

© 2024 Jun Wu

TRUSTWORTHY TRANSFER LEARNING

BY

JUN WU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Doctoral Committee:

Professor Jingrui He, Chair
Professor Jiawei Han
Assistant Professor Han Zhao
Dr. Haixun Wang, Instacart

ABSTRACT

Deep transfer learning aims to improve the generalization performance of a learning algorithm on the target domain, by discovering and transferring latent knowledge from relevant source domains. This paradigm is motivated by the human intelligence that people have an inherent ability to apply knowledge already learned from one task to another related one. Despite the exploding interest in deep transfer learning techniques, several fundamental problems in understanding deep transfer learning remain unsolved.

This thesis investigates two fundamental research questions for understanding deep transfer learning. (Q1) *Knowledge Transferability*: How is the source knowledge transferred to the target domain with performance guarantees? The key challenge behind knowledge transferability is the data heterogeneity between source and target domains. Notably, samples within a domain can be either independent and identically distributed (IID) or non-independent and identically distributed (non-IID) in real scenarios. This leads to various underlying factors contributing to data heterogeneity across domains, such as feature distribution shifts in cross-domain image recognition, and graph structure shifts in cross-network mining. (Q2) *Knowledge Trustworthiness*: Can the transferred source knowledge be reliable and trustworthy for learning the prediction function on the target domain? Specifically, multiple trustworthy concerns exist in the transfer learning process, such as adversarial robustness, privacy, fairness, transparency, etc. Having both research questions in mind, we formulate the problem of *trustworthy transfer learning*. The goal is to learn a reliable and trustworthy prediction function on the target domain using relevant knowledge from source domains.

In this thesis, we systematically develop a suite of theoretical tools and algorithms, in order to understand deep transfer learning from the perspectives of knowledge transferability and trustworthiness. Concisely, we answer the first question (Q1) by theoretically deriving the generalization performance of deep transfer learning and empirically validating its efficacy across various IID and non-IID learning tasks. Furthermore, to answer the second question (Q2), this thesis delves into various trustworthy properties of deep transfer learning, including demonstrating the adversarial robustness of deep transfer learning techniques, developing privacy-preserving transfer learning approaches, and enhancing performance fairness among participant domains. Finally, we present several promising research directions, including (i) unifying knowledge transferability across different data modalities; (ii) uncovering the correlations between data distribution shifts and model transparency, and (iii) unveiling the trade-off between knowledge transferability and trustworthiness.

ACKNOWLEDGMENTS

First and foremost, I am deeply grateful to my advisor, Dr. Jingrui He, for her unwavering support, guidance, patience, and encouragement throughout my Ph.D. journey. Her steadfast encouragement and understanding have been invaluable, especially during my most challenging times. During the COVID-19 pandemic, noticing my struggles with both my research and personal life, Dr. He consistently encouraged me not to give up, reminding me that success was just one step further. Dr. He's dedication and mentorship have not only shaped my research but also supported me personally. Without her continued support, I could hardly have imagined completing my Ph.D. program and starting my career in academia. Furthermore, Dr. He serves as an enduring role model who inspires me to pursue excellence as a researcher, mentor, and instructor in the future.

I would also like to extend my sincere gratitude to other members of my thesis committee, Dr. Jiawei Han, Dr. Haixun Wang, and Dr. Han Zhao, for their invaluable comments and constructive suggestions on my thesis. Notably, Dr. Han provides insightful guidance for my research. Bridging the gap between theoretical analysis and practical algorithms is vital in real-world applications. It is critical to identify the conditions under which the developed algorithms perform well and determine how to leverage information effectively for model analysis. Besides, Dr. Wang's expertise in data mining has inspired me to rethink research topics such as federated learning and graph learning, leading us to collaborate on impactful cutting-edge research. Furthermore, Dr. Zhao has shared a lot of detailed comments that have greatly supported my research in the area of transfer learning. His extensive expertise in this field has provided me with an in-depth understanding of transfer learning under various assumptions, thus enhancing the depth of my studies.

I deeply appreciate the help and support from all the members of iSAIL lab and IDEA lab, including Dawei Zhou, Yao Zhou, Arun Reddy Nelakurthi, Pei Yang, Xu Liu, Lecheng Zheng, Xue Hu, Dongqi Fu, Yikun Ban, Yunzhe Qi, Haonan Wang, Ziwei Wu, Wenxuan Bao, Tianxin Wei, Xinrui He, Issac Raheem Joy, and Zihao Li from iSAIL lab; Liangyue Li, Chen Chen, Si Zhang, Boxin Du, Jian Kang, Qinghai Zhou, Zhe Xu, Lihui Liu, Baoyu Jing, Yuchen Yan, Shengyu Feng, Yuheng Zhang, Derek Wang, Yian Wang, Zhichen Zeng, Ishika Agarwal, Eunice Chan, Xinyu He, Blaine Hill, Zhining Liu, Ruizhong Qiu, Hyunsik Yoo, Alex Zheng, David Zhou, Ruike Zhu, Xiao Lin, Yaojing Wang, Yu Wang, Ziyue Zhu, Hao Wang, Haoran Li, Shweta Jain, Tianwen Chen, and Yunyong Ko from IDEA lab. I would also like to express special thanks to Dr. Hanghang Tong, whose whose rigorous research

and optimistic attitude foster an inclusive environment that enhances both my research and personal life.

