Journal of Machine Learning Research 25 (2024) 1-40 Submitted 9/23; Revised 9/24; Published 11/24

Gradual Domain Adaptation: Theory and Algorithms

Yifei He[*] YIFEIHE3@ILLINOIS.EDU
University of Illinois Urbana-Champaign

Haoxiang Wang[*] HWANG264@ILLINOIS.EDU
University of Illinois Urbana-Champaign

Bo Li BOL@UCHICAGO.EDU
University of Chicago

Han Zhao HANZHAO@ILLINOIS.EDU
University of Illinois Urbana-Champaign

Editor: Quentin Berthet

Abstract

Unsupervised domain adaptation (UDA) adapts a model from a labeled source domain to an unlabeled target domain in a one-off way. Though widely applied, UDA faces a great challenge whenever the distribution shift between the source and the target is large. Gradual domain adaptation (GDA) mitigates this limitation by using intermediate domains to gradually adapt from the source to the target domain. In this work, we first theoretically analyze gradual self-training, a popular GDA algorithm, and provide a significantly improved generalization bound compared with Kumar et al. (2020). Our theoretical analysis leads to an interesting insight: to minimize the generalization error on the target domain, the sequence of intermediate domains should be placed uniformly along the Wasserstein geodesic between the source and target domains. The insight is particularly useful under the situation where intermediate domains are missing or scarce, which is often the case in real-world applications. Based on the insight, we propose **G**enerative Gradual D**O**main **A**daptation with Optimal **T**ransport (GOAT), an algorithmic framework that can generate intermediate domains in a data-dependent way. More concretely, we first generate intermediate domains along the Wasserstein geodesic between two given consecutive domains in a feature space, then apply gradual self-training to adapt the source-trained classifier to the target along the sequence of intermediate domains. Empirically, we demonstrate that our GOAT framework can improve the performance of standard GDA when the given intermediate domains are scarce, significantly broadening the real-world application scenarios of GDA. Our code is available at `https://github.com/uiuctml/GOAT`.

Keywords: Gradual Domain Adaptation, Distribution Shift, Optimal Transport, Out-of-distribution Generalization

1. Introduction

Modern machine learning models suffer from data distribution shifts across various settings and datasets (Gulrajani and Lopez-Paz, 2021; Sagawa et al., 2021; Koh et al., 2021; Hendrycks et al., 2021; Wiles et al., 2022), i.e., trained models may face a significant performance drop when the test data come from a distribution largely shifted from the training data distribution.

[*] Equal contribution.

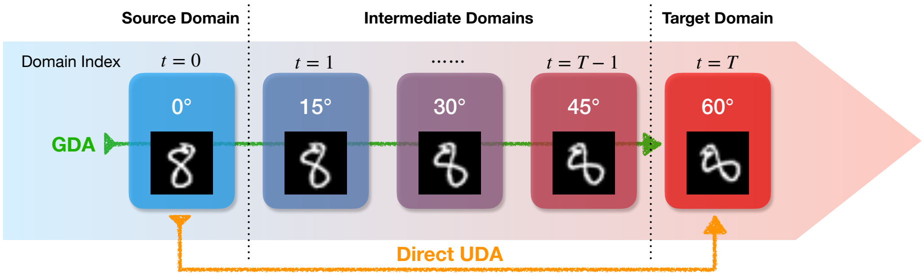


Figure 1: A schematic diagram comparing Unsupervised Domain Adaptation (UDA) vs. Gradual Domain Adaptation (GDA), using the example of Rotated MNIST. In GDA, given labeled data from a source domain, models are adapted to the target domain, with the help of unlabeled data from intermediate domains gradually shifting from the source to target.

Unsupervised domain adaptation (UDA) is a promising approach to address the distribution shift problem by adapting models from the training distribution (source domain) with labeled data to the test distribution (target domain) with unlabeled data (Ganin et al., 2016; Long et al., 2015; Zhao et al., 2018; Tzeng et al., 2017). Typical UDA approaches include adversarial training (Ajakan et al., 2014; Ganin et al., 2016; Zhao et al., 2018), distribution matching (Zhang et al., 2019; Tachet des Combes et al., 2020; Li et al., 2021, 2022), optimal transport (Courty et al., 2016, 2017), and self-training (aka pseudo-labeling) (Liang et al., 2019, 2020; Zou et al., 2018, 2019; Wang et al., 2022a). However, as the distribution shifts become large, these UDA algorithms suffer from significant performance degradation (Kumar et al., 2020; Sagawa et al., 2021; Abnar et al., 2021; Wang et al., 2022a). This empirical observation is consistent with theoretical analyses (Ben-David et al., 2010; Zhao et al., 2019a; Tachet des Combes et al., 2020), which indicate that the expected test accuracy of a trained model in the target domain degrades as the distribution shift becomes larger.

When facing a large data distribution shift, our key strategy is the classic *divide-and-conquer*: breaking the large shift into pieces of smaller shifts, resolving each piece with classic UDA approaches, and then combining all the intermediate solutions to recover a solution to the original data-shift problem (Figure 2). Concretely, the data distribution shift between the source and target can be divided into pieces with intermediate domains bridging the two (i.e., the source and target). This methodology of leveraging intermediate data to tackle large distribution shift is known as gradual domain adaptation (GDA) (Kumar et al., 2020; Abnar et al., 2021; Chen and Chao, 2021; Gadermayr et al., 2018; Wang et al., 2020; Bobu et al., 2018; Wulfmeier et al., 2018; Wang et al., 2022a).

In the setting of GDA, where unlabeled intermediate data is available to the learner, Kumar et al. (2020) proposed a simple yet effective algorithm, gradual self-training (GST), which applies self-training consecutively along the sequence of intermediate domains towards the target. Kumar et al. (2020) also proved an upper bound on the target error of GST, but it is pessimistic and unrealistic in practice. In particular, given source error ε_0 and T intermediate domains each with n unlabelled data, the bound of Kumar et al. (2020) scales as $e^{\mathcal{O}(T)}\big(\varepsilon_0 + \mathcal{O}\big(\sqrt{\log(T)/n}\big)\big)$, which grows exponentially in T. This indicates that the more intermediate domains for adaptation, the worse performance that gradual self-training would obtain in the target domain. In contrast, people have empirically observed that a

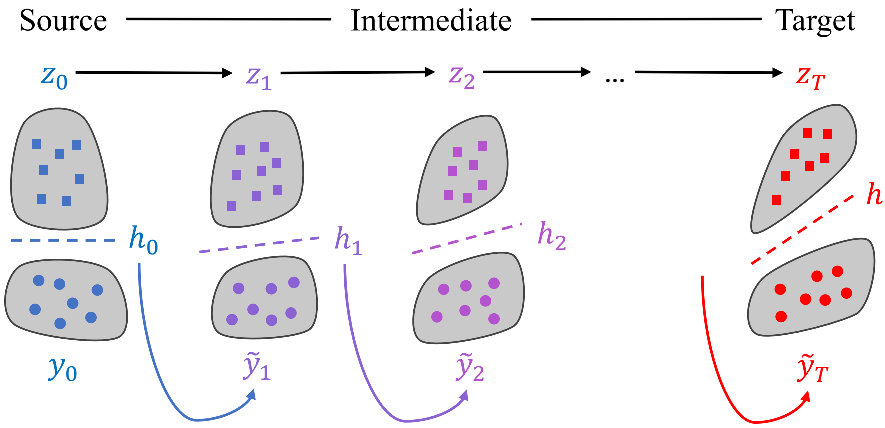


Figure 2: An illustration of the divide-and-conquer strategy to address large data distribution shift (best viewed in color). The distribution shift between the source and target is divided into $T-1$ smaller pieces with (given or generated) unlabeled intermediate data. The model h_t is gradually adapted in each step to reach the final solution.

relatively large T is beneficial for gradual domain adaptation (Abnar et al., 2021; Chen and Chao, 2021). On the other hand, despite its simplicity, the self-training algorithm already exhibits some structures of the continual changing distributions:

Observation 1 *As the number of intermediate domains T increases, the accumulated error of the self-training algorithm also increases proportionally, due to the lack of ground-truth labels and the use of pseudo-labels.*

Observation 2 *As the number of intermediate domains T increases, by using the pseudo-labels, the effective sample size used by the self-training algorithm scales as $\mathcal{O}(nT)$.*

Clearly, there is a fundamental tradeoff in the number of intermediate domains T on the error of the self-training algorithm over the sequence of distributions. The existing generalization bound given by Kumar et al. (2020) does not characterize this phenomenon. Furthermore, due to the exponential scaling factor, this upper bound becomes vacuous when T is only moderately large. Based on the above two observations and the sharp gap between existing theory and empirical observations of gradual domain adaptation, we attempt to address the following important and fundamental questions:

> *For gradual domain adaptation, given the source domain and target domain, how does the number of intermediate domains impact the target generalization error? Is there an optimal choice for this number? If yes, then how to construct the optimal path of intermediate domains?*

To answer these questions, we first carry out a novel theoretical analysis on gradual self-training (Kumar et al., 2020), then present a practical algorithm accordingly, which significantly outperforms vanilla gradual self-training. For the theoretical analysis, our setting is more general than that of Kumar et al. (2020), in the sense that i) we have a milder assumption on the distribution shift, ii) we put almost no restriction on the loss function,

Table 1: Comparison between our theoretical analysis and Kumar et al. (2020). Our analysis is applicable to a more general setting and the generalization error bound is exponentially tighter in terms of the dependency on T.

	Kumar et al. (2020)	Our Result
Applicable Loss Functions	Ramp loss	All $\rho-$Lipschitz losses
Applicable Distance Metrics	$\infty-$Wasserstein metric	All $p-$Wasserstein metrics
Generalization Error Bound	$e^{\mathcal{O}(T)}\big(\varepsilon_0 + \mathcal{O}\big(\sqrt{\log(T)/n}\big)\big)$	$\varepsilon_0{+}\mathcal{O}\big(T\Delta{+}\frac{T}{\sqrt{n}}\big){+}\widetilde{\mathcal{O}}\big(\frac{1}{\sqrt{nT}}\big)$

and iii) our technique applies to all the p-Wasserstein distance metrics. As a comparison, existing analysis is restricted to ramp loss[1] and only applies to the ∞-Wasserstein metric. At a high level, we first focus on analyzing a pair of consecutive domains, and upper bound the error difference of any classifier over domains bounded by their p-Wasserstein distance; then, we telescope this lemma to the entire path over a sequence of domains, and finally obtain an error bound for gradual self-training: $\varepsilon_0{+}\mathcal{O}\big(T\Delta{+}\frac{T}{\sqrt{n}}\big){+}\widetilde{\mathcal{O}}\big(\frac{1}{\sqrt{nT}}\big)$, where Δ is the average p-Wasserstein distance between consecutive domains. We summarize the improvement of our analysis compared with Kumar et al. (2020) in Table 1.

Interestingly, our bound indicates the existence of an optimal choice of T that minimizes the generalization error, which could explain the success of moderately large T used in practice. Notably, the $T\Delta$ in our bound could be interpreted as the length of the path of intermediate domains bridging the source and target, suggesting that one should also consider minimizing the path length $T\Delta$ in practices of gradual domain adaptation. For example, given fixed source and target domains, the path length $T\Delta$ is minimized as the intermediate domains are distributed along the Wasserstein geodesic between the source domain and target domain.

The above insight is particularly helpful under the situation where intermediate domains are missing or scarce, which is often the case in real-world applications. It inspires a natural method to generate more intermediate domains useful for GDA. Based on this finding, we propose Generative Gradual Domain Adaptation with Optimal Transport (GOAT). At a high-level, GOAT contains the following steps:

i Generate intermediate domains (z_t in Figure 2) between each pair of consecutive given domains along the Wasserstein geodesic in a feature space.

ii Apply gradual self-training (GST) over the sequence of given and generated domains. This produces a sequence of models h_t and pseudo-labels $\tilde{y}_t$ as demonstrated in Figure 2.

Empirically, we conduct experiments on Rotated MNIST, Color-Shift MNIST, Portraits (Ginosar et al., 2015) and Cover Type (Blackard and Dean, 1999), four benchmark datasets commonly used in the literature of GDA. The experimental results show that our GOAT significantly outperforms vanilla GDA, especially when the number of given intermediate domains is small. The empirical results also confirm the theoretical insights: i) when the

[1] Ramp loss can be seen as a truncated hinge loss so that it is bounded and more amenable for technical analysis.

distribution shift between a pair of consecutive domains is large, one can generate more intermediate domains to further improve the performance of GDA; ii) there exists an optimal choice for the number of generated intermediate domains.

2. Preliminaries

Notation $\mathcal{X}, \mathcal{Y}$ denote the input and the output space, and X, Y denote random variables taking values in $\mathcal{X}, \mathcal{Y}$. In this work, each domain has a data distribution μ over $\mathcal{X} \times \mathcal{Y}$, thus it can be written as $\mu = \mu(X, Y)$. When we only consider samples and disregard labels, we use $\mu(X)$ to refer to the sample distribution of μ over the input space $\mathcal{X}$.

2.1 Problem Setup

Binary Classification In the theoretical anlysis, we focus on binary classification with labels $\{-1, 1\}$. Also, we consider $\mathcal{Y}$ as a compact space in $\mathbb{R}$.

Gradually shifting distributions We have $T{+}1$ domains indexed by $\{0, 1, ..., T\}$, where domain 0 is the source domain, domain T is the target domain and domain $1, \ldots, T{-}1$ are the intermediate domains. These domains have distributions over $\mathcal{X} \times \mathcal{Y}$, denoted as $\mu_0, \mu_1, \ldots, \mu_T$.

