Gradual Domain Adaptation without Indexed Intermediate Domains

Anonymous Author(s)
Affiliation
Address
email

Abstract

The effectiveness of unsupervised domain adaptation degrades when there is a large discrepancy between the source and target domains. Gradual domain adaption (GDA) is one promising way to mitigate such an issue, by leveraging additional unlabeled data that gradually shift from the source to the target. Through sequentially adapting the model along the "indexed" intermediate domains, GDA substantially improves the overall adaptation performance. In practice, however, the extra unlabeled data may not be separated into intermediate domains and indexed properly, limiting the applicability of GDA. In this paper, we investigate how to discover the domain sequence when it is not already available. Concretely, we propose a coarse-to-fine framework, which starts with a *coarse* domain discovery step via progressive domain discriminator training. This coarse domain sequence then undergoes a *fine* indexing step via a novel cycle-consistency loss, which encourages the next intermediate domain to preserve sufficient discriminative knowledge of the current intermediate domain. The resulting domain sequence can then be used by a GDA algorithm. On benchmark data sets of GDA, we show that our approach, which we name **I**ntermediate **DO**main **L**abeler (**IDOL**), can lead to comparable or even better adaptation performance compared to the pre-defined domain sequence, making GDA more applicable and robust to the quality of domain sequences.

1 Introduction

The distributions of real-world data change dynamically due to many factors like time, locations, and environments, etc. Such a fact poses a great challenge to machine-learned models, which implicitly assumes that the test data distribution is covered by the training data distribution. To resolve this generalization problem, unsupervised domain adaption (UDA), which aims to adapt a learned model to the test domain given its unlabeled data [11, 13], has been an active sub-field in machine learning.

Typically, UDA assumes that the "source" domain, in which the model is trained, and the "target" domain, in which the model is deployed, are discrepant but sufficiently related. Concretely, Ben-David et al. [1], Zhao et al. [70] show that the generalization error of UDA is bounded by the discrepancy of the marginal or conditional distributions between domains. In other words, the effectiveness of UDA may degrade along with the increase in discrepancy. Taking one popular algorithm, self-training [33, 43, 44], for example. Self-training adapts the source model by progressively label the unlabeled target data (i.e., pseudo-labels) and use them to fine-tune the model [2, 25, 27, 28, 36, 39, 59, 72, 73]. Self-training works if the pseudo-labels are accurate (as it essentially becomes supervised learning), but is vulnerable if they are not, which occurs when there exists a large domain gap [30].

To address this issue, several recent works investigate *gradual domain adaption (GDA)* [10, 22, 30, 66], in which beyond the source and target domains, the model can access additional unlabeled data from the intermediate domains that shift gradually from the source to the target. By adapting

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

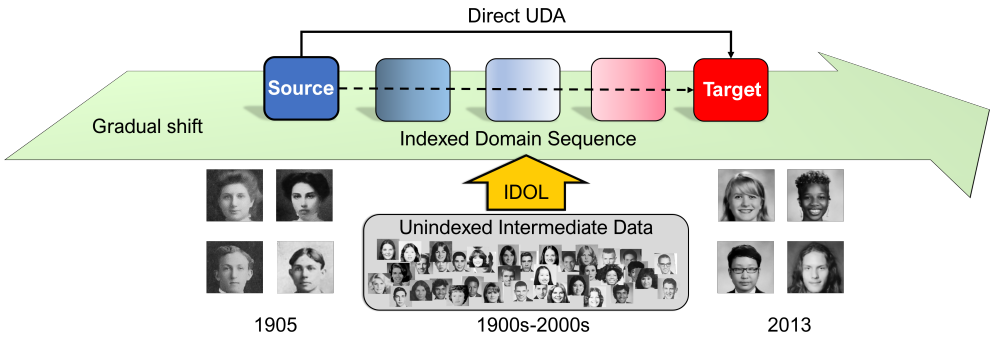


Figure 1: **Gradual domain adaption (GDA) without indexed intermediate domains.** In this setting, one is provided with labeled source, unlabeled target, and additional unlabeled intermediate data that has not been grouped and indexed into a domain sequence. Our approach **intermediate domain labeler (IDOL)** can successfully discover the domain sequence, which can then be leveraged by a GDA algorithm to achieve a higher target accuracy than direct unsupervised domain adaptation (UDA). Images are from the Portraits dataset [12].

the model along the sequence of intermediate domains — i.e., from the ones close to the source to the ones close to the target — the large domain gap between source and target is chipped away by multiple sub-adaptation problems (between consecutive intermediate domains) whose domain gaps are smaller. Namely, every time the model moves a step closer to the target, instead of taking a huge jump that can significantly decrease the performance (see Figure 1 for an illustration). GDA makes sense, as real-world data change gradually more often than abruptly [9, 55, 61]. The recent work by Kumar et al. [30] further demonstrates the strength of GDA both empirically and theoretically.

One potential drawback of GDA, however, is the need for a well-defined sequence of intermediate domains. That is, prior to adaptation, the additional unlabeled data must be grouped into multiple domains, and indexed with intermediate domain labels that reflects the underlying data distribution shift from the source to the target. Such information may not be available directly; existing methods usually leverage side information like time tags [22, 30, 66] to define the sequence, which may be sub-optimal. In some applications, even the side information may not be accessible (e.g., due to privacy concerns), greatly limiting the applicability of GDA.

In this paper, we therefore study GDA in the extreme case — *the additional unlabeled data are neither grouped nor indexed*. Specifically, we investigate how to discover the "domain sequence" from data, such that it can be used to drive GDA. We propose a two-stage coarse-to-fine framework named **I**ntermediate **DO**main **L**abeler (**IDOL**). In the first stage, IDOL labels each intermediate data instance with a *coarse* score that reflect how close it is to the source or target. We study several methods that estimate the distance from a data instance to a domain. We find that a progressively trained domain discriminator — which starts with source vs. target data but gradually adds data close to either of them into training — performs the best, as it better captures the underlying data manifold.

In the second stage, IDOL then builds upon the *coarse* scores to group data into domains by further considering the discriminative (e.g., classification) knowledge the model aims to preserve from the source to the target. Since the additional unlabeled data are fully unlabeled, we take a greedy approach to identify the next intermediate domain (i.e., a group of data) that can best preserve discriminative knowledge in the current intermediate domain. Concretely, we employ self-training [33] along the sequence discovered so far to provide pseudo-labels for the current domain, and propose a novel *cycle-consistency loss* to discover the next domain, such that the model adapted to the next domain can be "adapted back" to the current domain and predict the same pseudo-labels. The output of IDOL is a sequence of intermediate domains that can be used by any GDA algorithms [22, 30, 66].