In addition, I would like to thank all my collaborators from academia and industry: Elizabeth A. Ainsworth, Hanghang Tong, Dr. Haohan Wang, Andrew Leakey, Kaiyu Guan, Sheng Wang, Wenxuan Bao, and Issac Raheem Joy from UIUC; Yongming Liu from ASU; Fatemeh Vafaei from UNSW Sydney; Haixun Wang from Instacart; and Dr. Jiejun Xu from HRL Laboratories. Their valuable suggestions and feedback have significantly enriched the breadth and depth of my Ph.D. work.

Last but not least, I would like to express my deepest gratitude to my parents, Hanping Wu and Xingli Xiao, for their unconditional love, support, and encouragement throughout this journey. Their belief in me has been a constant source of motivation, thereby making this achievement possible. Thank you for always being there for me. This accomplishment is as much yours as it is mine.

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	Background and Motivation	1
1.2	Key Challenges	3
1.3	Research Overview	7
1.4	Organization	10
CHAPTER 2	LITERATURE REVIEW	11
2.1	Knowledge Transferability	11
2.2	Knowledge Trustworthiness	13
CHAPTER 3	IID KNOWLEDGE TRANSFERABILITY	17
3.1	Static Transfer Learning with Single Source	18
3.2	Static Transfer Learning with Multiple Sources	33
3.3	Dynamic Transfer Learning with Closed-Set Targets	50
3.4	Dynamic Transfer Learning with Open-Set Targets	59
CHAPTER 4	NON-IID KNOWLEDGE TRANSFERABILITY	74
4.1	Graph Transfer Learning with Homogeneous Structure	75
4.2	Graph Transfer Learning with Heterogeneous Structure	85
CHAPTER 5	ADVERSARIAL ROBUSTNESS OF KNOWLEDGE TRANSFER . .	102
5.1	Adversarial Vulnerability of Knowledge Transfer	103
5.2	Adversarially Robust Knowledge Transfer	114
CHAPTER 6	PRIVACY AND FAIRNESS OF KNOWLEDGE TRANSFER	129
6.1	Soft Clustering for Privacy-Preserving Knowledge Transfer	130
6.2	Hard Clustering for Privacy-Preserving Knowledge Transfer	145
CHAPTER 7	CONCLUSION AND FUTURE RESEARCH	153
7.1	Conclusion	153
7.2	Vision for the Future	154
REFERENCES		157

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Artificial Intelligence (AI) creates computer systems that can replicate human behavior and mimic human intelligence to perform tasks [1]. As explained by Stanford Professor John McCarthy, AI is “the science and engineering of making intelligent machines”. It is noted that previously learned knowledge enables individuals to acquire new skills with remarkable efficiency. This intelligence has motivated a transfer learning paradigm [2], which refers to the transfer of knowledge or information from source domain to target domain. In recent years, by instantiating the learning models with modern neural networks [3, 4], a deep transfer learning paradigm has been introduced with enhanced transferability capabilities [5, 6, 7]. This paradigm has achieved promising performance in a myriad of high-impact applications, ranging from object recognition [6, 8, 9, 10], semantic classification [11, 12, 13], to social network analysis [14, 15], from agriculture [16, 17, 18], bioinformatics [19, 20], healthcare [21], to e-commerce [22, 23, 24].

In contrast to standard machine learning involving samples from a single domain, transfer learning focuses on modeling heterogeneous data from different domains. The distribution shifts across domains can result in a model trained in one domain performing poorly in another [25, 26]. Thus, the intuition behind transfer learning techniques is to bridge the gap between source and target domains by discovering and transferring their shared knowledge. Compared to learning from the target domain alone, the transferred knowledge could significantly improve the prediction performance on the target domain, especially when the target domain has limited or no labeled data. Following [2, 5], we provide a formal definition of (deep) transfer learning as follows.

Definition 1.1 (Transfer Learning [2]). Given a source domain and a target domain, transfer learning aims to improve the prediction function on the target domain using the knowledge from the source domains.