Classifier and Loss Consider the hypothesis class as $\mathcal{H}$ and the loss function as ℓ. We define the population loss of classifier $h \in \mathcal{H}$ in domain t as

$$\varepsilon_t(h) \equiv \varepsilon_{\mu_t}(h) \triangleq \mathbb{E}_{\mu_t}[\ell(h(X),\ Y)] = \mathbb{E}_{X,Y \sim \mu_t}[\ell(h(X),\ Y)]$$

Unsupervised Domain Adaptation (UDA) In UDA, we have a source domain and a target domain. During the training stage, the learner can access m labeled samples from the source domain and n unlabeled samples from the target domain. In the test stage, the trained model will then be evaluated by its prediction accuracy on samples from the target domain. The objective for UDA is to find the classifier $h^\star$ which minimizes the loss on the target domain

$$h^\star = \underset{h \in \mathcal{H}}{\arg\min}\, \mathbb{E}_{X,Y \sim \mu_t}[\ell(h(X),\ Y)]. \tag{1}$$

Gradual Domain Adaptation (GDA) Most UDA algorithms adapt models from the source to target in a one-step fashion, which can be challenging when the distribution shift between the two is large. Instead, in the setting of GDA, there exists a sequence of additional $T - 1$ unlabeled intermediate domains bridging the source and target. We denote the underlying data distributions of these intermediate domains as $\mu_1(X, Y), \ldots, \mu_{T-1}(X, Y)$, with $\mu_0(X, Y)$ and $\mu_T(X, Y)$ being the source and target domains, respectively. In this case, for each domain $t \in \{1, \ldots, T\}$, the learner has access to S_t, a set of n unlabeled data drawn i.i.d. from $\mu_t(X)$. Same as UDA, the goal of GDA is still to make accurate predictions on test data from the target domain (Eq. 1), while the learner can train over m labeled source data and nT unlabeled data from $\{S_t\}_{t=1}^{T}$. To contrast the setting of UDA and GDA, we provide an illustration in Fig. 1 that compares UDA with GDA, using the Rotated MNIST dataset as an example.

We make a mild assumption on the input data below, which can be easily achieved by data preprocessing. This assumption is common in machine learning theory works (Cao and Gu, 2019; Arora et al., 2019; Rakhlin and Sridharan, 2014).

Assumption 1 (Bounded Input Space) *Consider the input space $\mathcal{X}$ is compact and bounded in the d-dimensional unit L_2 ball, i.e., $\mathcal{X} \subseteq \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$.*

This assumption effectively normalizes the input space and eliminates explicit dependence on the input dimension d from our bounds[2].

To quantify distribution shifts between domains, we adopt the well-known Wasserstein distance metric in the Kantorovich formulation (Kantorovich, 1939), which is widely used in the optimal transport literature (Villani, 2009).

Definition 1 (p-Wasserstein Distance) *Consider two measures μ and ν over $\mathbb{S} \subseteq \mathbb{R}^d$. For any $p \geq 1$, given a distance metric d, their p-Wasserstein distance is defined as*

$$W_p(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{S}\times\mathbb{S}} d(x, y)^p \, \mathrm{d}\gamma(x, y) \right)^{1/p} \tag{2}$$

where $\Gamma(\mu, \nu)$ is the set of all measures over $\mathbb{S} \times \mathbb{S}$ with marginals equal to μ and ν respectively.

In this paper, we consider p as a preset constant satisfying $p \geq 1$. Then, we can use the p-Wasserstein metric to measure the distribution shifts between consecutive domains.

Definition 2 (Distribution Shifts) *For $t = 1, \ldots, T$, denote*

$$\Delta_t = W_p(\mu_{t-1}, \mu_t) \tag{3}$$

Then, we define the average of distribution shifts between consecutive domains as

$$\Delta = \frac{1}{T} \sum_{t=1}^{T} \Delta_t \tag{4}$$

Remarks on Wasserstein Metrics The p-Wasserstein metric has been widely adopted in many sub-areas of machine learning, such as generative models (Arjovsky et al., 2017; Tolstikhin et al., 2018) and domain adaptation (Courty et al., 2014, 2016, 2017; Redko et al., 2019). Most of these works use $p = 1$ or 2, which is known to be good at quantifying many real-world data distributions (Peyré et al., 2019). However, the analysis in Kumar et al. (2020) only applies to $p = \infty$, which is uncommon in practice and can lead to a loose upper bound due to the monotonicity property of W_p. Since W_∞ distance focuses on the maximum transportation cost between the measures, it is more prone to unboundedness, making it a less robust choice compared with W_1 and W_2.

2.2 Gradual Self-Training

The vanilla self-training algorithm (denoted as ST) adapts classifier h with empirical risk minimization (ERM) over pseudo-labels generated on an unlabelled dataset S, i.e.,

$$h' = \mathrm{ST}(h, S) = \arg\min_{f \in \mathcal{H}} \sum_{x \in S} \ell(f(x), h(x)) \tag{5}$$

[2] If we instead assumed $\|x\|_2 \leq \sqrt{d}$, which would correspond to $\mathcal{X} \subseteq [0, 1]^d$, the constant B in Assumption 4 would gain a factor of $\sqrt{d}$.

where $h(x)$ represents pseudo-labels provided by the trained classifier h, and h' is the new classifier fitted to the pseudo-labels. The technique of hard labelling (i.e., converting $h(x)$ to one-hot labels) is used in some practices of self-training (Xie et al., 2020; Van Engelen and Hoos, 2020), which can be viewed as adding a small modification to the loss function ℓ.

Gradual self-training (Kumar et al., 2020), applies self-training to the intermediate domains and the target domain successively, i.e., for $t = 1, \dots, T$,

$$h_t = \mathrm{ST}(h_{t-1}, S_t) = \arg\min_{f \in \mathcal{H}} \sum_{x \in S_t} \ell(f(x), h_{t-1}(x)) \tag{6}$$

where h_0 is the model fitted on the source data. h_T is the final trained classifier that is expected to enjoy a low population error in the target domain, i.e., ε_T.

Intuitively, one can expect that when the distribution shift between each consecutive pair of intermediate domains is large, the quality of the pseudo-labels obtained from the previous classifier can degrade significantly, hence hurting the final target generalization. This scenario is particularly relevant when the number of given intermediate domains is relatively small.

3. Theoretical Analyses

In this section, we theoretically analyze gradual self-training under assumptions more relaxed than Kumar et al. (2020), and obtain a significantly improved error bound. Our theoretical analysis is roughly split into two steps: i) we focus on a pair of arbitrary consecutive domains with bounded distributional distance, and upper bound the prediction error difference of any classifier in the two domains by the distributional distance (Lemma 1); ii) we view gradual self-training from an online learning perspective, and adopt tools in the online learning literature to analyze the algorithm together with results of step (i), leading to an upper bound (Theorem 1) of the target generalization error of gradual self-training. Notably, our bound provides several profound insights on the optimal path of intermediate domains used in gradual domain adaptation (GDA), and also sheds light on the design of GDA algorithms. The proofs of all theoretical statements are provided in Appendix A.

3.1 Error Difference over Distribution Shift

Intuitively, gradual domain adaptation (GDA) splits the large distribution shift between the source domain and target domain into smaller shifts that are segmented by intermediate domains. Thus, in the view of reductionism (Anderson, 1972), one should understand what happens in a pair of consecutive domains in order to comprehend the entire GDA mechanism.

To start, we adopt three assumptions from the prior work (Kumar et al., 2020)[3].

Assumption 2 (R-Lipschitz Classifier) *We assume each classifier $h \in \mathcal{H}$ is R-Lipschitz in ℓ_2 norm, i.e., $\forall x, x' \in \mathcal{X}$,*

$$|h(x) - h(x')| \le R\|x - x'\|_2$$

[3] Assumption 3 is not explicitly made by Kumar et al. (2020). Instead, they directly assume the loss function to be ramp loss, which is a more strict assumption than our Assumption 3.

Assumption 3 (ρ-Lipschitz Loss) *We assume the loss function ℓ is ρ-Lipschitz, i.e., $\forall y, y' \in \mathcal{Y}$,*

$$|\ell(y,\cdot) - \ell(y',\cdot)| \leq \rho\|y - y'\|_2 \tag{7}$$

$$|\ell(\cdot,y) - \ell(\cdot,y')| \leq \rho\|y - y'\|_2 \tag{8}$$

Assumption 4 (Bounded Model Complexity) [4] *We assume the Rademachor complexity (Bartlett and Mendelson, 2002), $\mathcal{R}$, of the hypothesis class, $\mathcal{H}$, is bounded for any distribution μ considered in this paper. That is, for some constant $B > 0$,*

$$\mathcal{R}_n(\mathcal{H};\mu) = \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(x_i)\right] \leq \frac{B}{\sqrt{n}} \tag{9}$$

where the expectation is w.r.t. $x_i \sim \mu(X)$ and $\sigma_i \sim \text{Uniform}(\{-1, 1\})$ for $i = 1, \ldots, n$.

With these assumptions, we can bound the population error difference of a classifier between a pair of shifted domains in the following proposition. The proof is in Appendix A.1.

Lemma 1 (Error Difference over Shifted Domains) *Consider two arbitrary measures μ, ν over $\mathcal{X} \times \mathcal{Y}$. Then, for arbitrary classifier h and loss function ℓ satisfying Assumption 2, 3, the population loss of h on μ and ν satisfies*

$$|\varepsilon_\mu(h) - \varepsilon_\nu(h)| \leq \rho\sqrt{R^2+1}\; W_p(\mu,\nu) \tag{10}$$

where W_p is the Wasserstein-p distance metric and $p \geq 1$.

Eq. (6) depicts each iteration of gradual self-training with an past classifier h_t and a new one h_{t+1}, which are fitted to S_t and S_{t+1}, respectively. Naturally, one might be curious about how well the performance of h_{t+1} in domain t+1 is compared with h_t in domain t. We answer this question as follows, with proof in Appendix A.2.

Proposition 1 (The stability of the ST algorithm) *Consider two arbitrary measures μ, ν, and denote S as a set of n unlabelled samples i.i.d. drawn from μ. Suppose $h \in \mathcal{H}$ is a pseudo-labeler that provides pseudo-labels for samples in S. Define $\hat{h} \in \mathcal{H}$ as an ERM solution fitted to the pseudo-labels,*

$$\hat{h} = \arg\min_{f\in\mathcal{H}} \sum_{x\in S} \ell(f(x), h(x)) \tag{11}$$

Then, for any $\delta \in (0,1)$, the following bound holds true with probability at least $1-\delta$,

$$\left|\varepsilon_\mu(\hat{h}) - \varepsilon_\nu(h)\right| \leq \mathcal{O}\left(W_p(\mu,\nu) + \frac{\rho B + \sqrt{\log\frac{1}{\delta}}}{\sqrt{n}}\right) \tag{12}$$

Comparison with Kumar et al. (2020) The setting of Kumar et al. (2020) is more restrictive than ours. For example, its analysis is specific to ramp loss (Huang et al., 2014), a rarely used loss function for binary classification. Kumar et al. (2020) also studies the

[4] This assumption is actually reasonable and not strong. For example, under Assumption 1 and 2, linear models directly satisfy (9), as proved in (Kumar et al., 2020; Liang, 2016).

error difference over consecutive domains, and prove a multiplicative bound (in Theorem 3.2 of Kumar et al. (2020)), which can be re-expressed in terms of our notations and assumptions as

$$\varepsilon_\mu(\hat{h}) \le \frac{2}{1-R\Delta_\infty}\varepsilon_\nu(h)+\varepsilon_\mu^*+\mathcal{O}\left(\frac{\rho B+\sqrt{\log\frac{1}{\delta}}}{\sqrt{n}}\right) \tag{13}$$

where $\varepsilon_\mu^* \triangleq \min_{f\in\mathcal{H}} \varepsilon_\mu(f)$ is the optimal error of $\mathcal{H}$ in μ, and $\Delta_\infty \triangleq \max_{y\in\{-1,1\}}(W_\infty(\mu(X|Y=y),\nu(X|Y=y)))$ can be seen as an analog to the $W_p(\mu,\nu)$ in (12). Kumar et al. (2020) assumes $1-R\Delta_\infty > 0$, thus the error $\varepsilon_\nu(h)$ is increased by the factor $\frac{2}{1-R\Delta_\infty} > 1$ in the above error bound of $\varepsilon_\mu(\hat{h})$. This leads to a target domain error bound *exponential* in T (Corollary 3.3. of Kumar et al. (2020)) when one applies (13) to the sequence of domains iteratively in gradual self-training (i.e., Eq. (6)). In contrast, our (12) indicates $\varepsilon_\mu(\hat{h}) \le \varepsilon_\nu(h) +$ other terms, which increases the error $\varepsilon_\nu(h)$ in an *additive* way, leading to a target domain error bound *linear* in T.

Remarks on Generality Lemma 1 and Proposition 1 are not restricted to gradual domain adaptation. Of independent interest, they can be leveraged as useful theoretical tools to handle distribution shifts in other machine learning problems, including unsupervised domain adaptation, transfer learning, out-of-distribution (OOD) robustness, and group fairness.

3.2 An Online Learning View of GDA

One can naively apply Proposition 1 to gradual self-training over the sequence of domains (i.e., Eq. (6)) iteratively and obtain an error bound of the target domain as

$$\varepsilon_T(h_T) \le \varepsilon_0(h_0) + \mathcal{O}\left(T\Delta+T\frac{\rho B+\sqrt{\log\frac{1}{\delta}}}{\sqrt{n}}\right) \tag{14}$$

Obviously, the larger T, the higher the error bound becomes (this holds even if one assumes $T\Delta \le$ constant for fixed source and target domains). However, this contradicts with empirical observations that a moderately large T is optimal (Kumar et al., 2020; Abnar et al., 2021; Chen and Chao, 2021).

To resolve this discrepancy, we take an online learning view of gradual domain adaptation, which allows us to obtain a more optimistic error bound. Specifically, we consider the domains $t=0,\dots,T$ arriving sequentially to the model. This process can be formalized as follows. For each domain $t=0,\dots,T$:

1. Observe unlabeled data $S_t=\{x_i^t\}_{i=1}^n$ from domain μ_t.

2. If $t=0$ (source domain), use true labels. Otherwise, generate pseudo-labels $\hat{y}_i^t = h_{t-1}(x_i^t)$ using the previous model h_{t-1}.

3. Update the model: $h_t = \arg\min_{f\in\mathcal{H}}\sum_{i=1}^n \ell(f(x_i^t),\hat{y}_i^t)$.

This online learning perspective allows us to leverage tools from sequential learning theory, particularly the framework of Rakhlin et al. (2015), which views online binary classification as a process on a complete binary tree. By applying this view, we can utilize results on sequential Rademacher complexity (Definition 4) and the discrepancy measure between distributions (Definition 5). The key advantage of this approach is that it enables us to obtain bounds that depend on the total number of samples nT, rather than just T as in the naive approach. Specifically, terms of order $\mathcal{O}(\sqrt{1/T})$ in the naive bound become $\mathcal{O}(\sqrt{1/nT})$ in our improved bound. Moreover, this view allows us to better characterize how the error accumulates across domains, leading to the improved linear dependence on T in our final bound (Theorem 1), compared to the exponential dependence in previous work (Kumar et al., 2020).