We validated IDOL on three data sets studied in [30], including Rotated MNIST [31] and Portraits over years [12]. IDOL can successfully discover the domain sequence that leads to comparable GDA performance to using the pre-defined sequence (i.e., by side information). More importantly, IDOL is compatible with pre-defined sequences — by treating them as the coarse sequences — to further improve upon them. We also investigate IDOL in scenarios that the additional unlabeled data may contain outliers, and demonstrate IDOL's effectiveness even in such challenging cases. To our knowledge, our work is the first to tackle GDA without grouped and indexed intermediate domains. The success of IDOL opens up broader application scenarios that GDA can contribute to.

2 Background: Gradual Domain adaptation (GDA)

2.1 Setup of UDA and GDA

We consider an unsupervised domain adaptation (UDA) problem, in which a classifier is provided with labeled data from the source domain $\mathcal{S} = \{(\boldsymbol{x}_i^{\mathcal{S}}, y_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}$ and unlabeled data from the target domain $\mathcal{T} = \{\boldsymbol{x}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$. Both $\mathcal{S}$ and $\mathcal{T}$ are sampled IID from the underlying joint distributions of the source $P_{\mathcal{S}}$ and target $P_{\mathcal{T}}$, respectively. A source model $\boldsymbol{\theta}_{\mathcal{S}}$ is typically produced by training on $\mathcal{S}$ directly. The goal of UDA is to learn a target model $\boldsymbol{\theta}_{\mathcal{T}}$ such that it can perform well on the target, measured by a loss $\mathcal{L}(\boldsymbol{\theta}_{\mathcal{T}}, P_{\mathcal{T}})$. We focus on the standard UDA setup that assumes no label shifts [5].

Gradual domain adaptation (GDA), on top of UDA, assumes the existence of a sequence of domains $P_0, P_1, \dots P_{M-1}, P_M$, where P_0 and P_M are the source domain and target domain, respectively, along with $M-1$ intermediate domains $P_1, \dots, P_{M-1}$. The data distributions of domains are assumed to change gradually from the source to the target along the sequence. One can access unlabeled data $\mathcal{U}_m = \{\boldsymbol{x}_i^{\mathcal{U}_m}\}_{i=1}^{|\mathcal{U}_m|}$ from each of the intermediate domain, forming a sequence of unlabeled data $\mathcal{U}_1, \dots, \mathcal{U}_{M-1}$. We denote the union of them by $\mathcal{U} = \{\boldsymbol{x}_i^{\mathcal{U}}\}_{i=1}^{|\mathcal{U}|}$. For brevity, we assume each $\mathcal{U}_m$ is disjoint from the others but the size $|\mathcal{U}_m|$ is the same $\forall m \in \{1, \cdots, M-1\}$.

2.2 GDA with pre-defined domain sequences using self-training [30]

We summarize the theoretical analysis of GDA using self-training by Kumar et al. [30] as follows.

Self-training for UDA. Self-training [33] takes a pre-trained (source) model $\boldsymbol{\theta}$ and the target unlabeled data $\mathcal{T}$ as inputs. Denote by $\boldsymbol{z} = f(\boldsymbol{x}; \boldsymbol{\theta})$ the predicted logits $\boldsymbol{z}$, self-training first uses $f(\cdot; \boldsymbol{\theta})$ to predict on target data points $\boldsymbol{x} \in \mathcal{T}$, sharpens the predictions by `argmax` into pseudo labels, and updates the current model $\boldsymbol{\theta}$ by minimizing the loss ℓ (e.g., cross entropy) w.r.t the pseudo labels,

$$\boldsymbol{\theta}_{\mathcal{T}} = \texttt{ST}(\boldsymbol{\theta}, \mathcal{T}) = \arg\min_{\boldsymbol{\theta}' \in \boldsymbol{\Theta}} \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \ell\big(f(\boldsymbol{x}_i; \boldsymbol{\theta}'), \texttt{sharpen}(f(\boldsymbol{x}_i; \boldsymbol{\theta}))\big), \tag{1}$$

where $\boldsymbol{\theta}_{\mathcal{T}}$ denotes the resulting target model and $\texttt{ST}(\boldsymbol{\theta}, \mathcal{T})$ denotes the self-training process. A baseline use of self-training for UDA is to set $\boldsymbol{\theta}$ as the source model $\boldsymbol{\theta}_{\mathcal{S}}$. However, when the domain gap between $\mathcal{S}$ and $\mathcal{T}$ is large, $\boldsymbol{\theta}_{\mathcal{S}}$ cannot produce accurate pseudo-labels for effective self-training.

Gradual self-training for GDA. In the GDA setup, self-training is performed along the domain sequence, from the current model $\boldsymbol{\theta}_m$ (updated using $\mathcal{U}_m$ already) to the next one $\boldsymbol{\theta}_{m+1}$ using $\mathcal{U}_{m+1}$

$$\boldsymbol{\theta}_{m+1} = \texttt{ST}(\boldsymbol{\theta}_m, \mathcal{U}_{m+1}), \tag{2}$$

starting from the source model $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{\mathcal{S}}$. By doing so, the model gradually adapts from $\mathcal{S}$ to $\mathcal{T}$ through the sequence $\mathcal{U}_1, \dots, \mathcal{U}_M$ to produce the final target model $\boldsymbol{\theta}_{\mathcal{T}}$.

$$\boldsymbol{\theta}_{\mathcal{T}} = \boldsymbol{\theta}_M = \texttt{ST}\big(\boldsymbol{\theta}_{\mathcal{S}}, (\mathcal{U}_1, \dots, \mathcal{U}_M)\big). \tag{3}$$

As the domain gap between consecutive domains is smaller than between $\mathcal{S}$ and $\mathcal{T}$, the pseudo-labels will be more accurate in each step. As a result, self-training in GDA can lead to a lower target error than in UDA. We note that, the domain sequences in [30] are pre-defined using side information.

Theoretical analysis. Kumar et al. [30] make three assumptions for theoretical analysis:

- Separation: for every domain P_m, the data are separable such that there exists an R-bounded (linear) classifier $\boldsymbol{\theta}_m \in \boldsymbol{\Theta}$ with $\|\boldsymbol{\Theta}\|_2 \le R$ that achieves a low loss of $\mathcal{L}(\boldsymbol{\theta}_m, P_m)$.
- Gradual shift: for the domain sequence $P_0, \dots, P_m$ with no label shifts, the maximum per-class Wasserstein-infinity distance $\rho(P_m, P_{m+1})$ between consecutive domains should be $\le \rho < \frac{1}{R}$.
- Bounded data: finite samples X from each domain P_m are bounded, i.e., $E_{X \sim P_m}[\|X\|_2^2] \le B^2$.