It is termed as deep transfer learning [5] when the learning algorithm is instantiated with a deep neural network. As explained in [2], transfer learning is a general term to describe the transfer of knowledge or information from source to target domains. With different data and model assumptions, transfer learning leads to various specific problem settings, e.g., hypothesis transfer learning [27, 28], domain adaptation [18, 29, 30, 31], out-of-distribution generalization (a.k.a. domain generalization) [32, 33, 34, 35], knowledge distillation [36, 37],

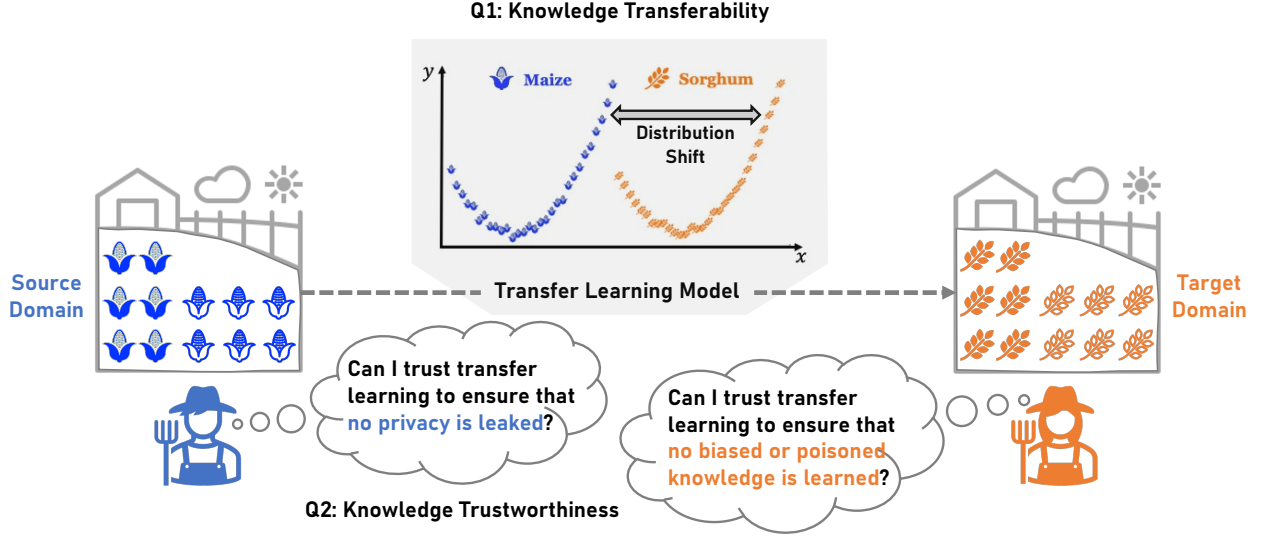


Figure 1.1: A motivating example of trustworthy transfer learning

etc. For example, domain adaptation and out-of-distribution generalization focus on data-level knowledge transfer, as source domains are associated with sufficient training samples. In this case, though the target domain has no labeled data, the domain-invariant knowledge transferred from the source domains can be applied to the target domain under mild data assumptions (i.e., source and target domains share the same input and output spaces) [29, 38]. In contrast, hypothesis transfer learning and knowledge distillation aim to investigate the model-level knowledge transfer, as only source hypotheses pre-trained on source domains can be utilized [27].

Despite the exploding interest in deep transfer learning, there remain several unsolved fundamental problems in understanding deep transfer learning, including *how to transfer* the knowledge across domains under realistic assumptions (e.g., dynamics [8, 39, 40], non-IIDness [14, 24, 41], open-world domain gap [10, 42, 43], etc.), and *whether to trust* the transferred knowledge under a variety of trustworthy concerns (e.g., adversarial robustness [44], privacy [45], fairness [46], transparency [6], etc.). Diving into these research problems would shed light on the underlying principles of understanding deep transfer learning. Therefore, in this thesis, we generalize standard transfer learning [2] to trustworthy transfer learning [47].

Definition 1.2 (Trustworthy Transfer Learning). Given a source domain and a target domain, the goal of trustworthy transfer learning is to learn a reliable and trustworthy prediction function on the target domain using the knowledge from the source domains.

Figure 1.1 provides a motivating example of trustworthy transfer learning in agriculture

data analysis¹. In this example, a target farmer aims to train a model over the collected sorghum data. The task is to predict the biochemical traits (e.g., Nitrogen content, Chlorophyll, etc.) of sorghum samples using the leaf hyperspectral reflectance [16, 48]. Nevertheless, it is expensive and time-consuming to collect the labeled training samples. A feasible solution is to leverage knowledge from a relevant maize data set collected by a source farmer. By handling the distribution shift between maize and sorghum domains, transfer learning leads to improved performance with theoretical guarantees under mild conditions. Notably, this transfer learning process might involve several trustworthy concerns from source and target farmers. To name a few, will the privacy of source data be leaked in transfer learning? How does the poisoned and biased source knowledge negatively affect the prediction performance on the target domain? What is the fundamental trade-off between transfer performance and trustworthy properties?