To proceed, certain structural assumptions and complexity measures are necessary. For example, VC dimension (Vapnik, 1999) and Rademacher complexity (Bartlett and Mendelson, 2002) are proposed for supervised learning. Similarly, in online learning, Littlestone dimension (Littlestone, 1988), sequential covering number (Rakhlin et al., 2010) and sequential Rademacher complexity (Rakhlin et al., 2010, 2015) are developed as useful complexity measures. To study gradual self-training in an online learning framework, we adopt the framework of Rakhlin et al. (2015), which views online binary classification as a process in the structure of a *complete binary tree* and defines the *sequential Rademacher complexity* upon that.

Definition 3 (Complete Binary Trees) *We define two complete binary trees $\mathscr{X}, \mathscr{Y}$, and the path $\boldsymbol{\sigma}$ in the trees:*
$\mathscr{X} \triangleq (\mathscr{X}_0, ..., \mathscr{X}_T)$*, a sequence of mappings with* $\mathscr{X}_t : \{\pm 1\}^t \to \mathcal{X}$ *for* $t = 0, ..., T$.
$\mathscr{Y} \triangleq (\mathscr{Y}_0, ..., \mathscr{Y}_T)$*, a sequence mappings with* $\mathscr{Y}_t : \{\pm 1\}^t \to \mathcal{Y}$ *for* $t = 0, ..., T$.
$\boldsymbol{\sigma} = (\sigma_0, ..., \sigma_T) \in \{\pm 1\}^t$*, a path in* $\mathscr{X}$ *or* $\mathscr{Y}$.

Definition 4 (Sequential Rademacher Complexity) *Consider $\boldsymbol{\sigma}$ as a sequence of Rademacher random variables and a t-dimensional probability vector $\mathbf{q}_t = (q_0, ..., q_{t-1})$, then the sequential Rademacher complexity of $\mathcal{H}$ is*

$$\mathcal{R}_t^{\text{seq}}(\mathcal{H}) = \sup_{\mathscr{X}, \mathscr{Y}} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \sum_{\tau=0}^{t-1} \sigma_\tau q_\tau \ell\big(h(\mathscr{X}_\tau(\boldsymbol{\sigma})), \mathscr{Y}_\tau(\boldsymbol{\sigma})\big) \right]$$

To better understand this measure, we present examples of two common model classes[5], which are provided in Rakhlin and Sridharan (2014).

Example 1 (Linear Models) *For the linear model class that is R-Liphschtiz, i.e., $\mathcal{H} = \{x \to w^\top x : \|w\|_2 \leq R\}$, we have $\mathcal{R}_t^{\text{seq}}(\mathcal{H}) \leq \frac{R}{\sqrt{t}}$ for $t \in \mathbb{Z}_+$.*

Example 2 (Neural Networks) *Consider $\mathcal{H}$ as the hypothesis class of R-Lipschitz L-layer fully-connected neural nets with 1-Lipschitz activation function (e.g., ReLU, Sigmoid, TanH). Then, its sequential Rademacher complexity is bounded as $\mathcal{R}_t^{\text{seq}}(\mathcal{H}) \leq \mathcal{O}\left(R\sqrt{\frac{(\log t)^{3(L-1)}}{t}}\right)$ for $t \in \mathbb{Z}_+$.*

[5] The probability vector $\mathbf{q}_t$ is taken to be uniform in these cases.

Besides the model complexity measure, we also adopt a measure of discrepancy among multiple data distributions, which is proposed in works of online learning for time-series data (Kuznetsov and Mohri, 2014, 2015, 2016, 2017, 2020).

Definition 5 (Discrepancy Measure) *For any t-dimensional probability vector $\mathbf{q}_t = (q_0, ..., q_{t-1})$, the discrepancy measure* $\operatorname{disc}(\mathbf{q}_t)$ *is defined as*

$$\operatorname{disc}(\mathbf{q}_t) = \sup_{h \in \mathcal{H}} \left(\varepsilon_{t-1}(h) - \sum_{\tau=0}^{t-1} q_\tau \cdot \varepsilon_\tau(h) \right) \tag{15}$$

Intuitively, this discrepancy measure quantifies the maximum difference between the error of a hypothesis on the last domain ($\varepsilon_{t-1}(h)$) and a weighted average of its errors on all previous domains ($\sum_{\tau=0}^{t-1} q_\tau \cdot \varepsilon_\tau(h)$). This measure captures how much the "difficulty" of the learning problem can change across domains. In the context of gradual domain adaptation, a small discrepancy suggests that the domains are changing gradually, making it easier for the model to adapt. Conversely, a large discrepancy indicates significant shifts between domains, which could make adaptation more challenging. The supremum over $\mathcal{H}$ in the definition ensures that we're considering the worst-case scenario across all possible hypotheses in our model class. This conservative approach helps us derive bounds that hold regardless of which specific hypothesis our learning algorithm might choose.

We can further bound this discrepancy in our setting (defined in Sec. 2) as follows. The proof is in Appendix A.3.

Lemma 2 (Discrepancy Bound) *With Lemma 1, the discrepancy measure* (15) *can be upper bounded as*

$$\operatorname{disc}(\mathbf{q}_t) \le \rho\sqrt{R^2+1} \sum_{\tau=0}^{t-1} q_\tau (t - \tau - 1)\Delta \tag{16}$$

With $\mathbf{q}_t = \mathbf{q}_t^* = (\frac{1}{t}, ..., \frac{1}{t})$, this upper bound can be minimized as

$$\operatorname{disc}(\mathbf{q}_t^*) \le \rho\sqrt{R^2+1}\; t\Delta/2 = \mathcal{O}(t\Delta) \tag{17}$$

3.3 Generalization Bound for Gradual Self-Training

With our results obtained in Section 3.1 and tools introduced in Section 3.2, we can prove a generalization bound for gradual self-training within online learning frameworks such as Kuznetsov and Mohri (2016, 2020). However, if we use these frameworks in an off-the-shelf way, the resulting generalization bound will have multiple terms with dependence on T and no dependence on n (the number of samples per domain), since these online learning works do not care about the data size of each domain. This will cause the resulting bound to be loose in terms of n. To resolve this, we come up with a novel reductive view of the learning process of gradual self-training, which is more fine-grained than the original view in Kumar et al. (2020). This reductive view enables us to make the generalization bound to depend on n in an intuitive way, which also tightens the final bound. We defer explanations of this view to Appendix A.4 along with the proof of Theorem 1.

Finally, we prove a generalization bound for gradual self-training that is much tighter than that of Kumar et al. (2020).

Theorem 1 (Generalization Bound for Gradual Self-Training) *For any $\delta \in (0,1)$, the population loss of gradually self-trained classifier h_T in the target domain is upper bounded with probability at least $1-\delta$ as*

$$\varepsilon_T(h_T) \le \sum_{t=0}^{T} q_t \varepsilon_t(h_t) + \|\mathbf{q}_{n(T+1)}\|_2 \left(1+\mathcal{O}\left(\sqrt{\log(1/\delta)}\right)\right)$$
$$+\mathrm{disc}(\mathbf{q}_{T+1})+\mathcal{O}\left(\sqrt{\log T}\mathcal{R}^{\mathrm{seq}}_{n(T+1)}(\ell \circ \mathcal{H})\right) \tag{18}$$

For the class of neural nets considered in Example 2,

$$\varepsilon_T(h_T) \le \varepsilon_0(h_0) + \mathcal{O}\left(T\Delta+\frac{T}{\sqrt{n}}+T\sqrt{\frac{\log 1/\delta}{n}}+\frac{1}{\sqrt{nT}}+\sqrt{\frac{(\log nT)^{3L-2}}{nT}}+\sqrt{\frac{\log 1/\delta}{nT}}\right) \tag{19}$$

Remark The bound in Eq. (19) is rather intuitive[6]: the first term $\varepsilon_0(h_0)$ is the source error of the initial classifier, and $T\Delta$ corresponds to the total length of the path of intermediate domains connecting the source domain and the target domain. The asymptotic $\mathcal{O}(T/\sqrt{n})$ term is due to the accumulated estimation error of the pseudo-labeling algorithm incurred at each step. The $\mathcal{O}(1/\sqrt{nT})$ term characterizes the overall sample size used by the algorithm along the path, i.e., the algorithm has seen n samples in each domain, and there are T total domains that gradual self-training runs on.

Comparison with Kumar et al. (2020) Using our notation, the generalization bound of Kumar et al. (2020) can be re-expressed as

$$\varepsilon_T(h_T) \le e^{\mathcal{O}(T)}\left(\varepsilon_0(h_0)+\mathcal{O}(\frac{1}{\sqrt{n}}+\sqrt{\frac{\log T}{n}})\right), \tag{20}$$

which grows *exponentially* in T as a multiplicative factor. In contrast, our bound (19) grows only additively and linearly in T, achieving an *exponential improvement* compared with the bound of Kumar et al. (2020) shown in (20).

3.4 Optimal Path of Gradual Self-Training

It is worth pointing out that our generalization bound in Theorem 1 applies to any path connecting the source domain and target domain with T steps, as long as μ_0 is the source domain and μ_T is the target domain. In particular, if we define $\Delta_{\max}$ to be an upper bound[7] on the average W_p distance between any pair of consecutive domains along the path, i.e., $\Delta_{\max} \ge \frac{1}{T}\sum_{t=1}^{T} W_p(\mu_{t-1},\mu_t)$, and let $\mathcal{P}$ to be the collection of paths with T steps connecting μ_0 and μ_T:

$$\mathcal{P} := \{(\mu_t)_{t=0}^{T} \mid \frac{1}{T}\sum_{t=1}^{T} W_p(\mu_{t-1},\mu_t) \le \Delta_{\max}\} \ ,$$

then we can extend the generalization bound in Theorem 1:

$$\varepsilon_T(h_T) \le \varepsilon_0(h_0)+\inf_{\mathcal{P}} \widetilde{\mathcal{O}}\left(T\Delta_{\max}+\frac{T}{\sqrt{n}}+\sqrt{\frac{1}{nT}}\right) \tag{21}$$

[6] Eq. (19) is derived from Eq. (18) by substituting the expression for $\mathcal{R}^{\mathrm{seq}}_{n(T+1)}(\ell \circ \mathcal{H})$ from Example 2.

[7] Need to be large enough to ensure that $\mathcal{P}$ is non-empty.

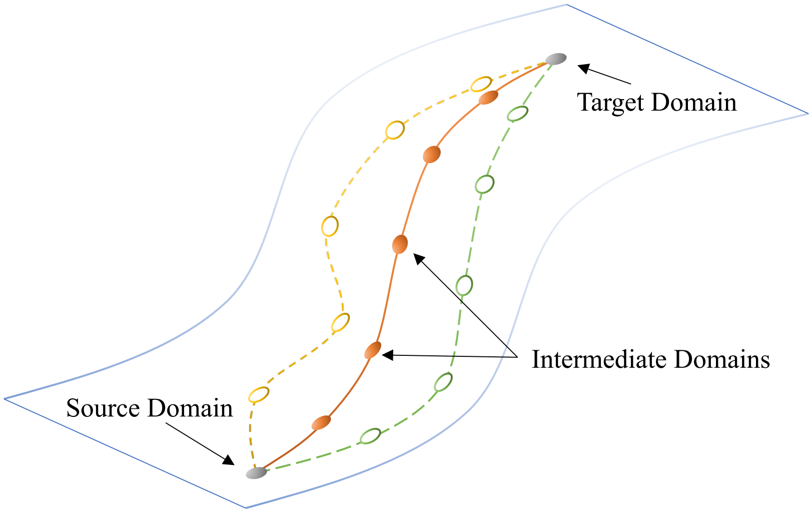


Figure 3: An illustration of the optimal path in gradual domain adaptation, with a detailed explanation in Sec. 3.4. The orange path is the geodesic connecting the source domain and target domain.

Minimizing the RHS of the above upper bound w.r.t. T (the proof is provided in Appendix A.6), we obtain the optimal choice of T on the order of

$$\widetilde{\mathcal{O}}\left(\left(\frac{1}{1+\Delta_{\max}\sqrt{n}}\right)^{2/3}\right). \tag{22}$$

However, the above asymptotic optimal length may not be achievable, since we need to ensure that $T\Delta_{\max}$ is at least the length of the geodesic connecting the source domain and target domain. To this end, define L to be the W_p distance between the source domain and target domain, we thus have the optimal choice T^* as

$$T^* = \max\left\{\frac{L}{\Delta_{\max}}, \widetilde{\mathcal{O}}\left(\left(\frac{1}{1+\Delta_{\max}\sqrt{n}}\right)^{2/3}\right)\right\}. \tag{23}$$

Intuitively, the inverse scaling of T^* and $\Delta_{\max}$ suggests that, if the average distance between consecutive domains is large, it is better to take fewer intermediate domains.

Illustration of the Optimal Path To further illustrate the notion of the optimal path connecting the source domain and target domain implied by our theory, we provide an example in Fig. 3. Consider the metric space induced by W_p over all the joint distributions with finite p-th moment, where both the source and target could be understood as two distinct points. In this case, there are infinitely many paths of step size T connecting the source and target, such that the average pairwise distance is bounded by $\Delta_{\max}$. Hence, one insight we can draw from Eq. (21) is that: if the learner could construct the intermediate domains, then it is better to choose the path that is as close to the geodesic, i.e., the shortest path between the source and target (under W_p), as possible. This key observation opens a broad avenue forward toward algorithmic designs of gradual domain adaptation to *construct* intermediate domains for better generalization performance in the target domain.

4. Generative Gradual Domain Adaptation with Optimal Transport

Inspired by the theoretical results, in this section, we present our algorithm to automatically generate a series of intermediate domains between any pair of consecutive given domains, with the hope that when applied to the sequence of generated intermediate domains, GST could lead to better target generalization. Before presenting the proposed algorithm, we first formally introduce several notions that will be used in the design of our algorithm.

The optimal transport problem was initially formalized by Monge (1781), and Kantorovich (1939) further relaxed the deterministic nature of Monge's problem formulation. In this part, we adopt Kantorovich's formulation of optimal transport (Kantorovich, 1939), which aims at finding the optimal coupling that minimizes a total transport cost.