Under these assumptions of GDA, if the source model θ_0 has a low loss $\mathcal{L}(\theta_0, P_0)$ on P_0, then with a probability δ, the resulting target model θ_M through gradual self-training has

$$\mathcal{L}(\boldsymbol{\theta}_M, P_M) \le \beta^{M+1}\Big(\mathcal{L}(\boldsymbol{\theta}_0, P_0) + \frac{4BR + \sqrt{2\log 2M/\delta}}{\sqrt{|\mathcal{U}|}}\Big), \quad \text{where } \beta = \frac{2}{1-\rho R}. \tag{4}$$

That is, the target error is controlled by the intermediate domain shift ρ. If we can sample infinite data (i.e., $|\mathcal{U}| \to \infty$) and the source classifier is perfect (i.e., $\mathcal{L}(\theta_0, P_0) = 0$), with a small distance $\rho(P_m, P_{m+1}) \le \rho$ between consecutive domains, gradual self-training achieves zero target error.

3 GDA without indexed intermediate domains

3.1 Setup and motivation

In this paper, we study the case of GDA where the additional unlabeled data $\mathcal{U}$ are not readily divided into domain sequences $\mathcal{U}_1, \cdots, \mathcal{U}_{M-1}$. In other words, we are provided with a single, gigantic unlabled set that has a wide range of support from the source to the target domain. One naive way to leverage it is to adapt from $\mathcal{S}$ to $\mathcal{U}$[1] and then to $\mathcal{T}$, which however leads to a much larger target error than applying self-training along the properly indexed intermediate domains [30]. We attribute this to the large $\rho(\mathcal{S}, \mathcal{U})$ or $\rho(\mathcal{U}, \mathcal{T})$ according to Equation 4. To take advantage of $\mathcal{U}$, we must separate it into intermediate domains such that $\rho(P_m, P_{m+1})$ is small enough for every $m = 0, \ldots, M-1$.

3.2 Overview of our approach: intermediate domain labeler (IDOL)

Given the source data $\mathcal{S}$, the target data $\mathcal{U}_M = \mathcal{T}$, and the unlabeled data $\mathcal{U} = \{\boldsymbol{x}_i^{\mathcal{U}}\}_{i=1}^{|\mathcal{U}|}$, we propose the intermediate domain labeler (IDOL), whose goal is to sort intermediate data instance in $\mathcal{U}$ and chunk them into $M-1$ domains $\mathcal{U}_1, \cdots, \mathcal{U}_{M-1}$ (with equal sizes) for GDA to succeed.

Taking gradual self-training in Equation 3 as an example, IDOL aims to solve the problem:

$$\begin{aligned} &\min \mathcal{L}(\boldsymbol{\theta}_{\mathcal{T}}, P_{\mathcal{T}}), \\ &\text{s.t. } \boldsymbol{\theta}_{\mathcal{T}} = \texttt{ST}\big(\boldsymbol{\theta}_{\mathcal{S}}, (\mathcal{U}_1, \ldots, \mathcal{U}_M)\big) \quad \text{and} \quad (\mathcal{U}_1, \ldots, \mathcal{U}_{M-1}) = \text{IDOL}(\mathcal{S}, \mathcal{T}, \mathcal{U}; M-1), \end{aligned} \tag{5}$$

where the target model $\boldsymbol{\theta}_{\mathcal{T}}$ is produced by applying gradual self-training to the domain sequence discovered by IDOL. IDOL accepts the number of intermediate domains $M-1$ as a hyper-parameter.

Solving Equation 5 is hard, as evaluating any output sequence needs running through the entire gradual self-training process, not to mention that we have no labeled target data to estimate $\mathcal{L}(\boldsymbol{\theta}_{\mathcal{T}}, P_{\mathcal{T}})$. In this paper, we propose to solve Equation 5 approximately via a coarse-to-fine procedure.

The coarse stage. We aim to give each $\boldsymbol{x}_i^{\mathcal{U}} \in \mathcal{U}$ a score q_i, such that it tells $\boldsymbol{x}$'s position in between the source and target. Specifically, a higher/lower q_i indicates that $\boldsymbol{x}_i^{\mathcal{U}}$ is closer to the source/target domain. With these scores, we can already obtain a coarse domain sequence by *sorting them in a descending order and dividing them into $M-1$ chunks*.

The fine stage. Instead of creating the domain sequence $\mathcal{U}_1, \cdots, \mathcal{U}_{M-1}$ by looking at the individual score of each $\boldsymbol{x}_i^{\mathcal{U}} \in \mathcal{U}$, we further consider how data grouped into the same domain can collectively preserve the discriminative knowledge from the previous domains — after all, the goal of UDA or GDA is to pass the discriminative knowledge from the labeled source domain to the target domain. To this end, we propose a novel cycle-consistency loss to refine the coarse scores progressively.

3.3 The coarse stage: assigning domain scores

In this stage, we assign each $\boldsymbol{x}_i^{\mathcal{U}} \in \mathcal{U}$ a score q_i, such that higher/lower q_i indicates that $\boldsymbol{x}_i^{\mathcal{U}}$ is closer to the source/target domain. We discuss some design choices. For brevity, we omit the superscript $^{\mathcal{U}}$.

Confidence scores by the classifier $f(x; \boldsymbol{\theta})$. We first investigate scoring each intermediate data instance by the confidence score of the source model $\boldsymbol{\theta}_{\mathcal{S}}$, i.e., $q_i = \max f(\boldsymbol{x}_i; \boldsymbol{\theta}_{\mathcal{S}})$. This is inspired by outlier detection [37] and the common practice of self-training, which selects only data with a sufficiently large confidence scores to fine-tune upon [30, 33]. Here we employ an advanced version, which is to use the latest $\boldsymbol{\theta}$ (i.e., the one adapted to the current intermediate domain) to select the next intermediate domain. That is, every time we select the highest confidence instances from the *remaining* ones in $\mathcal{U}$ to adapt $\boldsymbol{\theta}$. Overall, we can give data selected in the m-th round ($m \in \{1, \cdots, M-1\}$) a score $\frac{M-1-m}{M-2} \in [0, 1]$. One drawback of this method is the blindness to the target domain $\mathcal{T}$. Specifically, we find that using confidence scores tends to selects easily classified examples, which do necessarily follows the gradual shift from the source to the target.