1.2 KEY CHALLENGES

The overarching focus of this thesis is trustworthy transfer learning with both theoretical foundations and rich methodology. Having this in mind, there are two key challenges behind trustworthy transfer learning.

C1: Data Heterogeneity. The source and target domains have different data distributions. For example, in visual object recognition [49], the distribution shifts between the sketch image domain and the real-world image domain are induced by variations in image styles and backgrounds. In agriculture [14], different plant species (e.g., maize and sorghum in Figure 1.1) lead to distribution shifts regarding the leaf hyperspectral reflectance features. More specifically, there are various types of distribution shifts, including covariate shift, label shift, concept shift, etc. Let $\mathbb{P}(X, Y)$ denote the joint distribution over input features X and output values Y . Covariate shift assumes that the source and target domains share the same conditional distribution $\mathbb{P}(Y|X)$, but have different marginal distribution $\mathbb{P}(X)$ [50]. Label shift (a.k.a. target shift) holds that the marginal $\mathbb{P}(Y)$ changes, while the conditional $\mathbb{P}(X|Y)$ is shared across domains [51]. In contrast, concept shift assumes that the marginal $\mathbb{P}(X)$ is fixed while the conditional $\mathbb{P}(Y|X)$ varies [52].

Most previous transfer learning works [31, 38, 53] assume that the samples in each domain are independent and identically distributed (IID). This hinders their applications to learning tasks involving non-IID data, such as cross-domain recommender systems in e-commerce [23, 41] and cross-network role identification in transportation [24, 54]. Hence this thesis studies

¹<https://aifarms.illinois.edu/research/>

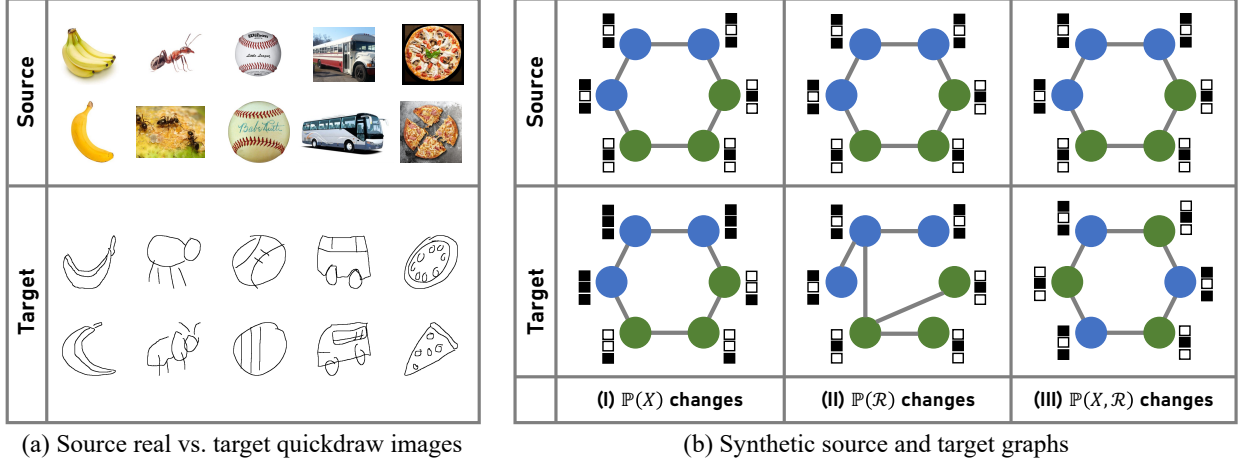


Figure 1.2: Data heterogeneity in image and graph domains. (a) Source real images and target quickdraw image domains have varied styles and backgrounds, leading to the distribution shift across domains [49]. (b) The source and target graphs have different joint distributions $\mathbb{P}(X, \mathcal{R})$ over the node attributes X and the graph structures $\mathcal{R}$, e.g., (I) with varied $\mathbb{P}(X)$, (II) with varied $\mathbb{P}(\mathcal{R})$, and (III) with varied $\mathbb{P}(X, \mathcal{R})$ while $\mathbb{P}(X)$ and $\mathbb{P}(\mathcal{R})$ are fixed.

a more general and realistic distribution shift assumption for transfer learning. We let $\mathbb{P}(X, \mathcal{R}, Y)$ denote the joint distribution over input features X , output values Y , and sample dependency $\mathcal{R}$, and assume $\mathcal{R} = \emptyset$ if samples are IID. In the following, we take cross-network mining [41, 54, 55] as an example². In this case, $\mathcal{R}$ denotes the complex graph structure indicating the node connections within a graph. In the following, we focus on the generalized covariate shift assumption, where the source and target domains share the same conditional distribution $\mathbb{P}(Y|X, \mathcal{R})$, while the marginal distribution $\mathbb{P}(X, \mathcal{R})$ varies across domains. Based on this assumption, distribution shifts across domains can be induced by changes in sample features, sample dependencies, or both.