Definition 6 (Optimal Coupling) *Given measures μ, ν over $\mathcal{X}$ and a lower semi-continuous cost function*[8] $c : \mathcal{X} \times \mathcal{X} \mapsto [0, \infty)$*, the optimal transport coupling γ^* is the one that attains the infimum of the total transport cost:*

$$\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} c(x, x') d\gamma(x, x') \ . \tag{24}$$

where $\Gamma(\mu, \nu)$ is the set of all probability measures on $\mathcal{X} \times \mathcal{X}$ with marginals as μ, ν.

One can create a path of measures that interpolates the given two, and the theory of optimal transport can help us find the optimal path that minimizes the path length measured under the Wasserstein metric, i.e., the sum of Wasserstein distances between each pair of consecutive measures along the path. This optimal path is termed the Wasserstein geodesic, which is formally defined below.

Definition 7 (Wasserstein Geodesic) *Given two measures ν_0, ν_1 over $\mathcal{X}$ and an optimal coupling $\gamma^\star$. Let $\sharp$ denote the push-forward operator on measures. Then, a (constant-speed) Wasserstein geodesic between ν_0, ν_1 under Euclidean metric can be defined by the path $\mathcal{P}(\nu_0, \nu_1) \coloneqq \{(g_t)_\sharp \gamma^\star : t \in [0, 1]\}$, where $g_t(x, y) = (1 - t)x + ty$.*

4.1 Motivations

The target domain error bound of gradual self-training, i.e., Eq. (19), has a dominant term $T\Delta$, which can be interpreted as the length of the path of intermediate domains connecting the source and target. Interestingly, we find that this path is related to the **Wasserstein geodesic** between the source μ_0 and target μ_T, and we formalize our findings as follows.

Proposition 2 (Path Length of Intermediate Domains) *For arbitrary intermediate domains $\mu_1, \ldots, \mu_{T-1}$, the following inequality holds,*

$$T\Delta = \sum_{t=1}^{T} W_p(\mu_{t-1}, \mu_t) \geq W_p(\mu_0, \mu_T), \tag{25}$$

[8] The existence of an optimal transport plan is contingent on the cost being lower semi-continuous. See, e.g., Proposition 2.1 from Villani (2021).

where the equality is obtained if and only if the intermediate domains $\mu_1, \ldots, \mu_{T-1}$ sequentially fall along the Wasserstein geodesic between μ_0 and μ_T.

Without explicit access to the intermediate domains, gradual domain adaptation cannot be applied. Interestingly, Proposition 2 sheds light on the task of intermediate domain generation to bridge this gap: *the generated intermediate domains should fall on or close to the Wasserstein geodesic in order to minimize the path length.*

Note that in GDA, we cannot directly measure Δ since it requires access to the joint distributions of the intermediate domains, whereas only unlabeled data are available to us. In order to bridge the gap, in this paper, we make the following assumption of the intermediate domains.

Assumption 5 (Feature Space) *There exists a feature space $\mathcal{Z}$ such that the covariate shift assumption holds over $\mathcal{Z}$. Specifically, the conditional distribution of Y given the feature Z is invariant across all the intermediate domains, i.e., for any two domains i and j with $i \neq j$, $\mu_i(Y|Z) = \mu_j(Y|Z)$.*

Note that covariate shift is one of the most common assumptions in the literature of domain adaptation (Ben-David et al., 2007; Adel et al., 2017; Arjovsky et al., 2019; Redko et al., 2019; Zhao et al., 2019b; Rosenfeld et al., 2020; Wang et al., 2022b). It is one way to ensure that the knowledge contained in different domains are inherently relevant such that the success of domain adaptation is possible (Zhang et al., 2013). It has been widely applied in various applications, including computer vision (Adel et al., 2017; Arjovsky et al., 2019; Redko et al., 2019; Zhao et al., 2019b), natural language processing (Ash et al., 2016), and robot control (Akiyama et al., 2010; Sugiyama, 2013). Under this assumption, the Wasserstein distance between the joint distance $W_p\left(\mu_{t-1}(Z,Y),\ \mu_t(Z,Y)\right)$ reduces to the one between the marginal feature distribution $W_p\left(\mu_{t-1}(Z),\ \mu_t(Z)\right)$.

4.2 Computation with Optimal Transport

From Definition 7, we know that one has to solve an optimal transport problem to generate intermediate domains along the Wasserstein geodesic. As a first step, we consider the optimal transport between a source domain and a target domain.

Solve Optimal Transport with Linear Programming In unsupervised domain adaptation (UDA), the source and target domains have finite training data. Hence, we can consider the measures of the source and target to be discrete, i.e., μ_0 and μ_T only have probability mass over the finite training data points. More formally, denoting the source dataset as $S_0 = \{x_{0i}\}_{i=1}^m$ and target dataset as $S_T = \{x_{Tj}\}_{i=1}^n$, the empirical measures μ_0 and μ_T can be expressed as

$$\mu_0 = \frac{1}{m}\sum_{i=1}^m \delta(x_{0i}) \ , \quad \mu_T = \frac{1}{n}\sum_{j=1}^n \delta(x_{Tj}), \tag{26}$$

where $\delta(x)$ represents the Dirac delta distribution at x (Dirac et al., 1930). Under the discrete case, the push-forward operator $\mathcal{T}^*$ that pushes μ_0 forward to μ_T can be obtained by solving a linear program (Peyré et al., 2019).

Proposition 3 *Consider μ_0 over source data $\{x_{0i}\}_{i=1}^m$ and μ_T over target data $\{x_{Tj}\}_{i=1}^n$. Given a transport cost function $c : \mathcal{X} \times \mathcal{X} \mapsto [0, \infty)$, there exists a randomized optimal transport map (induced from the optimal coupling γ^*), $\mathcal{T}^*$, which satisfies $\mathcal{T}^*_\sharp \mu_0 = \mu_T$. Furthermore, for $i \in [m]$, $\mathcal{T}^*$ maps x_{0i} as follows,*

$$\mathcal{T}^*_\sharp \delta(x_{0i}) = \sum_{j=1}^{n} \gamma^*_{ij} \delta(x_{Tj}), \tag{27}$$

where $\gamma^* \in \mathbb{R}^{m\times n}_{\geq 0}$ is the optimal transport plan, a non-negative matrix of dimension $m \times n$. The plan γ^* can be obtained by solving the following linear program,

$$\gamma^* = \underset{\gamma \in \mathbb{R}^{m\times n}_{\geq 0}}{\arg\min} \sum_{i,j} \gamma_{i,j} c(x_{0i}, x_{Tj}) \tag{28}$$

$$s.t. \quad \gamma \mathbf{1}_n = \frac{1}{m}\mathbf{1}_m \quad and \quad \gamma^T \mathbf{1}_m = \frac{1}{n}\mathbf{1}_n$$

Generating Intermediate Domains with Optimal Transport Proposition 3 demonstrates that one can use linear programming (LP) to solve the optimal transport problem between a source dataset and a target dataset. With the optimal transport plan γ^*, one can leverage Definition 7 to generate intermediate domains along the Wasserstein geodesic. Specifically, for $t = 1, \ldots, T-1$, the measure of the intermediate domain t can be obtained by the following push-forward

$$\mu_t = \left(\frac{T-t}{T}\mathbf{Id} + \frac{t}{T}\mathcal{T}^*\right)_\sharp \mu_0 = \frac{1}{m}\sum_{i,j} \gamma^*_{ij} \delta\left(\frac{T-t}{T}x_{0i} + \frac{t}{T}x_{Tj}\right) \tag{29}$$

Intuitively, μ_t can be interpreted as a discrete measure over n_{γ^*} data points with data weights assigned by γ^*_{ij}, where $n_{\gamma^*} \coloneqq \sum_{i,j} \mathbb{1}[\gamma_{ij} > 0]$ is the number non-zero entries in the matrix γ^*.

Space Complexity Clearly, one needs to store the optimal transport plan matrix $\gamma^* \in \mathbb{R}^{m\times n}_{\geq 0}$, in order to generate intermediate domains with (29). Thus, the space complexity appears to be $\mathcal{O}(mn)$. However, by leveraging the theory of linear programming, one can show that the maximum number of non-zero elements of the solution γ^* to the LP (28) is at most $m+n-1$ (Peyré et al., 2019). Thus, the space complexity can be reduced to $\mathcal{O}(m+n)$ when using a sparse matrix format to store γ^*.

Time Complexity For simplicity, let us consider $m = n$. Then, the time complexity of solving the LP (28) is known to be $O(n^3 \log(n))$ (Cuturi, 2013; Pele and Werman, 2009).

4.3 Proposed Algorithm

We present our proposed algorithm in Algorithm 1. Notice that Algorithm 1 directly generates intermediate domains between the source and target domains. However, in practice, there might be a few given intermediate domains that can be used by GDA. In this case, one can simply treat each pair of consecutive domains as a source-target domain pair, and apply Algorithm 1 iteratively to the pairs of consecutive given domains from the source to target.

Next, we explain the key designs of the proposed algorithm.

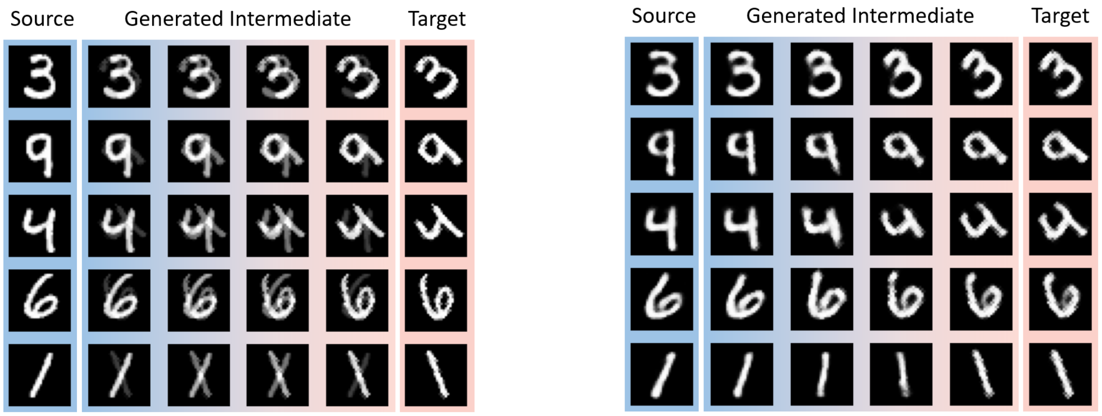
Algorithm 1 Generative Gradual Domain Adaptation with Optimal Transport (GOAT)

```
Require: $S_0^X = \{x_{0i}\}_{i=1}^m$, $S_T^X = \{x_{Ti}\}_{i=1}^n$; Encoder $\mathcal{E}$; Source-trained classifier $h_0$
  Encode: $S_0^Z = \{z_{0i} = \mathcal{E}(x_{0i})\}_{i=1}^m, S_T^Z = \{z_{Tj} = \mathcal{E}(x_{Tj})\}_{j=1}^n$
  Optimal Transport (OT): Solve for the OT plan $\gamma^* \in \mathbb{R}_{\geq 0}^{m \times n}$ between $S_0^Z$ and $S_T^Z$
  Cutoff: Use a cutoff threshold to keep $\mathcal{O}(n+m)$ elements of $\gamma^*$ above the threshold and
  zero out the rest //Only applies to the entropy-regularized version of OT
  Intermediate Domain Generation:
  for $t = 1, \ldots, T$ do
    Initialize an empty set $S_t^Z$
    for each non-zero element $\gamma_{ij}^*$ of $\gamma^*$ do
      $z \leftarrow \frac{T-t}{T} z_{0i} + \frac{t}{T} z_{Tj}$
      Add $(z, \gamma_{ij}^*)$ to $S_t$
    end for
  end for
  Gradual Domain Adaptation:
  for $t = 1, \ldots, T$ do
    $h_t = \mathrm{ST}(h_{t-1}, S_t)$ //Can also apply sample weights to losses based on $\gamma_{ij}^*$
  end for
output Target-adapted classifier $h_T$
```

4.3.1 Fast Computation of Optimal Transport (OT)

The super-cubic time complexity of solving the LP in (28) essentially prevents this optimal transport approach from being scaled up to large datasets. To remedy this issue, we propose to solve an approximate objective of the OT problem (28) when it takes too long to solve the original OT exactly. Specifically, we add an entropic regularization term to the objective (28), turning it to be strictly convex, and the time complexity of solving this regularized objective is reduced to nearly $\mathcal{O}(n^2)$ from the original $\mathcal{O}(n^3 \log n)$ (Cuturi, 2013; Dvurechensky et al., 2018). However, the solution to this regularized objective, i.e., the OT plan γ^*, is not guaranteed to have at most $n + m - 1$ elements anymore. Thus, the space complexity increases to $\mathcal{O}(mn)$ from $\mathcal{O}(m + n)$. In light of this challenge, we design a cutoff trick to zero out entries of tiny magnitude in γ^* (see details in Algo. 1), reducing the space complexity back to $\mathcal{O}(m + n)$. More details regarding this part are provided in Appendix C.

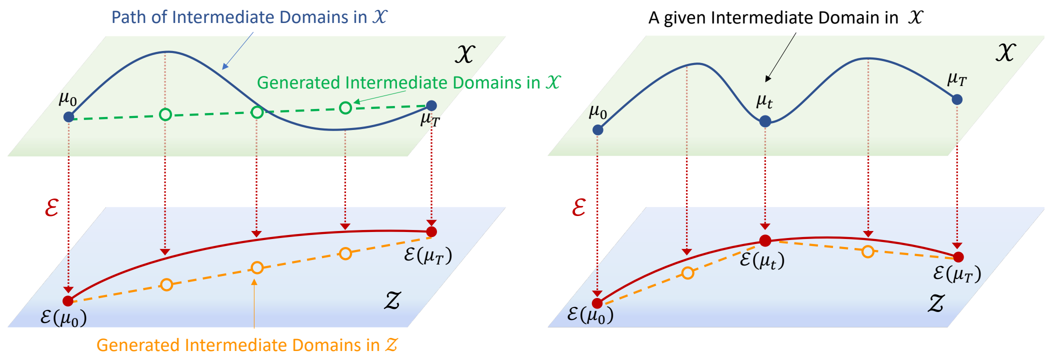
Note that beyond the Sinkhorn algorithm, several alternative approaches enhance the efficiency of OT computation. For instance, the Greenkhorn algorithm (Altschuler et al., 2017) improves the performance of the Sinkhorn algorithm, with a complexity of $\tilde{\mathcal{O}}(n^2/\varepsilon^2)$, where ε is the desired accuracy. Additionally, Low-rank Optimal Transport (LOT) (Forrow et al., 2019; Scetbon et al., 2021, 2022) approaches the problem by reducing the size of measures before solving OT. This method specifically seeks couplings of low rank, which significantly reduces computational demands. Another approach, sliced-Wasserstein distance (Bonneel et al., 2015; Kolouri et al., 2019), involves computing linear projection of high-dimensional distributions to one-dimensional distributions, then averaging the resulting Wasserstein distances, which can be computed using closed-form formulas. Given the varied applicability



(a) Input-space generation.