Manifold distance with the source features. To model the flow from the source to the target, we argue that it is crucial to consider each side as references. We thus extract features of $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$ using the source model $\boldsymbol{\theta}_{\mathcal{S}}$ and apply a manifold algorithm (e.g., UMAP [45]) to discover the

[1] In [30], the authors investigated sampling $\mathcal{U}_1, \cdots, \mathcal{U}_{M-1}$ from $\mathcal{U}$ uniformly at random with replacement.

underlying manifold. Denote by $\gamma(\boldsymbol{x})$ the dimension-reduced features of $\boldsymbol{x}$ after UMAP , we compute the ratio of distance to the nearest point in the $\mathcal{S}$ and $\mathcal{T}$ as the score $q_i = \frac{\min_{\boldsymbol{x}^{\mathcal{T}} \in \mathcal{T}} \|\gamma(\boldsymbol{x}_i) - \gamma(\boldsymbol{x}^{\mathcal{T}})\|_2}{\min_{\boldsymbol{x}^{\mathcal{S}} \in \mathcal{S}} \|\gamma(\boldsymbol{x}_i) - \gamma(\boldsymbol{x}^{\mathcal{S}})\|_2}$.

Domain discriminator. We investigate another idea to emphasize the roles of the source and target, which is to train a domain discriminator [11]. Concretely, we train a binary classifier $g(\cdot; \boldsymbol{\phi})$ using deep neural networks, which is learned to separate the source $\mathcal{S}$ (class: 1) and the target data $\mathcal{T}$ (class: 0). We use a binary-cross entropy loss to optimize the learnable parameter $\boldsymbol{\phi}$

$$\mathcal{L}(\boldsymbol{\phi}) = -\frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x}^{\mathcal{S}} \in \mathcal{S}} \log(\sigma(g(\boldsymbol{x}^{\mathcal{S}}; \boldsymbol{\phi}))) - \frac{1}{|\mathcal{T}|} \sum_{\boldsymbol{x}^{\mathcal{T}} \in \mathcal{T}} \log(1 - \sigma(g(\boldsymbol{x}^{\mathcal{T}}; \boldsymbol{\phi})),$$

where σ is the sigmoid function. We can then assign a score to $\boldsymbol{x}_i \in \mathcal{U}$ by $q_i = g(\boldsymbol{x}_i; \boldsymbol{\phi})$.

Progressive training for the domain discriminator The domain discriminator $g(\cdot; \boldsymbol{\phi})$, compared to the manifold distance, better contrasts the source and target domains but does not leverage any of the data $\boldsymbol{x} \in \mathcal{U}$. That is, it might give a faithful score to an example $\boldsymbol{x}$ that is very close to $\mathcal{S}$ or $\mathcal{T}$, but for other examples that are far away and hence out-of-domain from both ends, the score by $g(\cdot; \boldsymbol{\phi})$ becomes less reliable, not to mention that neural networks are usually poorly calibrated for these examples [19, 37]. We therefore propose to progressively augment the source and target data with $\boldsymbol{x} \in \mathcal{U}$ that has either a fairly high or low $g(\boldsymbol{x}; \boldsymbol{\phi})$, and fine-tune the domain discriminator $g(\boldsymbol{x}; \boldsymbol{\phi})$. We perform this process for K rounds, and every time include $\frac{|\mathcal{U}|}{2K}$ new examples into the source and target. By doing so, the domain discriminator $g(\boldsymbol{x}; \boldsymbol{\phi})$ is updated with data from $\mathcal{U}$ and thus can provide more accurate information to distinguish the remaining examples into the source or target side. Overall, we can give data selected in the k-th round ($k \in \{1, \cdots, K\}$) a score $\frac{2K-k}{2K}$ if it is augmented to the source side, or $\frac{k}{2K}$ if it is augmented to the target side.

3.4 The fine stage: cycle-consistency for refinement

The domain scores developed in subsection 3.3 can already give each individual example $\boldsymbol{x} \in \mathcal{U}$ a rough position of which intermediate domain it belongs to. However, for GDA to succeed, we must consider intermediate data in a collective manner. That is, each intermediate domain should preserve sufficient discriminative (e.g., classification) knowledge from the previous one, such that after M rounds of adaptation through GDA, the resulting $\boldsymbol{\theta}_{\mathcal{T}}$ in Equation 5 will be able to perform well on $\mathcal{T}$. This is reminiscent of some recent claims in UDA [41, 70]: matching only the marginal distributions (i.e., blind to the class labels) between domains may lead to sub-optimal results.

But how can we measure the amount of discriminative knowledge preserved by an intermediate domain or the target domain if we do not have its labeled data? To seek for the answer, we revisit Equation 4 and consider the case $M = 1$. If the two domains P_0 and P_1 are sufficiently close and the initial model $\boldsymbol{\theta}_0$ is well-trained, the resulting model $\boldsymbol{\theta}_1$ after self-training will perform well. Reversely, we can treat such a well-performing $\boldsymbol{\theta}_1$ as the well-trained initial model, and apply self-training again — this time from P_1 back to P_0. The resulting model $\boldsymbol{\theta}'_0$, according to the same rationale, should perform well on P_0. In other words, $\boldsymbol{\theta}'_0$ and $\boldsymbol{\theta}_0$ should predict similarly on data sampled from P_0. The **similarity between $\boldsymbol{\theta}'_0$ and $\boldsymbol{\theta}_0$ in terms of their predictions** therefore can be used as a proxy to measure **how much discriminative knowledge $\boldsymbol{\theta}_1$, after adapted to P_1, preserves**. It is worth noting that measuring the similarity between $\boldsymbol{\theta}'_0$ and $\boldsymbol{\theta}_0$ require no labeled data from P_0 or P_1.

Now let us consider a more general case that M is not limited to 1. Let us set $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{\mathcal{S}}$, where $\boldsymbol{\theta}_{\mathcal{S}}$ has been trained with the labeled source data $\mathcal{S}$ till convergence. Based on the concept mentioned above, we propose a novel learning framework to discover the domain sequences,

$$\begin{aligned} \arg\min_{(\mathcal{U}_1, \ldots, \mathcal{U}_{M-1})} & \mathbb{E}_{\boldsymbol{x} \sim P_0} \left[\ell\big(f(\boldsymbol{x}; \boldsymbol{\theta}'_0), \texttt{sharpen}(f(\boldsymbol{x}; \boldsymbol{\theta}_0))\big) \right], \\ \text{s.t., } & \boldsymbol{\theta}'_0 = \texttt{ST}(\boldsymbol{\theta}_M, (\mathcal{U}_{M-1}, \ldots, \mathcal{U}_0)), \\ & \boldsymbol{\theta}_M = \texttt{ST}(\boldsymbol{\theta}_0, (\mathcal{U}_1, \ldots, \mathcal{U}_M)), \end{aligned} \tag{6}$$

where $\mathcal{U}_0$ is the source data $\mathcal{S}$ with labels removed, and $\mathcal{U}_M = \mathcal{T}$ is the target data. $\boldsymbol{\theta}'_0$ is a gradually self-trained model from $\boldsymbol{\theta}_0$ that undergoes a cycle $\mathcal{U}_1 \rightarrow, \cdots, \rightarrow \mathcal{U}_M \rightarrow, \cdots, \rightarrow \mathcal{U}_0$. In other words, Equation 6 aims to find the intermediate domain sequences such that performing gradual self-training along it in a forward and then backward fashion leads to cycle-consistency [71].