- *C1-1: Sample Feature Distribution Shifts.* Standard covariate shift assumption holds that the marginal feature distribution $\mathbb{P}(X)$ varies between source and target domains [50]. It is commonly used in IID learning tasks, e.g., object recognition [31, 49] (see Figure 1.2(a)), document classification [12], etc. Based on this assumption, this thesis explores the knowledge transferability of transfer learning in both static [13, 18] and dynamic [8, 10] scenarios. Concisely, static transfer learning holds that the source and target domains have static data distributions, while the data distribution of the source or target domain can change over time for dynamic transfer learning [40].

²In this thesis, we use “graph” and “network” interchangeably to represent the graph-structured data in every domain.

- *C1-2: Sample Dependency Shifts.* As shown in Figure 1.2(b), in the context of cross-network mining, the distribution shift between source and target graphs can be induced by node attributes X and graph structure $\mathcal{R}$ [24]. Here, Figure 1.2(b) provides an intuitive example to explain why a graph learning model trained on a source graph may perform poorly on a target graph. In the scenario (III) of Figure 1.2(b), the source and target graphs have identical marginal distributions $\mathbb{P}(X)$ and $\mathbb{P}(\mathcal{R})$, but different joint distributions $\mathbb{P}(X, \mathcal{R})$. Specifically, the source graph follows the homophily assumption that nodes within the same class tend to be connected. With this assumption, graph neural networks (GNNs) [56, 57, 58] can be designed to learn node representations by recursively aggregating information for the first-order neighborhood. In this case, GNNs enforce each node to more likely aggregate information from its neighbors within the same class. Then, the trained GNNs suffer when applied to the target graph under the heterophily assumption [59] where connected nodes may have different classes and dissimilar node attributes. This is because the trained GNNs might lead to noisy and uninformative node representations in the target graph by aggregating messages from nearby neighbors belonging to different classes.

C2: Trustworthy Concerns. The generalization performance of transfer learning algorithms have been studied over the past decades [7, 14, 18, 27, 29, 31, 38, 53]. Complementary to generalization performance, it is crucial to understand the trustworthiness of the transferred knowledge in the transfer learning process, especially for decision making in high-stakes applications [60]. It is explained [61] that “trust is the relationship between a trustor and a trustee: the trustor trusts the trustee”. We see that in the context of transfer learning, the trustor can be the owners/users/regulators of either source or target domain. The trustee can be the transfer learning model itself, or more specifically, the knowledge transferred from the source domain to the target domain. As summarized in [60, 61, 62, 63], there are multiple trustworthy aspects that encourage the “trustor” to trust the “trustee” in real scenarios. In this thesis, we focus on the following trustworthy concerns of transfer learning systems (see Figure 1.3).

- *C2-1: Adversarial Robustness.* Adversarial robustness refers to the stability of transfer learning models in the presence of adversarial perturbation. It has been shown [64, 65] that modern neural network models can be easily fooled by the adversarial samples that are perceptibly indistinguishable with respect to the clean input samples. This motivates us to rethink transfer learning from the perspective of adversarial robustness. An open question is whether one can manipulate the prediction performance on the target domain, by injecting adversarial noise into the source domain. This is even more