(b) Feature-space generation.

Figure 4: Samples from generated intermediate domains.



(a)

(b)

Figure 5: Illustration of the intermediate domain generation in GOAT. (a) without any given intermediate domain, (b) with one given intermediate domain.

and use cases of these methods, we recommend that practitioners select the OT algorithm that best suits their specific needs.

4.3.2 Intermediate Domain Generation in Feature Space

The intermediate domain generation approach described above directly generates data in the input space $\mathcal{X}$. However, the generation does not have to be restricted to the input space. One can show that with a Lipschitz continuous encoder $\mathcal{E} : \mathcal{X} \mapsto \mathcal{Z}$ mapping inputs to the feature space $\mathcal{Z}$ (i.e., $z \leftarrow \mathcal{E}(x)$ for any input x), the order of the generation bound (19) stays the same[9] (the proof is in Appendix B).

Feature Space vs. Input Space We use an encoder by default in Algorithm 1, since we empirically observe that directly generating intermediate domains in the input space is usually sub-optimal (see Figure 6a for a detailed analysis). To give the readers an intuitive understanding, we provide a demo of Rotated MNIST in Figure 4: if we apply the

[9] Some terms in the bound get multiplied by a factor of the Lipschitz constant of $\mathcal{E}$.

intermediate domain generation of Algorithm 1 in the input space, the generated data do not approximate the digit rotation well; when applying the algorithm in the latent space of a VAE (fitted to the source and target data), the generated data (obtained by the decoder of the VAE) captures the digit rotation accurately.

Figure 5a explains this superiority of the feature space over the input space with a schematic diagram: the input-space Wasserstein geodesic can not well approximate the ground-truth distribution shift (e.g., rotation) due to the linearity of push-forward operators under the Euclidean metric; with a proper encoder $\mathcal{E}$, the feature-space Wasserstein geodesic can capture the distribution shift more accurately.

Leveraging the Given Intermediate Domain(s) With a given intermediate domain, we generate intermediate domains with GOAT between the two pairs of consecutive domains, respectively. Figure 5b shows that this approach can make the generated domains closer to the ground-truth path of distribution shift, explaining why GOAT benefits from given intermediate domains.

Gradual Domain Adaptation (GDA) on Generated Intermediate Domains With the generated data of intermediate domains, one can run the GDA algorithm consecutively over the source-intermediate-target domains in the feature space. As for the choice of GDA algorithm, we adopt Gradual Self-Training (GST) (Kumar et al., 2020), mainly due to its simplicity. Nevertheless, one can freely apply any other GDA algorithm on top of the generated domains.

5. Experiments

Our goal of the experiment is to demonstrate the performance gain of training on generated intermediate domains in addition to given domains. We compare our method with gradual self-training (Kumar et al., 2020), which only self-trains a model along the sequence of given domains iteratively. In Sec. 5.4, we further analyze the choices of encoder $\mathcal{E}$ and transport plan γ^* used by Algorithm 1. More details of our experiments are provided in Appendix D.

5.1 Datasets

Rotated MNIST A semi-synthetic dataset built on the MNIST dataset (LeCun and Cortes, 1998), with 50K images as the source domain and the same 50K images rotated by 45 degrees as the target domain. Intermediate domains are evenly distributed between the source and target.

Color-Shift MNIST We normalize the pixel values of MNIST to be in $[0, 1]$. We use 50K images as the source domain and the same 50K images with pixel values shifted by 1 as the target domain, i.e., the target pixel values are shifted to be in $[1, 2]$. Intermediate domains are also evenly distributed.

Portraits (Ginosar et al., 2015) A real-world gender classification image dataset consisting of portraits of high school seniors from 1905 to 2013. Following Kumar et al. (2020), the dataset is sorted chronologically and split into a source domain (first 2000 images), 7 intermediate domains (next 14000 images), and a target domain (last 2000 images).

Cover Type (Blackard and Dean, 1999) A tabular dataset aiming at predicting the forest cover type at certain locations given 54 features. Following Kumar et al. (2020), we sort the data by the distance to water body in ascending order, splitting the data into a source domain (first 50K data), 10 intermediate domains (each with 40K data) and a target domain (final 50K data).

5.2 Implementation

Our code is built in PyTorch (Paszke et al., 2019), and our experiments are run on NVIDIA RTX A6000 GPUs. For Rotated MNIST, Color-Shift MNIST and Portraits, we use a convolutional neural network (CNN) of 4 convolutional layers of 32 channels followed by 3 fully-connected layers of 1024 hidden neurons, with ReLU activation. For Cover Type, we use a multi-layer perceptron (MLP) of 3 hidden layers with 256 hidden neurons. We also adopt common practices of Adam optimizer (Kingma and Ba, 2015), Dropout (Srivastava et al., 2014), and BatchNorm (Ioffe and Szegedy, 2015). To calculate the optimal transport plan between the source and target, we use the Earth Mover Distance solver from (Flamary et al., 2021). The number of generated intermediate domains is a hyperparameter, and we show the performance for 1,2,3 or 4 generated domains between each pair of consecutive given domains. Following practices (Kumar et al., 2020), in self-training, we filter out the 10% data where the model's prediction is least confident at.

When implementing of GOAT (Algorithm 1), we take the first two conv layers as the encoder $\mathcal{E}$, and treat the layers after them as the classifier h. Sec. 5.4 explains this choice.

Notes on number of generated domain Although Eq. (23) shows the relationship between the optimal number of domains and source-target distance, it is still unclear what exact number should be chosen. To solve the problem, one can use a heuristic hyperparameter tuning approach. Specifically, a subset of the target set with highly confident pseudo-labels can be used as a validation set. Then, with all other components of the algorithm fixed, one can evaluate the performance using different numbers of domains on the target validation set and select the (empirically) optimal number of intermediate domains. However, as subsequent sections will demonstrate, the hyperparameter tuning stage is generally not necessary for the success of GOAT. Instead, our findings indicate that GOAT's performance is robust across varying numbers of domains.

5.3 Empirical Results

Comparison with UDA methods We first empirically validate our claim that the traditional one-off UDA methods do not work well on datasets with large distribution shifts. Here, we compare GDA methods with three popular UDA methods: DANN (Ganin et al., 2016), DeepCoral (Sun and Saenko, 2016) and DeepJDOT (Damodaran et al., 2018). These UDA methods do not have mechanisms to incorporate additional unlabeled data during training, so we use the source and target data as in the conventional UDA framework. In contrast, GDA methods such as GST and our GOAT have the capability to incorporate intermediate domains with unlabeled data. For illustration, we use two given intermediate domains for both GDA algorithms. It is important to note that under this setting, the amount of labeled data used in UDA and GDA is identical. We report the comparison in

Table 2: GDA methods outperform one-off UDA methods on datasets with large distribution shifts.

	Rotated MNIST	Color-Shift MNIST	Portraits	Cover Type
DANN (Ganin et al., 2016)	44.6±2.3	56.5±3.2	73.8±1.5	63.3±1.6
DeepCoral (Sun and Saenko, 2016)	49.6±1.8	63.5±2.1	71.9±1.3	66.8±1.5
DeepJDOT (Damodaran et al., 2018)	51.6±2.1	65.8±2.7	72.5±1.3	67.0±1.2
GST (2 given domains)	61.6±2.1	67.6±4.8	77.0±1.3	66.9±1.4
GOAT (2 given domains)	**70.3± 2.4**	**90.3±1.4**	**79.9±1.2**	**69.8±1.4**

Table 3: Accuracy (%) on Rotated MNIST.

# Given		# Generated Domains of GOAT			
Domains	0 (GST)	1	2	3	4
0	**50.3±0.7**	48.5±2.2	47.2±1.7	48.2±2.7	47.5±2.8
1	56.3±1.9	55.2±2.6	54.6±1.6	**57.1±2.2**	56.2±1.9
2	61.6±2.1	68.0±1.4	67.0±2.2	68.1±2.2	**70.3±2.4**
3	66.3±2.0	74.0±1.1	**74.4±1.8**	73.2±2.0	74.0±2.3
4	75.5±2.0	83.8±2.0	84.0±1.6	**86.4±2.0**	82.7±1.8

Table 4: Accuracy (%) on Color-Shift MNIST.

# Given		# Generated Domains of GOAT			
Domains	0 (GST)	1	2	3	4
0	40.5±5.5	54.4±6.9	63.2±4.1	75.7±3.8	**79.1±3.0**
1	54.2±5.9	74.7±5.3	79.5±2.9	79.3±3.4	**85.3±3.8**
2	67.6±4.8	78.3±3.4	84.8±2.5	89.0±1.5	**90.3±1.4**
3	73.9±7.6	80.9±6.9	87.4±4.2	**90.7±2.3**	**90.4±1.5**
4	77.4±7.2	84.4±4.6	**91.8±1.8**	91.0±1.8	**91.3±1.2**

Table 5: Accuracy (%) on Portraits.

# Given		# Generated Domains of GOAT			
Domains	0 (GST)	1	2	3	4
0	73.3±1.3	**74.0±1.3**	73.5±2.2	73.6±2.5	**74.2±2.5**
1	74.5±1.6	76.4±1.3	75.5±2.6	**76.8±1.5**	74.7±1.7
2	77.0±1.3	77.4±2.1	79.4±2.4	**79.9±1.2**	77.2±0.9
3	80.7±2.3	80.9±1.6	81.8±1.3	**82.3±1.3**	81.3±1.5
4	82.0±1.4	82.8±1.5	**83.6±1.5**	82.4±1.4	81.8±1.6

Table 6: Accuracy (%) on Cover Type.

# Given		# Generated Domains of GOAT			
Domains	0 (GST)	1	2	3	4
0	63.0±2.3	64.2±2.2	65.0±2.4	**66.2±2.1**	**66.5±2.0**
1	65.9±2.1	68.5±2.0	68.4±1.5	**69.1±1.5**	**69.1±1.5**
2	66.9±1.4	68.9±1.6	68.4±2.1	69.3±1.1	**69.8±1.4**
3	66.9±1.3	68.3±1.4	**69.9±1.8**	68.0±1.5	68.8±1.1
4	67.7±1.7	**69.6±2.1**	68.1±2.0	**69.7±1.2**	69.4±2.0

Table 2. The results demonstrate the advantage of the GDA methods over traditional UDA approaches, as GDA methods consistently outperform UDA methods with various types of distribution shifts. GOAT further improves the performance on top of GST, with a detailed discussion in subsequent paragraphs.

Comparison with Gradual Self-Training We empirically compare our proposed GOAT with Gradual Self-Training (GST) (Kumar et al., 2020). The results on Rotated MNIST, Color-Shift MNIST, Portraits and Cover Type are shown in Tables 3 to 6. Each experiment is run 5 times with 95% confidence interval reported. The leftmost column corresponds to the performance of GST only on given intermediate domains, which is equivalent to GOAT without any generated intermediate domain.

In Tables 3 to 6, the rows ("# Given Domains") indicate the number of *given intermediate domains*. The columns ("# Generated domains of GOAT") represent the number of *generated intermediate domains between* ***each pair*** *of consecutive given domains* (e.g., between the source domain and the first ground-truth intermediate domain, or between the i-th and $(i+1)$-th ground-truth intermediate domains). For instance, in the case of "# given domains = 3" and "# generated domains = 3", we have 5 ground-truth domains (source, target and 3 intermediate domains) and $4 \times 3 = 12$ generated domains (since there are 4 pairs of adjacent domains along the sequence of 5 ground-truth domains), leading to 17 domains in total.

Table 7: Comparison with CoVi (Na et al., 2022) on vision datasets.

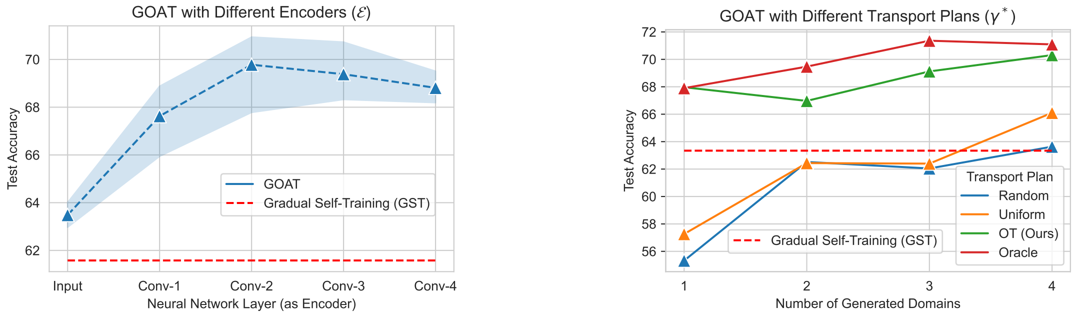
# Given	Rotated MNIST			Color-Shift MNIST			Portraits		
Domains	GST	CoVi	GOAT	GST	CoVi	GOAT	GST	CoVi	GOAT
0	**50.3±0.7**	48.4±2.1	48.5±2.2	40.5±5.5	40.0±5.2	**79.1±3.0**	73.3±1.3	73.7±3.5	**74.2±2.5**
1	56.3±1.9	**57.2±1.8**	**57.1±2.2**	54.2±5.9	59.4±5.7	**85.3±3.8**	74.5±1.6	75.3±1.8	**76.8±1.5**
2	61.6±2.1	64.2±3.4	**70.3±2.4**	67.6±4.8	77.6±7.6	**90.3±1.4**	77.0±1.3	**79.8±3.0**	**79.9±1.2**
3	66.3±2.0	71.4±1.9	**74.4±1.8**	73.9±7.6	86.4±4.7	**90.4±1.5**	80.7±2.3	**82.3±1.4**	**82.3±1.3**
4	75.5±2.0	80.7±3.4	**86.4±2.0**	77.4±7.2	90.9±4.0	**91.3±1.2**	82.0±1.4	83.1±1.9	**83.6±1.5**

Results: i) From the columns of Tables 3 to 6, we can observe that the performance of GOAT monotonically increases with more given intermediate domains, indicating that GOAT indeed benefits from given intermediate domains. ii) From the rows of Tables 3 to 6, we can see that with a fixed number of given domains, our GOAT can consistently outperform Gradual Self-Training (GST). The only exception is the case of Rotated MNIST without any given intermediate domain, which might be due to the challenge illustrated in Fig. 5(a). Overall, the empirical results shown in these tables demonstrate that our GOAT can consistently improve gradual self-training (GST) with generated intermediate domains when only a few given intermediate domains are available.

