

Results for OSDA. Tables 6 and 7 show the comparative results for OSDA. Despite the fact that our method is not designed specifically for OSDA, it is highly competitive with the state-of-the-art methods. Compared with ROS, our best competitor, the differences in HOS are only 0.6% for Office-31 and 0.4% for Office-Home on average.

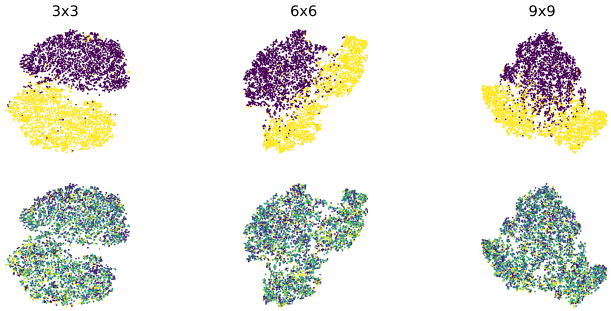


Figure 7: **t-SNE visualization**. The features learned by our domain label estimation method in "sv(0-4), sy(4-7)" are visualized. The points are color-coded by their domain labels (top) and class labels (bottom).

Results for MS-OSDA. Table 8 shows the results. Although Ours is worse than MOSDANET [28], the state-of-the-art in MS-OSDA, it is the second-best in most cases.

Results for BTDA. Table 9 shows the results. Similar to the results in MS-OSDA, Ours is slightly worse than AMEAN [7] but clearly better than the others.

5.3. Analysis

Number of Clusters for Domain Label Estimation. We evaluate the sensitivity to the number of clusters for domain label estimation. We use "sv(0,1), sy(2,3), mt(4,5), mm(6,7)", where the true number of domains is four. The domain label estimation quality in NMI and HOS are shown in Figs. 6a and 6b, respectively. Both reach a maximum at four (the true number of domains), and the HOS value is stable for higher numbers, suggesting that it is not severely sensitive to the number of clusters.

Grid Size for Class-destructive Transformation. Fig. 6c reports the performance of our method for various grid sizes in "sv(0-4), sy(4-7)". The accuracy is satisfactory when the partition is coarse but decreases as it becomes finer. This is not surprising because the domain information is destroyed by over-partitioning, as discussed in Sec. 4.1. We show a t-SNE visualization of the features learned by our domain label estimation in Fig 7, which supports the results.

6. Conclusion

We proposed Generalized Domain Adaptation (GDA), a general representation of UDA that covers the major UDA variants as well as new challenging settings where existing UDA methods fail. To solve the new settings, we proposed a novel self-supervised class-destructive learning approach, which estimates domain labels without using any supervision. Experiments demonstrated that our method outperforms the state-of-the-art methods in the new settings and that it is highly competitive on existing UDA variants.

References

[1] Silvia Bucci, Antonio D'Innocente, and Tatiana Tommasi. Tackling partial domain adaptation with self-supervision. In *Proc. ICIAP*, 2019. 4

[2] Silvia Bucci, Mohammad Reza Loghmani, and Tatiana Tommasi. On the effectiveness of image rotation for open set domain adaptation. In *Proc. ECCV*, 2020. 1, 2, 3, 6, 7, 8

[3] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proc. ECCV*, 2018. 1, 2, 3

[4] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang. Learning to transfer examples for partial domain adaptation. In *Proc. CVPR*, 2019. 2, 3

[5] Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proc. CVPR*, 2019. 4

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 4

[7] Ziliang Chen, Jingyu Zhuang, Xiaodan Liang, and Liang Lin. Blending-target domain adaptation by adversarial meta-adaptation networks. In *Proc. CVPR*, 2019. 1, 2, 3, 4, 6, 7, 8

[8] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *Proc. ICLR*, 2018. 6

[9] Jiahui Fu, Xiaofu Wu, Suofei Zhang, and Jun Yan. Improved open set domain adaptation with backpropagation. In *Proc. ICIP*, 2019. 7

[10] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *Proc. ICML*, 2015. 2, 5, 6, 7

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016. 2, 7

[12] Behnam Gholami, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic. Unsupervised multi-target domain adaptation: An information theoretic approach. *IEEE TIP*, 29:3993–4002, 2020. 1, 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 6

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[15] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 6

[16] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In *Proc. CVPR*, 2019. 1, 2, 3, 7

[17] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. *Proc. ICML*, 2015. 2, 7

[18] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *Proc. NeurIPS*, 2016. 2, 7

[19] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proc. ICML*, 2017. 7

[20] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *Proc. CVPR*, 2018. 1, 2

[21] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Proc. NeurIPS*, 2009. 2

[22] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proc. CVPR*, 2020. 4

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Proc. NeurIPS*, 2011. 6

[24] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, 2016. 4

[25] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proc. ICCV*, 2017. 1, 2

[26] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proc. ICCV*, 2019. 1, 2

[27] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *Proc. ICML*, 2019. 2, 3

[28] Sayan Rakshit, Dipesh Tamboli, Pragati Shuddhodhan, Biplab Banerjee Meshram, Gemma Roig, and Subhasis Chaudhuri. Multi-source open-set deep adversarial domain adaptation. In *Proc. ECCV*, 2020. 2, 3, 7, 8

[29] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proc. ECCV*, 2010. 6

[30] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self-supervision. In *Proc. NeurIPS*, 2020. 5

[31] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proc. CVPR*, 2018. 6

[32] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proc. ECCV*, 2018. 1, 2, 3, 6, 7

[33] Walter J. Scheirer, Lalit P. Jain, and Terrance E. Boult. Probability models for open set recognition. *IEEE TPAMI*, 36(11):2317–2324, 2014. 7

[34] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proc. AAAI*, 2016. 2

[35] Baochen Sun and Kate Saenko. Deep CORAL: Correlation alignment for deep domain adaptation. In *Proc. ECCV*, 2016. 2

[36] Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. A two-stage weighting framework for multi-source domain adaptation. In *Proc. NeurIPS*, 2011. 2

[37] Shuhan Tan, Jiening Jiao, and Wei-Shi Zheng. Weakly supervised open-set domain adaptation by dual-domain collaboration. In *Proc. CVPR*, 2019. 2

[38] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proc. CVPR*, 2018. 2, 5, 7

[39] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. CVPR*, 2017. 2

[40] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 2

[41] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proc. CVPR*, 2017. 8

[42] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proc. CVPR*, 2018. 2

[43] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proc. CVPR*, 2019. 1, 2, 3, 6, 7

[44] Huanhuan Yu, Menglei Hu, and Songcan Chen. Multi-target unsupervised domain adaptation without exactly shared categories. *arXiv preprint arXiv:1809.00852*, 2018. 2

[45] Han Zhao, Shanghang Zhang, Guanhang Wu, José M. F. Moura, Joao P. Costeira, and Geoffrey J. Gordon. Adversarial multiple source domain adaptation. In *Proc. NeurIPS*, 2018. 1, 2