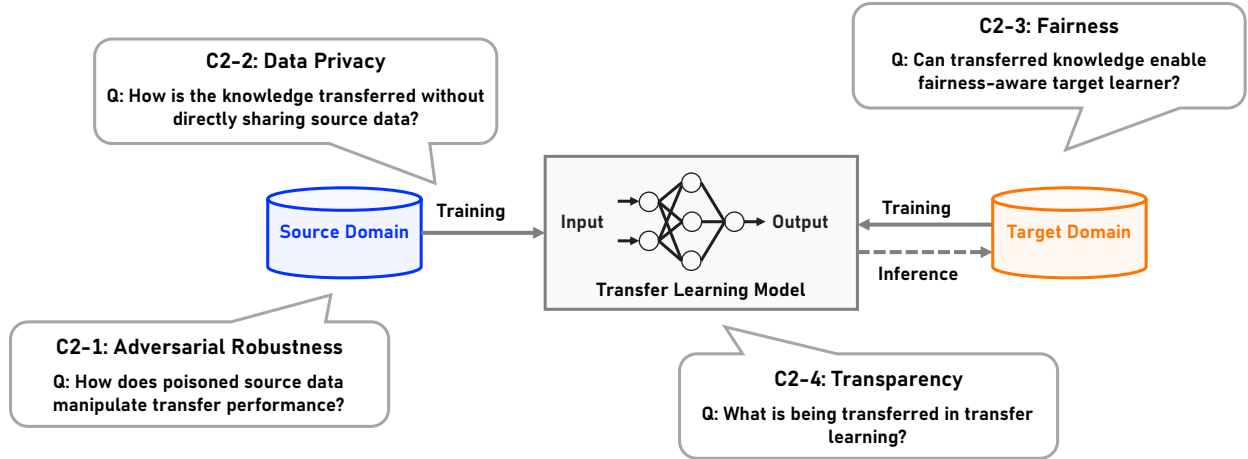


Figure 1.3: Illustration of trustworthy concerns in transfer learning

alarming given recent findings showing that publicly available source data [66, 67, 68] and pre-trained source models [69, 70, 71, 72] have been widely used in a variety of user-specific target tasks. Thus, it is important to investigate how the poisoned knowledge on the source domain might affect the prediction performance on the target domain.

- *C2-2: Privacy.* Privacy is of paramount importance in the privacy-sensitive applications, such as patient clinical data analytics [73] and mobile keyboard prediction [74]. Privacy in transfer learning indicates how the private source data can be protected, i.e., preventing the source data from being leaked in the knowledge transfer process. Then, a privacy-preserving transfer learning framework enables the transfer of knowledge from a private source domain to a user-specific target domain.
- *C2-3: Fairness.* It is observed [75, 76, 77] that machine learning models might generate unfair outcomes, thus leading to unexpected social implications. Generally, unfairness refers to the discriminatory results of a machine learning model, such as bias towards gender, ethnicity, or underrepresented people [78, 79]. Mathematically, there are various definitions of algorithmic fairness, e.g., individual fairness [75], group fairness [76], counterfactual fairness [80], performance fairness [81], etc. In this thesis, we focus on understanding the performance fairness of transfer learning. Specifically, when multiple domains collaborate to train machine learning models, all involved domains can benefit from the transferred knowledge.
- *C2-4: Transparency.* Transparency involves disclosing information of machine learning models [61]. For example, to make humans easily understand and trust the models, it requires to explain the prediction behaviors [82] and quantify the uncertainty [83] of

machine learning models. We notice that in the context of transfer learning, there remains a gap in understanding what knowledge is being transferred and how to quantify the prediction uncertainty on the target domain using the transferred source knowledge.

In summary, the trustworthiness of transfer learning might involve the following questions. When considering the owners of the source domain as the “trustor”, do they trust that the transferred knowledge will not leak their data privacy (*C2-2: Privacy*)? Conversely, if the “trustor” indicates the owners of the target domain, do they trust that the transferred knowledge is not poisoned (*C2-1: Adversarial Robustness*) or biased (*C2-3: Fairness*), and how well can the transferred knowledge be explained (*C2-4: Transparency*)? In this thesis, these questions are unified as “knowledge trustworthiness” in the context of transfer learning.

1.3 RESEARCH OVERVIEW

Having the aforementioned challenges in mind, we have systematically developed a suite of theoretical tools and algorithms to understand deep transfer learning. Notably, my thesis work has two major components: knowledge transferability and knowledge trustworthiness. In practice, my research works have been successfully applied to a variety of high-impact domains, including agriculture [16, 18, 84], e-commerce [24, 85], and cybersecurity [86].

1.3.1 Knowledge Transferability

As discussed in Subsection 1.2, one of the key challenges in transfer learning is the distribution shifts between source and target domains. These shifts can be caused by various factors in different learning tasks. To this end, my research quantitatively measures and enhances knowledge transferability across domains for IID and non-IID learning tasks.

IID Transferability. A common assumption in machine learning is that samples within a domain are independent and identically distributed (IID). When source and target samples are independently drawn from different data distributions, transfer learning techniques are introduced to bridge the domain gap. My research focuses on both *static transfer learning* scenarios [13, 18] where the source and target domains have static data distributions, and *dynamic transfer learning* scenarios [8, 10, 39, 87] where the data distribution of the source or target domain is evolving over time.