Optimization. Solving Equation 6 is still not trivial due to its combinatorial nature. We therefore propose to solve it **greedily, starting with discovering $\mathcal{U}_1$, and then $\mathcal{U}_2$, and then so on.** Concretely,

we decompose Equation 6 into a series of sub-problems, as illustrated in Figure 2, each is defined as

$$\begin{aligned} &\arg\min_{\mathcal{U}_{m+1}\subset\mathcal{U}_{\setminus m}} \sum_{\boldsymbol{x}\in\mathcal{U}_m} \ell\big(f(\boldsymbol{x};\boldsymbol{\theta}'_m), \texttt{sharpen}(f(\boldsymbol{x};\boldsymbol{\theta}_m))\big), \\ &\text{s.t., } \boldsymbol{\theta}'_m = \texttt{ST}(\boldsymbol{\theta}_{m+1}, \mathcal{U}_m), \\ &\qquad\quad \boldsymbol{\theta}_{m+1} = \texttt{ST}(\boldsymbol{\theta}_m, \mathcal{U}_{m+1}), \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}_m$ is the model already adapted to the current domain $\mathcal{U}_m$, and $\mathcal{U}_{\setminus m} = \mathcal{U} \setminus \cup_{j=1}^{m}\mathcal{U}_j$ is the remaining data that have not been selected by the m intermediate domains so far. That is, each sub-problem discovers only the next intermediate domain $\mathcal{U}_{m+1}$ via cycle-consistency.

Let $N = |\mathcal{U}_{\setminus m}|$ and $\mathcal{U}_{\setminus m} = \{\boldsymbol{x}_i\}_{i=1}^N$, selecting $\mathcal{U}_{m+1} \subset \mathcal{U}_{\setminus m}$ is equivalent to selecting a binary indicator vector $\boldsymbol{q} \in \{0,1\}^N$, in which $q_i = 1$[2] means that $\boldsymbol{x}_i \in \mathcal{U}_{\setminus m}$ is included into $\mathcal{U}_{m+1}$. This representation turns the self-training step $\boldsymbol{\theta}_{m+1} = \texttt{ST}(\boldsymbol{\theta}_m, \mathcal{U}_{m+1})$ in Equation 7 into

$$\texttt{ST}(\boldsymbol{\theta}_m, \boldsymbol{q}) = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{1}{N}\sum_{i=1}^{N} q_i \times \ell\big(f(\boldsymbol{x}_i;\boldsymbol{\theta}), \texttt{sharpen}(f(\boldsymbol{x}_i;\boldsymbol{\theta}_m))\big), \tag{8}$$

We choose to solve Equation 7 approximately by relaxing the binary vector $\boldsymbol{q} \in \{0,1\}^N$ into a real vector $\boldsymbol{q} \in \mathbb{R}^N$, which fits perfectly into the meta-reweighting framework [26, 50]. Concretely, meta-reweighting treats $\boldsymbol{q} \in \mathbb{R}^N$ as learnable differentiable parameters associated to training examples (in our case, the data in $\mathcal{U}_{\setminus m}$). Meta-reweighting for Equation 7 can be implemented via the following six steps for multiple iterations.

1. Detach: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_m$,
2. Forward: $\boldsymbol{\theta}(\boldsymbol{q}) \leftarrow \boldsymbol{\theta} - \frac{\eta_{\boldsymbol{\theta}}}{|\mathcal{U}_{\setminus m}|} \times \frac{\partial \sum_{i\in\mathcal{U}_{\setminus m}} q_i \times \ell(f(\boldsymbol{x}_i;\boldsymbol{\theta}), \texttt{sharpen}(f(\boldsymbol{x}_i;\boldsymbol{\theta}_m)))}{\partial\boldsymbol{\theta}}$,
3. Detach: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}(\boldsymbol{q})$,
4. Backward: $\boldsymbol{\theta}(\boldsymbol{q}) \leftarrow \boldsymbol{\theta}(\boldsymbol{q}) - \frac{\eta_{\boldsymbol{\theta}}}{|\mathcal{U}_m|} \times \frac{\partial \sum_{j\in\mathcal{U}_m} \ell(f(\boldsymbol{x}_j;\boldsymbol{\theta}(\boldsymbol{q})), \texttt{sharpen}(f(\boldsymbol{x}_j;\boldsymbol{\theta}')))}{\partial\boldsymbol{\theta}(\boldsymbol{q})}$,
5. Update: $\boldsymbol{q} \leftarrow \boldsymbol{q} - \frac{\eta_{\boldsymbol{q}}}{|\mathcal{U}_m|} \times \frac{\partial \sum_{j\in\mathcal{U}_m} \ell(f(\boldsymbol{x}_j;\boldsymbol{\theta}(\boldsymbol{q})), \texttt{sharpen}(f(\boldsymbol{x}_j;\boldsymbol{\theta}_m)))}{\partial\boldsymbol{q}}$,
6. Update: $\boldsymbol{q}_i \leftarrow \max\{0, \boldsymbol{q}_i\}$.

The aforementioned updating rule gives $\boldsymbol{x}_i$ a higher value of q_i if it helps preserve the discriminative knowledge. After obtaining the updated $\boldsymbol{q} \in \mathbb{R}^N$, we then sort it in the descending order and select the top $\frac{|\mathcal{U}|}{M-1}$ examples to be $\mathcal{U}_{m+1}$ (i.e., every intermediate domain has a equal size).

Initialization. We initialize q_i by the coarse scores defined in subsection 3.3. This is for one important reason: Equation 7 searches for the next intermediate domain $\mathcal{U}_{m+1}$ only based on the current intermediate domain $\mathcal{U}_m$. Thus, it is possible that $\mathcal{U}_{m+1}$ will include some data points which help preserve the knowledge of $\mathcal{U}_m$ but are distributionally deviated from the gradual transition between $\mathcal{U}_m$ and the target domain $\mathcal{T}$. The scores defined in subsection 3.3 thus serve as the guidance; we can also view Equation 7 as a refinement step of the coarse scores. As will be seen in section 4, the quality of the initialization and the refinement both play important roles in IDOL's success.