Comparison on Intermediate Domain Generation In unsupervised domain adaptation (UDA), various algorithms have been developed to *generate intermediate domains* to facilitate adaptation, such as Gong et al. (2019); Na et al. (2021, 2022). Among these, CoVi (Na et al., 2022) stands out for its exceptional (state-of-the-art) performance on UDA benchmarks. CoVi utilizes MixUp (Zhang et al., 2018) to generate synthetic data that are used to adapt models, and it also employs techniques of contrastive learning, entropy maximization and label consensus. In the GDA setting, it is applied in a similar manner as GST, where the adaptation is done sequentially on two adjacent domains, from the source to the target. To ensure a fair comparison, we fix the network structure and training recipe of CoVi to match the implementation of our GOAT, and present the results with 95% confidence intervals over 5 random seeds. Since CoVi is a vision-specific model, we conduct the comparison on the three vision datasets: Rotated MNIST, Color-Shift MNIST and Portraits. The results are reported in Table 7. Our proposed algorithm, GOAT, demonstrates comparable or superior performance to CoVi across all numbers of given domains (0,1,2,3,4). This indicates that our algorithm is indeed powerful at i) generating high-quality intermediate domains useful for gradual domain adaptation and ii) utilizing given (ground-truth) intermediate domains.

5.4 Ablation Studies

Choice of Encoder ($\mathcal{E}$) Here, we study how the choice of the encoder (i.e., feature space) affects the performance of GOAT. Since we use a CNN, we can take each network layer as the feature space. Specifically, we consider the four convolutional layers and input space as candidate choices for the encoder. Once choosing a layer, we take all layers before it (including itself) as the encoder. In this ablation study, we use Rotated MNIST dataset with 2 given intermediate domains, and let GOAT generate 4 intermediate domains between consecutive given domains. From Fig. 6a, we can observe that directly applying GOAT



(a) Choices of the encoder ($\mathcal{E}$).

(b) Choices of the transport plan (γ^*).

Figure 6: Ablation studies on Rotated MNIST with 2 given intermediate domains. (a) *Different neural net layers as encoders for the intermediate domain generation of GOAT.* One can see that input space is not suitable for intermediate domain generation, and the second convolutional layer (Conv-2) is optimal. (b) *Different transport plans for the intermediate domain generation of GOAT.* Obviously, our optimal transport (OT) plan significantly outperforms the baseline transport plans (Random & Uniform), and its performance is even close to the oracle.

in the input space performs significantly worse than the optimal choice, Conv-2 (i.e., the second convolutional layer). This result justifies our use of an encoder for intermediate domain generation (instead of directly generating in the input space). Notably, Fig. 6a shows that deeper layers are not always better, showing a clear increase-then-decrease accuracy curve. Hence, we keep using Conv-2 as the encoder for GOAT in all experiments.

Choice of Transport Plan (γ^*) In our Algorithm 1, the data generated along the Wasserstein geodesic are essentially linear combinations of data from the pair of given domains, with weights (for each combination) assigned by the optimal transport (OT) plan γ^*. To validate that the performance gain of GOAT indeed comes from the Wasserstein geodesic estimation instead of just linear combinations, we conduct an ablation study on GOAT in Rotated MNIST with 2 given intermediate domains. Specifically, we consider four approaches to provide the transport plan γ^*: i) a random transport plan (weights are sampled from a uniform distribution), ii) a uniform transport plan (weights are the same for all combinations), iii) the optimal transport (OT) plan provided by Algorithm 1, iv) the oracle transport plan[10], which is the ground-truth transport plan in this study. For a fair comparison, when constructing the random and uniform plans, we ensure the number of non-zero elements is the same as that of the oracle plan (i.e., keeping the number of generated data the same). See more details in Appendix D.

From Fig 6b, we observe that, in general, the random and uniform plans do not obtain non-trivial performance gain compared with the baseline, the vanilla Gradual Self-Training (GST) without any generated domain. In contrast, our OT plan is significantly better and achieves similar performance as the oracle, demonstrating the high quality of the OT plan and justifying our algorithm design with the Wasserstein geodesic.

[10] The target data of the Rotated MNIST dataset are obtained by rotating training data. Thus there is a one-to-one mapping between source and target data. The oracle plan is built from the one-to-one mapping, i.e., an element γ^*_{ij} is non-zero if and only if x_{0i} is rotated to x_{Tj}.

6. Conclusion

In this work, we study gradual domain adaptation. On the theoretical side, we provide a significantly improved analysis for the generalization error of the gradual self-training algorithm, under a more general setting with relaxed assumptions. In particular, compared with existing results, our bound provides an *exponential* improvement on the dependency of the step size T, as well as a better sample complexity of $O(1/\sqrt{nT})$, as opposed to $O(1/\sqrt{n})$ as in the existing work. Based on the theoretical insight, we propose a novel algorithmic framework, Generative Gradual Domain Adaptation with Optimal Transport (GOAT), which automatically generates intermediate domains along the Wasserstein geodesic (between consecutive given domains) and applies GDA on the generated domains. Empirically, we show that GOAT can significantly outperform vanilla GDA when the given intermediate domains are scarce. Essentially, our GOAT is a promising framework that augments GDA with generated intermediate domains, leading GDA to be applicable to more real-world scenarios.

References

Samira Abnar, Rianne van den Berg, Golnaz Ghiasi, Mostafa Dehghani, Nal Kalchbrenner, and Hanie Sedghi. Gradual domain adaptation in the wild: When intermediate distributions are absent. *arXiv preprint arXiv:2106.06080*, 2021.

Tameem Adel, Han Zhao, and Alexander Wong. Unsupervised domain adaptation with a relaxed covariate shift assumption. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

Takayuki Akiyama, Hirotaka Hachiya, and Masashi Sugiyama. Efficient exploration through active learning for value function approximation in reinforcement learning. *Neural Networks*, 23(5):639–648, 2010.

Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017.

Philip W Anderson. More is different. *Science*, 177(4047):393–396, 1972.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332, 2019.

Jordan T Ash, Robert E Schapire, and Barbara E Engelhardt. Unsupervised domain adaptation using approximate label matching. *arXiv preprint arXiv:1602.04889*, 2016.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19: 137, 2007.

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):151–175, 2010.

Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains, 2018. URL `https://openreview.net/forum?id=BJsBjPJvf`.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51: 22–45, 2015.

Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10835–10845, 2019.

Hong-You Chen and Wei-Lun Chao. Gradual domain adaptation without indexed intermediate domains. *Advances in Neural Information Processing Systems*, 34, 2021.

Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–289. Springer, 2014.

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

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

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 447–463, 2018.

Paul Adrien Maurice Dirac et al. *The principles of quantum mechanics*. Oxford university press, 1930.

Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`.

Aden Forrow, Jan-Christian Hütter, Mor Nitzan, Philippe Rigollet, Geoffrey Schiebinger, and Jonathan Weed. Statistical optimal transport via factored couplings. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2454–2465. PMLR, 2019.

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.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, Crystal Lee, Philipp Krahenbuhl, and Alexei A. Efros. A century of portraits: A visual historical record of american high school yearbooks, 2015. URL `https://arxiv.org/abs/1511.02575`.

Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *CVPR*, pages 2477–2486, 2019.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, pages 15262–15271, 2021.

Xiaolin Huang, Lei Shi, and Johan AK Suykens. Ramp loss linear programming support vector machine. *The Journal of Machine Learning Research*, 15(1):2185–2211, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Leonid V Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1939.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized sliced wasserstein distances. *Advances in neural information processing systems*, 32, 2019.

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.

Vitaly Kuznetsov and Mehryar Mohri. Generalization bounds for time series prediction with non-stationary processes. In *International conference on algorithmic learning theory*, pages 260–274. Springer, 2014.

Vitaly Kuznetsov and Mehryar Mohri. Learning theory and algorithms for forecasting non-stationary time series. In *NIPS*, pages 541–549. Citeseer, 2015.

Vitaly Kuznetsov and Mehryar Mohri. Time series prediction and online learning. In *Conference on Learning Theory*, pages 1190–1213. PMLR, 2016.

Vitaly Kuznetsov and Mehryar Mohri. Generalization bounds for non-stationary mixing processes. *Machine Learning*, 106(1):93–117, 2017.

Vitaly Kuznetsov and Mehryar Mohri. Discrepancy-based theory and algorithms for forecasting non-stationary time series. *Annals of Mathematics and Artificial Intelligence*, 88 (4):367–399, 2020.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 1998. URL `http://yann.lecun.com/exdb/mnist/`.

Bo Li, Yezhen Wang, Shanghang Zhang, Dongsheng Li, Kurt Keutzer, Trevor Darrell, and Han Zhao. Learning invariant representations and risks for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1104–1113, 2021.

Bo Li, Yifei Shen, Yezhen Wang, Wenzhen Zhu, Dongsheng Li, Kurt Keutzer, and Han Zhao. Invariant information bottleneck for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7399–7407, 2022.

Jian Liang, Ran He, Zhenan Sun, and Tieniu Tan. Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation. In *CVPR*, pages 2975–2984, 2019.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, pages 6028–6039, 2020.

Percy Liang. Statistical learning theory, 2016. URL `https://web.stanford.edu/class/cs229t/notes.pdf`.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1094–1103, 2021.

Jaemin Na, Dongyoon Han, Hyung Jin Chang, and Wonjun Hwang. Contrastive vicinal space for unsupervised domain adaptation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 92–110. Springer, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pages 460–467. IEEE, 2009.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Alexander Rakhlin and Karthik Sridharan. Statistical learning and sequential prediction, 2014.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning: Random averages, combinatorial parameters, and learnability. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning via sequential complexities. *J. Mach. Learn. Res.*, 16(1):155–186, 2015.

Ievgen Redko, Nicolas Courty, Rémi Flamary, and Devis Tuia. Optimal transport for multi-source domain adaptation under target shift. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 849–858. PMLR, 2019.

Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. The risks of invariant risk minimization. *arXiv preprint arXiv:2010.05761*, 2020.

Shiori Sagawa, Pang Wei Koh, Tony Lee, Irena Gao, Sang Michael Xie, Kendrick Shen, Ananya Kumar, Weihua Hu, Michihiro Yasunaga, Henrik Marklund, Sara Beery, Etienne David, Ian Stavness, Wei Guo, Jure Leskovec, Kate Saenko, Tatsunori Hashimoto, Sergey Levine, Chelsea Finn, and Percy Liang. Extending the WILDS benchmark for unsupervised adaptation. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. URL `https://openreview.net/forum?id=2EhHKKXMbG0`.

Meyer Scetbon, Marco Cuturi, and Gabriel Peyré. Low-rank sinkhorn factorization. In *International Conference on Machine Learning*, pages 9344–9354. PMLR, 2021.

Meyer Scetbon, Gabriel Peyré, and Marco Cuturi. Linear-time gromov wasserstein distances using low rank couplings and costs. In *International Conference on Machine Learning*, pages 19347–19365. PMLR, 2022.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Masashi Sugiyama. Learning under non-stationarity: Covariate shift adaptation by importance weighting masashi sugiyama. In *Handbook of Computational Statistics: Concepts and Methods*, 2013. URL `https://api.semanticscholar.org/CorpusID:15472933`.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 443–450. Springer, 2016.

Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33:19276–19289, 2020.

Michel Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HkL7n1-0b`.

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

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

Vladimir Vapnik. *The nature of statistical learning theory.* Springer science & business media, 1999.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9898–9907. PMLR, 13–18 Jul 2020.

Haoxiang Wang, Bo Li, and Han Zhao. Understanding gradual domain adaptation: Improved analysis, optimal path and beyond. In *International Conference on Machine Learning*, pages 22784–22801. PMLR, 2022a.

Haoxiang Wang, Haozhe Si, Bo Li, and Han Zhao. Provable domain generalization via invariant-feature subspace recovery. In *International Conference on Machine Learning*, volume 162, pages 23018–23033. PMLR, 2022b. URL `https://proceedings.mlr.press/v162/wang22x.html`.

Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A fine-grained analysis on distribution shift. In *International Conference on Learning Representations*, 2022.

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.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pages 819–827. Pmlr, 2013.

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

Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *Advances in neural information processing systems*, 31:8559–8570, 2018.

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, 2019a.

Han Zhao, Junjie Hu, Zhenyao Zhu, Adam Coates, and Geoffrey J Gordon. Deep generative and discriminative domain adaptation. In *AAMAS*, pages 2315–2317, 2019b.

Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, pages 289–305, 2018.

Yang Zou, Zhiding Yu, Xiaofeng Liu, B.V.K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. In *ICCV*, October 2019.

Appendix A. Proof

A.1 Proof of Lemma 1

Lemma 1 (Error Difference over Shifted Domains) *Consider two arbitrary measures μ, ν over $\mathcal{X} \times \mathcal{Y}$. Then, for arbitrary classifier h and loss function ℓ satisfying Assumption 2, 3, the population loss of h on μ and ν satisfies*

$$|\varepsilon_\mu(h) - \varepsilon_\nu(h)| \leq \rho\sqrt{R^2+1}\; W_p(\mu, \nu) \tag{10}$$

where W_p is the Wasserstein-p distance metric and $p \geq 1$.