Previous efforts on static transfer learning focus on assessing knowledge transferability in the latent feature space learned by deep neural networks [53, 88, 89]. They might suffer from

(i) a two-stage estimation of distribution shifts, which requires the heuristically pre-defined feature layers in deep neural networks, and (ii) unclear convergence during model training. To solve these problems, my research work [18] develops a novel distribution-informed neural network (**DINO**) for building the distribution-aware relationship of inputs and outputs from different domains. The connection between **DINO** and transferable Gaussian processes [90] is theoretically analyzed. Furthermore, **DINO** benefits from (i) unified estimation of knowledge transferability via maximum mean discrepancy in the RKHS induced by a neural tangent kernel [91], and (ii) feasible convergence and generalization error analysis.

In the context of dynamic transfer learning with time-evolving source and target distributions, my works [8, 87] formulate the meta-pairs of tasks from any consecutive time stamps, and then learn the commonly shared knowledge in a meta-learning framework. It is shown that the developed frameworks have the following properties: (i) effective knowledge transferability across dynamic tasks; (ii) fast adaptation to the new target task; (iii) mitigation of catastrophic forgetting on historical target tasks; and (iv) flexibility in incorporating any existing static transfer learning algorithms.

Non-IID Transferability. Non-IID transfer learning involves a relaxed assumption that the samples within each domain can be non-independent and identically distributed (non-IID), e.g., connected nodes in graphs [92, 93], co-occurring words in texts [94], temporal observations in time series [95], etc. In this thesis, we focus on non-IID transfer learning with graph data. Compared to standard transfer learning in IID tasks [9, 31, 88], it is much more challenging to empirically estimate the distribution shifts between source and target graphs due to the complex graph structure.

Though graph neural networks (GNNs) have achieved promising performance in single-domain graph learning tasks [96, 97, 98], it is unclear how GNNs are transferable across graphs. My prior work [99] explores the connection between GNNs and Weisfeiler-Lehman graph isomorphism test [100, 101] in understanding the graph structure. This motivates us to develop a graph subtree discrepancy for measuring the distribution shift across graphs [24], when the source and target graphs follow the homophily assumption (i.e., nodes within the same class tend to be connected). Then based on this graph subtree discrepancy, we derive generalization error bounds for cross-network transfer learning. By minimizing the error bounds, a graph adaptive network (**GRADE**) framework is proposed for cross-network node classification and cross-domain recommendation tasks. Furthermore, we study a more realistic yet challenging graph transfer learning problem [14], where the source and target graphs have different structure assumptions (e.g., from a homophilic source graph to a heterophilic target graph). To solve this problem, a graph-structured Gaussian process framework (**GraphGP**) is proposed. The generalization analysis of **GraphGP** explicitly investigates

the connection between knowledge transferability and graph domain similarity.

1.3.2 Knowledge Trustworthiness

In addition to understanding knowledge transferability, this thesis delves into the impact of trustworthy properties on deep transfer learning. Specifically, we focus on investigating adversarial robustness, data privacy, and performance fairness in transfer learning frameworks, while leaving other trustworthy properties (e.g., transparency) as future research.

- *Adversarial Robustness.* My research work [44] is among the first to demonstrate the adversarial vulnerability of existing deep transfer learning techniques. This is motivated by the observation [102, 103] that feature-based marginal distribution matching cannot guarantee the success of deep transfer learning. The crucial idea of the proposed **I2Attack** framework [44] is to maximize the label-informed joint distribution discrepancy between the raw source domain and the poisoned source domain with the following constraints. (1) *Perceptibly Unnoticeable*: All the poisoned input images are natural-looking. (2) *Adversarially Indirect*: Only source samples are maliciously manipulated. (3) *Algorithmically Invisible*: Neither source classification error nor marginal domain discrepancy between source and target domains increases. These constraints imply that in the context of transfer learning, an adversary could potentially manipulate the source data to gain control over the prediction function on the target domain. Furthermore, we delve into the adversarial robustness of knowledge transfer from the perspective of bias-variance decomposition [104]. It is observed that the generalization error of a new sample can be triggered by both bias and variance of the models trained from different training sets. Then we introduce a novel **Fed_BVA** algorithm by iteratively generating the bias-variance-guided adversarial samples and updating the model parameters.
- *Privacy & Fairness.* Sharing raw source data for knowledge transfer becomes infeasible due to privacy concerns. To solve this problem, a federated learning paradigm [105, 106] has been proposed in recent years. This paradigm allows multiple clients/domains to collaborate in training machine learning models under the coordination of a central server. The raw data in each client is stored locally and not transferred during model training. However, we observe [45, 107] that some clients might suffer from negative transfer. In other words, some clients have worse performance than their local training counterparts (i.e., the client trains the model over its own local data without communication across clients via the server). This phenomenon is formulated as