4 Experiment

4.1 Setup

We mainly study on two benchmark datasets used in [30]. Rotated MNIST [31] is a popular synthetic datasets to study distributional shift. It gradually rotates the original 50,000 images in MNIST [32] dataset from [0, 60] degrees uniformly,

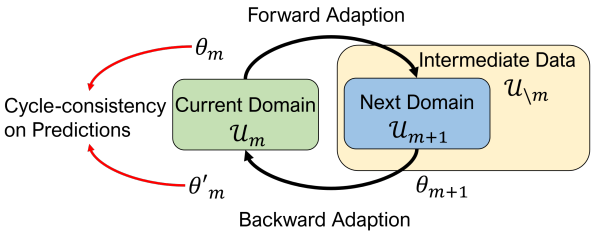
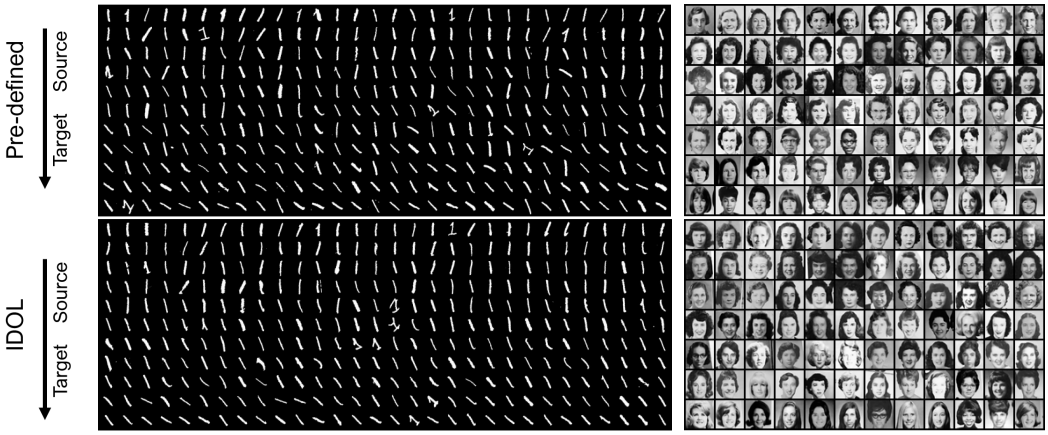


Figure 2: Cycle-consistency (cf. Equation 7).

[2]Here we use the same notation as the coarse scores defined in subsection 3.3, since later we will initialize these values indeed by the coarse scores.



(a) Rotated MNIST (class "1") (b) Portraits (class "female")

Figure 3: Random samples (the most representative class) from pre-defined indexes or IDOL sequences.

Coarse scores	Indexed?	Adaptation	Refined?	Rotated MNIST	Portraits
None	✗	Source only	-	31.9±1.7	75.3±1.6
		UDA ($\mathcal{T}$)	-	33.0±2.2	76.9±2.1
		UDA ($\mathcal{T}+\mathcal{U}$)	-	38.0±1.6	78.9±3.0
Pre-defined [30]	✓	GDA	✗	87.9±1.2	83.8±0.8
			✓	93.3±2.3	85.8±0.4
Random	✗	GDA	✗	39.5±2.0	81.1±1.8
			✓	57.5±2.7	82.5±2.2
Classifier confidence	✗	GDA	✗	45.5±3.5	79.3±1.7
Manifold distance			✗	72.4±3.1	81.9±0.8
Domain discriminator			✗	82.1±2.7	82.3±0.9
Progressive domain discriminator			✗	85.7±2.7	83.4±0.8
			✓	87.5±2.0	85.5±1.0

Table 1: Gradual self-training on Rotated MNIST and Portraits. Bottom section: IDOL with domain scores.

where $[0, 5)/[5, 55)/[55, 60]$ degrees are for the source/intermediate/target domains, respectively. However, since original images have no ground truths of the rotation, the rotation annotations are not expected to be perfect. Portraits dataset [12] is a real-world gender classification dataset of a collection of American high school seniors from the year 1905 to 2013. Naturally, the structural shifts are arbitrary along years (e.g., lip curvature [12], fashion styles). The examples are first sorted by the years and split. For both datasets, each domain contains 2000 images and $1,000$ images are reserved for both the source and target domain as validation sets.

We follow the setup in [30]: each model is a convolutional network is trained for 20 epochs for each domain consequently including training on the source, using Adam optimizer with learning rate 0.001, batch size 32, and weight decay 0.02. We also use this optimizer as the default if not specified. Hyper-parameter to IDOL including $K = 2M$ rounds for progressive training, 30 epochs of refinement each step, with mini-batch 128. More details are in the supplementary materials.

To study how the domain sequences affect gradual self-training, we consider baselines including *Source only*, *UDA* that directly self-trains on the target ($\mathcal{T}$) data and all the intermediate data ($\mathcal{U}$). We compare different domain sequences including *Pre-defined* indexes, e.g., *rotations* and *years*. Then, we consider the challenging setup that all intermediate data are unindexed. *Random* sequences are by dividing the unlabeled data into $M - 1$ domains randomly, with self-training on the target in the end. For our IDOL, different domain scores introduced in subsection 3.3 are adopted for the coarse domain sequences, and optionally apply cycle-consistency refinement to make them fine-grained.

4.2 Main study: comparison with pre-defined domain sequences

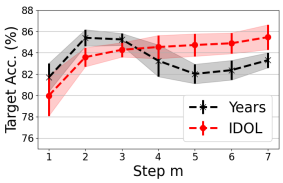
The main results on Rotated MNIST and Portraits are provided in Table 1. We first verify that without any domain sequences, UDA on target and intermediate data, and GDA on random sequences do not perform well, though they slightly improve on the source model. On the contrary, training with pre-defined indexes that arguably guides the transfer gradually, it significantly performs better.

IDOL is competitive to pre-defined sequences. IDOL can produce meaningful domain sequences (see Figure 3), *given only the unlabeled data without any order*. A domain discriminator with progressive training performs the best and comparable to pre-defined sequences. The classifier is not enough to capture domain shifts.

Refinement helps and coarse initialization matters. We observe the fine indexes produced by cycle-consistency refinement generally improve on coarse sequences, validating its effectiveness. Refining pre-defined sequences can further improve them. Notably, it is important to have high-quality coarse sequences as the initialization.