Proof The population error difference of h over the two domains (i.e., μ and ν is

$$\begin{aligned} |\varepsilon_\mu(h) - \varepsilon_\nu(h)| &= \left|\mathbb{E}_{x,y\sim\mu}[\ell(h(x),y)] - \mathbb{E}_{x',y'\sim\nu}[\ell(h(x'),y')]\right| \\ &= \left|\int \ell(h(x),y)d\mu - \int \ell(h(x'),y')d\nu\right| \end{aligned} \tag{30}$$

Let γ be an arbitrary coupling of μ and ν, i.e., it is a joint distribution with marginals as μ and ν. Then, (30) can be re-written and bounded as

$$|\varepsilon_\mu(h) - \varepsilon_\nu(h)| = \left|\int \ell(h(x),y)d\mu - \int \ell(h(x'),y')d\gamma\right| \tag{31}$$

$$\text{(triangle inequality)} \leq \int \left|\ell(h(x),y) - \int \ell(h(x'),y')\right| d\gamma \tag{32}$$

$$(\ell \text{ is } \rho\text{-Lipschitz}) \leq \int \rho\left(\|h(x) - h(x')\| + \|y - y'\|\right) d\gamma \tag{33}$$

$$(h \text{ is } R\text{-Lipschitz}) \leq \int \rho R\|x - x'\| + \rho\|y - y'\| d\gamma \tag{34}$$

$$(R > 0) \leq \int \rho\sqrt{R^2+1}\left(\|x - x'\| + \|y - y'\|\right) d\gamma \tag{35}$$

Since γ is an arbitrary coupling, we know that

$$|\varepsilon_\mu(h) - \varepsilon_\nu(h)| \leq \inf_\gamma \int \rho\sqrt{R^2+1}\left(\|x - x'\| + \|y - y'\|\right) d\gamma \tag{36}$$

$$= \rho\sqrt{R^2+1} W_1(\mu, \nu) \tag{37}$$

Since the Wasserstein distance W_p is monotonically increasing for $p \geq 1$, we have the following bound,

$$|\varepsilon_\mu(h) - \varepsilon_\nu(h)| \leq \rho\sqrt{R^2+1} W_1(\mu, \nu) \leq \rho\sqrt{R^2+1} W_p(\mu, \nu) \tag{38}$$

■

A.2 Proof of Proposition 1

Proposition 1 (The stability of the ST algorithm) *Consider two arbitrary measures μ, ν, and denote S as a set of n unlabelled samples i.i.d. drawn from μ. Suppose $h \in \mathcal{H}$ is a pseudo-labeler that provides pseudo-labels for samples in S. Define $\hat{h} \in \mathcal{H}$ as an ERM* *solution fitted to the pseudo-labels,*

$$\hat{h} = \arg\min_{f\in\mathcal{H}} \sum_{x\in S} \ell(f(x), h(x)) \tag{11}$$

Then, for any $\delta \in (0,1)$, the following bound holds true with probability at least $1-\delta$,

$$\left|\varepsilon_\mu(\hat{h}) - \varepsilon_\nu(h)\right| \le \mathcal{O}\left(W_p(\mu,\nu) + \frac{\rho B + \sqrt{\log\frac{1}{\delta}}}{\sqrt{n}}\right) \tag{12}$$

Proof Define $\widehat{\varepsilon}_\mu(h) \coloneqq \frac{1}{|S|}\sum_{x\in S}\ell(h(x),y)$ as the empirical loss over the dataset S, where S consists of samples i.i.d. drawn from $\mu(X)$ and y is the ground truth label of x.

Then, we have the following sequence of inequalities:

$$\begin{aligned}
\text{(Use Lemma A.1 of Kumar et al. (2020))} \quad \varepsilon_\mu(h) &\le \widehat{\varepsilon}_\mu(\hat{h}) + \mathcal{O}\left(R_n(\ell\circ\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{n}}\right) \\
\left(\text{since } h(x) = \hat{h}(x)\ \forall x \in S\right) \quad &= \widehat{\varepsilon}_\mu(h) + \mathcal{O}\left(R_n(\ell\circ\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{n}}\right) \\
\text{(Use Lemma A.1 of Kumar et al. (2020) again)} \quad &\le \varepsilon_\mu(h) + \mathcal{O}\left(2R_n(\ell\circ\mathcal{H}) + 2\sqrt{\frac{\log(1/\delta)}{n}}\right) \\
\text{(By Lemma 1)} \quad &\le \varepsilon_\nu(h) + \rho\sqrt{R^2+1}W_p(\mu,\nu) \\
&\quad + \mathcal{O}\left(R_n(\ell\circ\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{n}}\right) \\
\text{(By Talagrand's lemma with Assumption 3,4)} \quad &\le \varepsilon_\nu(h) + \rho\sqrt{R^2+1}W_p(\mu,\nu) \\
&\quad + \mathcal{O}\left(\frac{\rho B}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}}\right) \\
&\le \varepsilon_\nu(h) + \mathcal{O}\left(W_p(\mu,\nu) + \frac{\rho B}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{n}}\right)
\end{aligned}$$

For the step using Talagrand's lemma (Talagrand, 1995), the proof of Lemma A.1 of Kumar et al. (2020) also involves an identical step, thus we do not replicate the specific details here. ■

A.3 Proof of Lemma 2

Lemma 2 (Discrepancy Bound) *With Lemma 1, the discrepancy measure* (15) *can be upper bounded as*

$$\operatorname{disc}(\mathbf{q}_t) \le \rho\sqrt{R^2+1}\sum_{\tau=0}^{t-1} q_\tau (t-\tau-1)\Delta \tag{16}$$

With $\mathbf{q}_t = \mathbf{q}_t^* = (\frac{1}{t}, ..., \frac{1}{t})$, this upper bound can be minimized as

$$\operatorname{disc}(\mathbf{q}_t^*) \le \rho\sqrt{R^2+1}\ t\Delta/2 = \mathcal{O}(t\Delta) \tag{17}$$

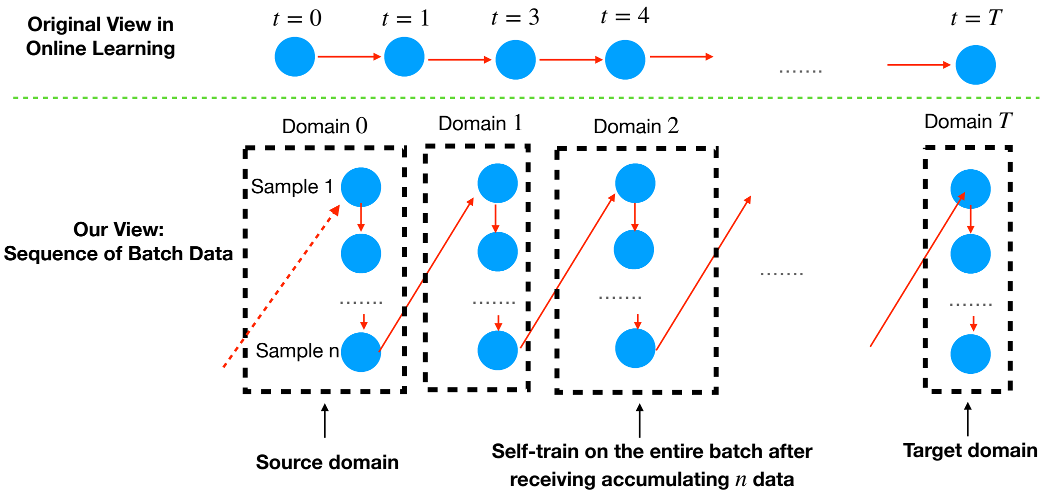

Original View in Online Learning
$t = 0$
$t = 1$
$t = 3$
$t = 4$
....
$t = T$
Our View: Sequence of Batch Data
Domain 0
Domain 1
Domain 2
Domain T
Sample 1
Sample n
....
Source domain
Self-train on the entire batch after receiving accumulating n data
Target domain

Figure 7: Our reductive view of gradual self-training that is helpful to Theorem 1.

Proof Within our setup of gradual self-training,

$$\begin{aligned}\mathrm{disc}(\mathbf{q}_t) &= \sup_{h\in\mathcal{H}}\left(\varepsilon_{t-1}(h) - \sum_{\tau=0}^{t-1} q_\tau \cdot \varepsilon_\tau(h)\right)\\ &= \sup_{h\in\mathcal{H}}\left(\sum_{\tau=0}^{t-1} q_\tau\left(\varepsilon_{t-1}(h) - \varepsilon_\tau(h)\right)\right)\\ &\le \sup_{h\in\mathcal{H}}\left(\sum_{\tau=0}^{t-1} q_\tau|\varepsilon_{t-1}(h) - \varepsilon_\tau(h)|\right)\\ \text{(By Lemma 1)} &\le \rho\sqrt{R^2+1}\sum_{\tau=0}^{t-1} q_\tau \cdot (t-\tau-1)\Delta\end{aligned}$$

With $\mathbf{q}_t = \mathbf{q}_t^* = (\frac{1}{t}, ..., \frac{1}{t})$, this bound becomes

$$\mathrm{disc}(\mathbf{q}_t^*) \le \rho\sqrt{R^2+1}\sum_{\tau=0}^{t-1} q_\tau \cdot (t-\tau-1)\Delta = \rho\sqrt{R^2+1}\ \frac{t}{2}\Delta = \mathcal{O}(t\Delta)$$

and it is trivial to show that this upper bound is smaller than any other $\mathbf{q}_t$ with $\mathbf{q}_t \neq \mathbf{q}_t^*$. ■

A.4 Proof of Theorem 1

Theorem 1 (Generalization Bound for Gradual Self-Training) *For any $\delta \in (0, 1)$, the population loss of gradually self-trained classifier h_T in the target domain is upper bounded* *with probability at least* $1-\delta$ *as*

$$\varepsilon_T(h_T) \le \sum_{t=0}^{T} q_t \varepsilon_t(h_t) + \|\mathbf{q}_{n(T+1)}\|_2 \left(1{+}\mathcal{O}\left(\sqrt{\log(1/\delta)}\right)\right)$$
$$+\mathrm{disc}(\mathbf{q}_{T+1}){+}\mathcal{O}\left(\sqrt{\log T}\mathcal{R}^{\mathrm{seq}}_{n(T+1)}(\ell \circ \mathcal{H})\right) \tag{18}$$

For the class of neural nets considered in Example 2,

$$\varepsilon_T(h_T) \le \varepsilon_0(h_0) + \mathcal{O}\left(T\Delta{+}\frac{T}{\sqrt{n}}{+}T\sqrt{\frac{\log 1/\delta}{n}}{+}\frac{1}{\sqrt{nT}}{+}\sqrt{\frac{(\log nT)^{3L-2}}{nT}}{+}\sqrt{\frac{\log 1/\delta}{nT}}\right) \tag{19}$$

A Reductive View of the Learning Process of Gradual Self-Training If we directly apply Corollary 2 of Kuznetsov and Mohri (2020), we can obtain a generalization bound as

$$\begin{aligned}\varepsilon_{\mu_T}(h) &\le \sum_{t=0}^{T} q_t \varepsilon_{\mu_t}(h) + \mathrm{disc}(\mathbf{q}_{T+1}) + \|\mathbf{q}_{T+1}\|_2 + 6M\sqrt{4\pi \log T}\mathcal{R}^{\mathrm{seq}}_T(\ell \circ \mathcal{H}) \\ &\quad + M\|\mathbf{q}_{T+1}\|_2\sqrt{8\log\frac{1}{\delta}} \\ &\le \sum_{t=0}^{T} q_t \varepsilon_{\mu_t}(h) + O(T\Delta) + \mathcal{O}(\frac{1}{\sqrt{T}}) + 6M\sqrt{4\pi\log T}\mathcal{R}^{\mathrm{seq}}_T(\ell\circ\mathcal{H}) + \mathcal{O}(\sqrt{\frac{\log\frac{1}{\delta}}{T}})\end{aligned} \tag{39}$$

where M is an upper bound on the loss (Lemma 3 proves such a M exists), and the last inequality is obtained by setting $\mathbf{q}_{T+1} = \mathbf{q}^*_{T+1} = (\frac{1}{T+1}, \dots, \frac{1}{T+1})$.

A typical generalization bound involves terms with dependence on N (the training set size), usually in the form $\mathcal{O}(\sqrt{\frac{1}{N}})$, and these terms vanish in the infinite-sample limit (i.e., $N \to \infty$). These terms also appear in standard generalization bounds of unsupervised domain adaptation (Ben-David et al., 2007; Zhao et al., 2019a), where N becomes the number of available unlabelled data in the target domain.

In the case of gradual domain adaptation, the total number of available unlabelled is Tn, and we would expect Tn will appear in a form similar to $\mathcal{O}(\sqrt{\frac{1}{nT}})$, which vanishes in the infinite-sample limit (i.e., $nT \to \infty$). However, the generalization bound (1) has terms $\mathcal{O}(\sqrt{\frac{1}{T}})$ and $\mathcal{O}(\sqrt{\frac{\log\frac{1}{\delta}}{T}})$, which does not vanish even with infinite data per domain, i.e., $n \to \infty$ (certainly results in $Tn \to \infty$).

We attribute this issue to the coarse-grained nature of online learning analyses such as Kuznetsov and Mohri (2016, 2020), which do not take data size per domain into consideration.

To address this issue, we propose a novel reductive view of the entire learning process of gradual self-training, leading to a more fined-grained generalization bound than Eq. (39).

We draw a diagram to illustrate this reductive view in Fig. 7. Specifically, instead of viewing each domain as the smallest element, we zoom in to the sample-level and view each sample as the smallest element of the learning process. We view the gradual self-training algorithm as follows: it has a fixed data buffer of size n, and each newly observed sample is pushed to the buffer; the model updates itself by self-training once the buffer is full; after the update, the buffer is emptied. Notice that this view does not alter the learning process of gradual self-training.

With this reductive view, the learning process of gradual self-training consists of nT smallest elements (i.e., each sample is a smallest element), instead of T elements (i.e., each domain is a smallest element) in the view of online learning works (Kuznetsov and Mohri, 2016, 2020). As a result, terms of order $\mathcal{O}\left(\sqrt{\frac{1}{T}}\right)$ in (39) becomes $\mathcal{O}\left(\sqrt{\frac{1}{nT}}\right)$, and terms of order $\mathcal{O}\left(\frac{T}{n}\right)$ also vanish as $n \to \infty$. Notably, the upper bounds on the terms $\sum_{t=0}^{T} q_t \varepsilon_{\mu_t}(h)$ and $\mathrm{disc}(\mathbf{q}_{T+1})$ in (14) do not become larger with this view, since there is no distribution shift within each domain (e.g., the learning process over the first n samples in Fig. 7 does not involve any distribution shift, and the iteration $n-1 \mapsto n$ incurs a distribution shifts, since the $(n-1)$-th sample is in the first domain while the n-th sample is in the second domain).

With this reductive view, we can finally obtain a tighter generalization bound for gradual self-training without the issues mentioned previously.

Proof With the inductive view introduced above, we can improve the naive bound (39) to

$$\varepsilon_{\mu_T}(h_T) \leq \sum_{t=0}^{T}\sum_{i=0}^{n-1} q_{nt+i}\varepsilon_{\mu_t}(h_T) + \mathrm{disc}(\mathbf{q}_{n(T+1)}) + \|\mathbf{q}_{n(T+1)}\|_2 + 6M\sqrt{4\pi\log nT}\mathcal{R}^{\mathrm{seq}}_{nT}(\ell\circ\mathcal{H}) \tag{40}$$

$$\begin{aligned}
&+ M\|\mathbf{q}_{n(T+1)}\|_2\sqrt{8\log\frac{1}{\delta}} \\
&\leq \frac{1}{T+1}\sum_{t=0}^{T}\varepsilon_{\mu_t}(h_T) + \rho\sqrt{R^2+1}\,\frac{T+1}{2}\Delta + \frac{1}{\sqrt{nT}} + 6M\sqrt{4\pi\log nT}R^{\mathrm{seq}}_{nT}(\ell\circ\mathcal{H}) \\
&+ M\sqrt{\frac{8\log 1/\delta}{nT}} \\
&\leq \varepsilon_{\mu_0}(h_0) + \mathcal{O}\left(T\Delta + T\sqrt{\frac{\log 1/\delta}{n}} + \frac{1}{\sqrt{nT}} + \rho R\sqrt{\frac{(\log nT)^7}{nT}} + \sqrt{\frac{\log 1/\delta}{nT}}\right)
\end{aligned}$$

where $\mathbf{q}_{n(T+1)}$ is taken as $\mathbf{q}_{n(T+1)} = \mathbf{q}^*_{n(T+1)} = (\frac{1}{n(T+1)}), \ldots, \frac{1}{n(T+1)})$. We used the following facts when deriving the inequalities above:

- The first term of (40) has the following bound

$$\begin{aligned}
\sum_{t=0}^{T}\sum_{i=0}^{n-1} q_{nt+i}\varepsilon_{\mu_t}(h_T) &= \frac{1}{T+1}\sum_{t=0}^{T}\varepsilon_{\mu_t}(h_T) \\
&\leq \varepsilon_{\mu_0}(h_0) + \mathcal{O}(T\Delta) + \mathcal{O}\left(\frac{1}{\sqrt{n}} + T\sqrt{\frac{\log 1/\delta}{n}}\right)
\end{aligned} \tag{41}$$

  which is obtained by recursively apply Lemma 1 and Proposition 1 to each term in the summation. For example, the last term in $\sum_{t=0}^{T}\varepsilon_{\mu_t}(h_T)$ can bounded by Proposition 1 as follows

$$\text{(By Proposition 1)} \quad \varepsilon_T(h_T) \le \varepsilon_{\mu_{T-1}}(h_{T-1}) + \mathcal{O}\left(W_p(\mu_T, \mu_{T-1}) + \frac{1}{\sqrt{n}} + \sqrt{\frac{\log 1/\delta}{n}}\right)$$

$$\text{(Same as the above step)} \le \dots$$

$$\le \varepsilon_{\mu_0}(h_0) + \mathcal{O}(T\Delta + \mathcal{O}\left(T\sqrt{\frac{\log 1/\delta}{n}}\right) \tag{42}$$

and the second last term can be bounded similarly with the additional help of Lemma 1

$$\text{(By Lemma 1)} \quad \varepsilon_{T-1}(h_T) \le \varepsilon_{\mu_T}(h_T) + \mathcal{O}(W_p(\mu_T, \mu_{T-1}))$$

$$\text{(Apply Eq. (42))} \le \varepsilon_{\mu_0}(h_0) + T\Delta + \mathcal{O}\left(T\sqrt{\frac{\log 1/\delta}{n}}\right)$$

All the rest terms (i.e., $\varepsilon_{T-2}(h_T), \dots, \varepsilon_0(h_T)$) can be bounded in the same way.

- The second term of (40) can be bounded by applying Lemma 2.
- The value of $R^{\text{seq}}_{nT}(\ell \circ \mathcal{H})$ can be bounded by combining Lemma 4 and Example 2.

■

A.5 Helper Lemmas

Lemma 3 (Bounded Loss) *For any $x \in \mathcal{X}, y \in \mathcal{Y}, h \in \mathcal{H}$, the loss $\ell(x, y)$ is upper bounded by some constant M, i.e., $l(h(x), y) \le M$.*

Proof Notice that i) the input x is bounded in a compact space, specifically, $\|x\|_2 \le 1$ (ensured by Assumption 1), ii) y lives in a compact space in $\mathbb{R}$ (defined in Sec. 2.1), iii) the hypothesis $h \in \mathcal{H}$ is R-Lipschitz, and iv) the loss function ℓ is ρ-Lipschitz.

Combining these conditions, one can easily find that there exists a constant M such that $l(h(x), y)$ for any $x \in \mathcal{X}, y \in \mathcal{Y}, h \in \mathcal{H}$. ■

Lemma 4 (Lemma 14.8 of Rakhlin and Sridharan (2014)) *For ρ-Lipschitz loss function l, the sequential Rademacher complexity of the loss class $\ell \circ \mathcal{H}$ is bounded as*

$$\mathcal{R}^{\text{seq}}_T(\ell \circ \mathcal{H}) \le \mathcal{O}(\rho\sqrt{(\log T)^3})\mathcal{R}^{\text{seq}}_T(\mathcal{H}) \tag{43}$$

Proof See Rakhlin and Sridharan (2014). ■

A.6 Derivation of the Optimal T

In Sec. 3.4, we show a variant of the generalization bound in (21) as

$$\varepsilon_T(h_T) \le \varepsilon_0(h_0) + \inf_{\mathcal{P}} \widetilde{\mathcal{O}}\left(T\Delta_{\max} + \frac{T}{\sqrt{n}} + \sqrt{\frac{1}{nT}}\right) \tag{44}$$

where $\Delta_{\max}$ is an upper bound on the average W_p distance between any pair of consecutive domains along the path, i.e., $\Delta_{\max} \geq \frac{1}{T}\sum_{t=1}^{T} W_p(\mu_{t-1}, \mu_t)$.

Given that $T, \Delta_{\max}, n$ are all positive, we know there exists an optimal $T = T^*$ that minimizes the function

$$f(T) := T\Delta_{\max} + \frac{T}{\sqrt{n}} + \sqrt{\frac{1}{nT}} \ , \tag{45}$$

and one can straightforwardly derive that

$$T^* = \left(\frac{1}{2(1+\Delta_{\max}\sqrt{n}\ )}\right)^{\frac{2}{3}} \ . \tag{46}$$

Proof The derivative of $f(T)$ is

$$f'(T) = \Delta_{\max} + \frac{1}{\sqrt{n}} - \frac{1}{2\sqrt{n}} T^{-\frac{3}{2}} \ , \tag{47}$$

and the second-order derivative of $f(T)$ is

$$f''(T) = \frac{3}{4\sqrt{n}} T^{-\frac{5}{2}} \ . \tag{48}$$

Eq. (48) indicates that $f(T)$ is strictly convex in $T \in (0, \infty)$. Then, we only need to solve for the equation

$$f'(T) = 0 \tag{49}$$

as $T \in (0, \infty)$, which gives our the solution

$$T^* = \left(\frac{1}{2(1+\Delta_{\max}\sqrt{n}\ )}\right)^{\frac{2}{3}} \ . \tag{50}$$

■

Appendix B. Theoretical Arguments

On Proposition 2 The inequality in (25) holds true since the Wasserstein distance metric W_p is known to enjoy the property of triangle inequality. In (25), the equality is obtained as the intermediate domains $\mu_1, \ldots, \mu_{T-1}$ sequentially fall along the Wasserstein geodesic between μ_0 and μ_T, since the geodesic is defined as the shortest path of distributions connecting μ_0 and μ_T under the W_p metric.

On Proposition 3 This linear program (LP) formulation of optimal transport is also called Kantorovich LP in the literature. One can find details and proof of Kantorovich LP in (Peyré et al., 2019).

On the Encoder With a $\rho_{\mathcal{E}}$-Lipschitz continuous encoder $\mathcal{E} : \mathcal{X} \mapsto \mathcal{Z}$ mapping inputs to the feature space $\mathcal{Z}$ (i.e., $z \leftarrow \mathcal{E}(x)$ for any input x), the order of the generation bound (1) stays the same. The reason is as follows: The bound (1) is linear in terms of ρ_h [11], which is the Liphschitz constant of the classifier h; With the encoder $\mathcal{E}$, one can effectively view the whole encoder-classifier model as $f : \mathcal{X} \mapsto \mathcal{Y}$ such that $f(x) = h(\mathcal{E}(x))$; Then, the Liphschitz

[11] The dependence on ρ_h is hidden with the big-O notation in (1)

constant of f is obviously $\rho = \rho_{\mathcal{E}} \rho_h$ since f is a composite function of $h \circ \mathcal{E}$; Finally, replacing h with f in the analysis, one can see that the order of the bound (1) stays the same, with some terms getting multiplied by a factor of $\rho_{\mathcal{E}}$ (i.e., equivalent to replacing the term ρ_h with $\rho = \rho_{\mathcal{E}} \rho_h$ in the bound).

Appendix C. More Details on the Proposed Algorithm

To reduce the $\mathcal{O}(n^3 \log n)$ complexity of the exact OT calculation to $\mathcal{O}(n^2)$, we can solve the entropy-regularized OT problem Cuturi (2013) instead. Consider source data $\{x_{0i}\}_{i=1}^m$ and target data $\{x_{Tj}\}_{i=1}^n$, the entropy-regularized OT plan γ_λ^* under the transport cost function c is obtained by solving

$$\begin{gathered} \gamma_\lambda^* = \underset{\gamma \in \mathbb{R}_{\geq 0}^{m \times n}}{\arg\min} \sum_{i,j} \gamma_{i,j} c(x_{0i}, x_{Tj}) + \lambda \sum_{i,j} \gamma_{i,j} \log \gamma_{i,j}, \\ \text{s.t.} \quad \gamma \mathbf{1}_n = \frac{1}{m} \mathbf{1}_m \quad \text{and} \quad \gamma^T \mathbf{1}_m = \frac{1}{n} \mathbf{1}_n, \end{gathered} \tag{51}$$

where λ is a regularization coefficient. The low computational complexity comes at the cost of a dense optimal transport plan, i.e., γ_λ^* is generally a dense matrix rather than a sparse one[12]. Thus, $\mathcal{O}(mn)$ non-zero entries will be generated in γ_λ^*, and this quadratic space complexity becomes intractable for large datasets. To remedy this issue, we design two methods to zero out insignificant entries in γ_λ^* to reduce the space complexity:

1. **Small-value cutoff.** Although the transport plan γ_λ^* resulted from entropy-regularized OT is dense, most entries still have values close to 0. Those entries of tiny magnitude can be zeroed out without having a noticeable impact on the final results.

2. **Confidence cutoff.** Consider the one-hot encoded matrix of source labels $Y_0 \in \{0, 1\}^{m \times \#\text{class}}$ and the entropy-regularized OT plan γ_λ^*. The logits of target prediction by optimal label transport is
$$\widehat{Y}_T = \gamma_\lambda^{*\mathrm{T}} Y_0. \tag{52}$$
Then, we can calculate a confidence score for each target prediction by the logits. Using a certain confidence threshold, the target samples that the transport plan is unconfident with can be filtered out, making the transport plan more sparse.

With proper choices of cutoff values, those methods can reduce the space complexity from $\mathcal{O}(mn)$ to $\mathcal{O}(m + n)$ without noticeable compromise on the final performance.

C.1 Number of Intermediate Domains

The number of intermediate domains can be considered as a hyperparameter. The theory shows that there exists an optimal number T^* in terms of self-training performance in the GDA setting:

$$T^* = \max \left\{ \frac{L}{\Delta}, \tilde{\mathcal{O}} \left(\left(\frac{1}{1 + \Delta \sqrt{n}} \right)^{2/3} \right) \right\}, \tag{53}$$

[12] As we discussed in Sec. 4.2, γ^* has at most $n + m - 1$ non-zero entries, thus it is a sparse matrix.

where L is the W_p distance between the source and target and Δ is the average W_p distance between any pair of consecutive domains.

Although Eq. (53) shows the relationship between the optimal number of domains and source-target distance, it is still unclear what exact number should be chosen. To solve the problem, we use a heuristic hyperparameter tuning approach. Specifically, we use a subset of the target set with highly confident pseudo-labels as a validation set. Then, with all other components of the algorithm fixed, we evaluate the performance using different numbers of domains on the target validation set and select the (empirically) optimal number of intermediate domains.

Appendix D. More Details on Experiments

Network Implementation. For the 4-layer CNN encoder used in experiments on Rotated MNIST and Portraits, we use convolutional layers with kernel size 3 and SAME padding. During self-training, we train on each domain for 10 epochs. Empirically, we verify that regularization techniques are important for the success of gradual self-training, including using dropout layers and early stopping.

For the VAE used to produce Fig. 4, we use 4 convolutional layers with kernel size 3 and max-pooling, followed by a fully-connected layer with 128 neurons as the encoder. For the decoder, we use four deconvolutional layers with kernel size 3 (Kingma and Welling, 2014). We use ReLU activation for the layers. The encoder and decoder are jointly trained on data from source and target in an unsupervised manner with the Adam optimizer (Kingma and Ba, 2015) (learning rate as 10^{-4} and batch size as 512).

Encoder Pretraining. We pretrain an encoder on the given domains. During pretraining, we use a 3-layer MLP on top of the encoder and perform self-training on the given domains. Specifically, we first fit the model on the source domain, then iteratively use the model to pseudo-label the next domain and self-train on it. After pretraining, the MLP is discarded and the encoder is fixed to provide features for the downstream tasks.

OT ablation. When designing different plans, we make sure that the number of non-zero entries is equal so that in the domains generated by those plans, the amount of data is the same. For the random plan, we first initialize a zero matrix, then sample the same amount of entries as the ground-truth plan in the matrix, and fill in a weight value between 0 to 1 uniformly at random. For the uniform plan, we use the same procedure except that we fill in the same weight for each sampled entry. In the end, we normalize the matrix.