Analysis. To understand why refinement helps, we take a closer look at the Portraits dataset that is naturally sorted by *years*. Is *years* the best factor to construct the domain sequence? We compare it with that refined by IDOL. In Figure 4, we monitor the target accuracy of each step of adaptation and observe the performance fluctuates. Interestingly, learning with IDOL stably and gradually improves the target accuracy and ends up higher accuracy, validating that refinement with cycle-consistency indeed produces better sequences.

(a) Target Acc. of gradual ST.



(b) Years vs. IDOL sequence.

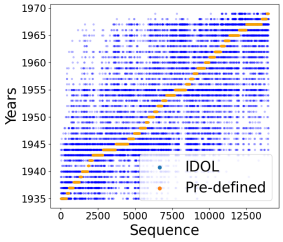


Figure 4: Portraits with year indexes or IDOL domain sequence.

Intermediate domains constructed with the domain scores are not necessarily match the target distributions. We monitor the number of examples per class, by computing $\frac{1}{M-1}\sum_{m=1}^{M-1}\frac{\max_y\{|\mathcal{U}_{m,y}|\}}{\min_y\{|\mathcal{U}_{m,y}|\}}$, which ideally should be 1.0, i.e., class-balanced. We observe fine-grained indexes are indeed more class-balanced (coarse vs fine-grained): 1.33 vs 1.23 on Rotated MNIST and 1.27 vs 1.25 on Portraits.

4.3 Case study: learning with partial annotations

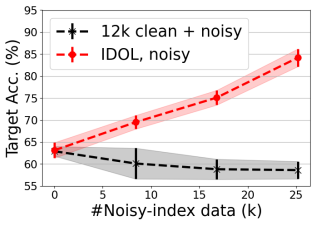
If partial information is given, how would IDOL help? We examine the question from three experiments. First, given *only unsorted domains* that the data are grouped into $M-1$ domains according to pre-defined indexes but the domains are unordered. We sort the domains with the mean domain scores $\frac{1}{|\mathcal{U}_m|}\sum_{\boldsymbol{x}_i\in\mathcal{U}_m} q_i$, and found it perfectly recovers the pre-defined order. Second, *more coarse domains*, we enlarge the $|\mathcal{U}_m|$ to $4,000$ images and found Rotated MNIST (w/ pre-defined indexes) degrades by 11% accuracy. IDOL can maintain it since IDOL outputs a sorted sequence of data points but not just domains. Third, *unknown sequence* which the starting/end intermediate domain might not match to the source/target. We consider the rotated MNIST experiment but now the intermediate domains are from $[-30, 90]$ degrees sequentially. GDA on such unknown sequence degrades to 77.0 ± 2.0 accuracy. Refinement still improves it to 81.3 ± 1.4.

4.4 Case study: learning with low-quality indexes

In the real world, the domain indexes for the intermediate data might not be perfect. We consider several realistic cases to show that our method can make GDA more robust to the quality of the domain sequences. First, *less intermediate data*: Equation 4 implies that the transfer error increases if the indexed intermediate data is too sparse. We repeat the experiments in subsection 4.2 but with only 30% indexed data. Second, we consider *noisy indexes*: with the 30% clean indexes, we further add more data with noisy random indexes.

Figure 5 summarizes the results. We first notice that with only 30% intermediate data, even the indexes are clean, the accuracy decreases drastically by 14% for MNIST and 5% for Portraits. For MNIST, adding more noisy-index data actually decreases more. Portraits gains some improvements even the data are noisy-index, likely because the domain shift is comparably subtle to the benefits from better generalization with more data. One advantage of IDOL is that we can annotate domain indexes for any unindexed data. On both datasets, IDOL stably improves with more data available, significantly outperforming without IDOL.

(a) Rotated MNIST.



(b) Portraits.

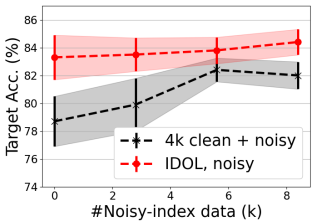


Figure 5: Different numbers of noise-index intermediate data.

4.5 Case study: CIFAR10-STL UDA with open-domain unlabeled sets

We examine IDOL on a competitive CIFAR10 to STL UDA task which is known to be challenging due to the small size [4]. We use the unlabeled set in STL, which is sub-sampled from ImageNet, where the data are from unknown domain and even out-class. Thus, we use $M = 2$ and filter out 80% intermediate data with classifier confidence. Here we do not aim to compete with the state of the arts but use it to study the effectiveness of IDOL. We first trained a ResNet-20 source model. In Table 2, we observe direct UDA on STL or unlabeled data are inferior to the source model even the learning rate is tuned, confirming the challenges of this setup. GDA with IDOL still performs the best.

Table 2: CIFAR10-STL.

Method	Target Acc.
Source only	76.6±0.4
UDA ($\mathcal{T}$) (lr $= 10^{-4}$)	69.4±0.4
UDA ($\mathcal{T}$) (lr $= 10^{-5}$)	75.1±0.3
UDA ($\mathcal{T} + \mathcal{U}$)	61.1±0.8
GDA w/ conf.	77.1±0.5
GDA w/ IDOL	78.1±0.4

5 Related Work

Domain adaption. UDA has been studied extensively [1, 48, 58], and many different approaches have been proposed such as minimizing domain divergence [11, 18, 40, 56, 57, 60] or cross-domain [34, 52, 53, 56] and domain-invariant features [16, 23, 49, 62–64, 68]. Self-training recently emerges as a simple yet effective approach [2, 25, 27, 28, 36, 59, 72]. It approaches UDA towards supervised learning in the target domain with accurate pseudo-labels from the source model [33, 43, 44]. Theoretical studies for applying self-training in UDA are provided in [3, 65].

Gradual domain adaption. Many UDA works show the benefits of gradually bridging the domain gap. With features as inputs, several work [7, 13, 17] construct Grassmannian manifolds between the source and the target. For deep models, progressive DA [24] proposes to create the synthetic intermediate domain with an image-to-image translation network. Other works [6, 15] instead generate intermediate data with jointly-trained generators. Na et al. [46] augments the intermediate data with Mixup [69]. Unlike all the above work are focusing on building synthetic intermediate domains given only the source and target data, gradual domain adaption proposed in [10, 22, 30, 66] studies how to leverage extra real intermediate data with pre-defined domain indexes.

Learning with cycle consistency. The concept of cycle consistency has been successfully applied in many machine learning tasks. High-levelly, cycle supervision enforces the translation to target should be able to be translated back to match the source. For instance, back-translation [20, 54] is a popular technique to perform unsupervised neural translation in natural language processing. Many computer vision tasks such as image generation [51], image segmentation[35], style transfer [71], etc., can also be improved by matching the structural outputs with cyclic signals. We propose a novel application that uses cycle consistency to discover intermediate domains for gradual adaption.

Discovering feature subspaces. We further discuss the connections of our work to existing efforts about subspace learning. One powerful technique is subspace clustering [8, 29, 47] that partition data into many subspaces (e.g. classes) unsupervisedly. However, it may not be applied for GDA directly since it groups with discriminative features which might not capture domain shifts. Specifically for UDA, some work focus on discovering sub-spaces in the source/target domains [14, 21, 38, 42, 67]. The assumption is that the datasets might not be collected from one environment but generated from many different distributions, re-formulating it as a multi-domain adaption problem. We note that, the sub-domains discovered in these methods are fundamentally different from the domain sequence we discussed. The sub-domains here are assumed to exist a dedicated adaption mapping to the target domain. The domain sequences in GDA are for constructing a path from the source to the target via intermediate data. How to link these learning techniques to GDA will be interesting future work.

6 Conclusion

Gradual domain adaptation leverages intermediate data to bridge the domain gap between the source and target domains. However, it relies on a strong premise — prior knowledge of domain indexes to sort the intermediate data into a domain sequence. We propose IDOL algorithm that can produce comparable or even better domain sequences without pre-defined indexes. With a simple progressively-trained domain discriminator, it matches the performance of using pre-defined indexes, and further improves with the proposed cycle-consistency refinement. Essentially, IDOL is a useful tool to augment any GDA algorithms with high-quality domain sequences given unindexed data.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 1, 9

[2] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *NeurIPS*, 2011. 1, 9

[3] Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. In *NeurIPS*, 2020. 9

[4] Safa Cicek and Stefano Soatto. Unsupervised domain adaptation via regularized conditional alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1416–1425, 2019. 9

[5] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain adaptation with conditional distribution matching and generalized label shift. In *NeurIPS*, 2020. 3

[6] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Chi Su, Qingming Huang, and Qi Tian. Gradually vanishing bridge for adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12455–12464, 2020. 9

[7] Zhen Cui, Wen Li, Dong Xu, Shiguang Shan, Xilin Chen, and Xuelong Li. Flowing on riemannian manifold: Domain adaptation by shifting covariance. *IEEE transactions on cybernetics*, 44(12):2264–2273, 2014. 9

[8] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013. 9

[9] Ali Farshchian, Juan A Gallego, Joseph P Cohen, Yoshua Bengio, Lee E Miller, and Sara A Solla. Adversarial domain adaptation for stable brain-machine interfaces. *arXiv preprint arXiv:1810.00045*, 2018. 2

[10] Michael Gadermayr, Dennis Eschweiler, Barbara Mara Klinkhammer, Peter Boor, and Dorit Merhof. Gradual domain adaptation for segmenting whole slide images showing pathological variability. In *International Conference on Image and Signal Processing*, pages 461–469. Springer, 2018. 1, 9

[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. 1, 5, 9

[12] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A Efros. A century of portraits: A visual historical record of american high school yearbooks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–7, 2015. 2, 7

[13] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2066–2073. IEEE, 2012. 1, 9

[14] Boqing Gong, Kristen Grauman, and Fei Sha. Reshaping visual datasets for domain adaptation. *Advances in Neural Information Processing Systems*, 2013. 9

[15] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2477–2486, 2019. 9

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 9

[17] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011. 9

[18] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009. 9

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017. 5

[20] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. *Advances in neural information processing systems*, 29:820–828, 2016. 9

[21] Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *European Conference on Computer Vision*, pages 702–715. Springer, 2012. 9

[22] Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–874, 2014. 1, 2, 9

[23] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 9

[24] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. 9

[25] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *CVPR*, 2018. 1, 9

[26] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7610–7619, 2020. 6

[27] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *ICCV*, 2019. 1, 9

[28] Seunghyeon Kim, Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection. In *ICCV*, 2019. 1, 9

[29] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Acm transactions on knowledge discovery from data (tkdd)*, 3(1):1–58, 2009. 9

[30] Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *International Conference on Machine Learning*, pages 5468–5479. PMLR, 2020. 1, 2, 3, 4, 6, 7, 9

[31] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007. 2, 6

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

[33] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 1, 2, 3, 4, 9

[34] Seungmin Lee, Dongwan Kim, Namil Kim, and Seong-Gyun Jeong. Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In *ICCV*, 2019. 9

[35] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6936–6945, 2019. 9

[36] Jian Liang, Ran He, Zhenan Sun, and Tieniu Tan. Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation. In *CVPR*, 2019. 1, 9

[37] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017. 4, 5

[38] Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X Yu, and Boqing Gong. Open compound domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2020. 9

[39] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013. 1

[40] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015. 9

[41] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017. 5

[42] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3771–3780, 2018. 9

[43] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *ACL*, 2006. 1, 9

[44] David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *ACL*, 2006. 1, 9

[45] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 4

[46] Jaemin Na, Heechul Jung, HyungJin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. *arXiv preprint arXiv:2011.09230*, 2020. 9

[47] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001. 9

[48] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010. 9

[49] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 9

[50] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018. 6

[51] Paolo Russo, Fabio M Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8099–8108, 2018. 9

[52] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. *arXiv preprint arXiv:1711.01575*, 2017. 9

[53] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018. 9

[54] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015. 9

[55] Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017. 2

[56] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *ICLR*, 2018. 9

[57] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *JMLR*, 8(May):985–1005, 2007. 9

[58] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016. 9

[59] Qingyi Tao, Hao Yang, and Jianfei Cai. Zero-annotation object detection with web knowledge transfer. In *ECCV*, 2018. 1, 9

[60] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 9

[61] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A Ryan, Margie L Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012. 2

[62] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5495–5504, 2018. 9

[63] Ximei Wang, Liang Li, Weirui Ye, Mingsheng Long, and Jianmin Wang. Transferable attention for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5345–5352, 2019.

[64] Yan Wang, Chen Xiangyu, You Yurong, Erran Li Li, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Chao Wei-Lun. Train in germany, test in the usa: Making 3d object detectors generalize. In *CVPR*, 2020. 9

[65] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *ICLR*, 2021. 9

[66] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 4489–4495. IEEE, 2018. 1, 2, 9

[67] Caiming Xiong, Scott McCloskey, Shao-Hang Hsieh, and Jason Corso. Latent domains modeling for visual domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014. 9

[68] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *AAAI*, 2020. 9

[69] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 9

[70] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532. PMLR, 2019. 1, 5

[71] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 5, 9

[72] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 1, 9

[73] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991, 2019. 1

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please be referred to the broader impact section in the supplementary material.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We use the datasets released for scientific research.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]