the unfairness of model performance among clients. To enforce performance fairness and protect client privacy, we propose a soft clustering approach named **FEDORA** [45] and a hard clustering approach named **FEDCOLLAB** [107] for knowledge transfer among relevant clients. The intuition behind **FEDORA** is that each client borrows knowledge from other clients to train the model without leaking client privacy. Specifically, there are two crucial components in **FEDORA**, including (i) adaptive parameter propagation where one client adaptively propagates its parameters to others based on their distribution similarity, and (ii) selective regularization where one client regularizes its local personalized model with received parameters, only when those parameters have a positive impact on the client learner. Furthermore, inspired by clustered federated learning [108, 109], **FEDCOLLAB** is proposed to optimize the collaboration structure such that a client will only share knowledge with other clients if they belong to the same cluster. Data privacy and performance fairness can then be enhanced using the optimized collaboration structure. Notably, it is confirmed [45, 107] that both domain discrepancy among clients and clients’ data quantity affect the optimal collaboration structure.

1.4 ORGANIZATION

The remainder of the thesis is organized as follows. Chapter 2 reviews the related literature on trustworthy transfer learning. Then we introduce IID knowledge transferability in Chapter 3 and non-IID knowledge transferability in Chapter 4. Next, we present our works on understanding knowledge trustworthiness in the context of transfer learning, including adversarial robustness in Chapter 5 and data privacy & performance fairness in Chapter 6. Finally, we conclude this thesis and summarize our future research in Chapter 7.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we present a systematic literature review on trustworthy transfer learning from the perspective of knowledge transferability and trustworthiness. Concisely, we summarize recent theories and algorithms for understanding knowledge transferability from two aspects: (1) IID transferability: the samples within each domain are independent and identically distributed (e.g., individual images), and (2) non-IID transferability: The samples within each domain can be interdependent (e.g., connected nodes within a graph). In addition to knowledge transferability, we review the impact of trustworthiness on deep transfer learning, e.g., whether the transferred knowledge is adversarially robust or algorithmically fair, how to transfer the knowledge under privacy-preserving constraints, etc.

2.1 KNOWLEDGE TRANSFERABILITY

Knowledge transferability studies how knowledge can be transferred from source domains with adequate label information to a relevant target domain with little or no label information. More precisely, it theoretically reveals the connections between the transfer accuracy on the target domain and the relatedness between the source and target domains.

2.1.1 IID Transferability

When the samples within each domain are independent and identically distributed (IID) in the context of transfer learning, knowledge transferability is commonly assessed using the following properties: distribution discrepancy, task diversity, and transferability estimation.

Assuming that the source and target domains share the same input and output spaces (i.e., domain adaptation [2]), it is theoretically shown [38, 110] that the generalization error of a learning algorithm on the target domain can be bounded in terms of the source prediction errors and the distribution discrepancy across domains. In this case, the relatedness of the source and target domains is characterized by the statistical discrepancy measures, e.g., $\mathcal{H}\Delta\mathcal{H}$ -divergence [29, 30], Discrepancy Distance [111, 112], Generalized Discrepancy [113, 114], Maximum Mean Discrepancy [9, 115, 116], Wasserstein Distance [117, 118], Margin Disparity Discrepancy [53], Kullback-Leibler (KL) Divergence [119], f -divergence [31], Cauchy-Schwarz Divergence [120], etc. In addition to statistical discrepancy measures, domain adversarial learning [88, 121, 122, 123, 124] empirically minimizes the distribution shifts across domains using generative adversarial networks (GANs) [125], thereby resulting in domain-

invariant representations.

Task diversity [7, 126, 127, 128, 129] is another tool to theoretically understand the generalization performance of transfer learning algorithms. It enables a relaxed data assumption that the source and target domains can have different output spaces. Assuming that there exists a generic nonlinear feature representation function that is shared across all domains, a common transfer learning approach holds [130] that each domain is associated with a shared representation learning function and a domain-specific prediction function. In [7], task diversity is formally defined to characterize how the worst-case representation difference can be controlled when the task-averaged representation difference is small. Here, the worst-case representation difference is the distance between two representation functions with the worst-case domain-specific prediction function, while the task-averaged representation difference indicates the distance between two representation functions over all the training domains. Based on the task diversity, the excess risk of transfer learning can be upper bounded [7, 127].

Transferability estimation refers to quantitatively measuring how effectively the knowledge can be transferred from a pre-trained source model to a target domain [131, 132, 133, 134, 135]. It is defined over the correlation between the source model and the target data without re-training or fine-tuning the model. Specifically, LEEP (Log Expected Empirical Prediction) [136] estimates the average loglikelihood of target samples using the dummy label distributions generated from the pre-trained source model. It uses only a single forward pass of the source model through the target data. Nevertheless, LEEP can not handle unsupervised pre-trained models and regression tasks. To solve these problems, LogME (Logarithm of Maximum Evidence) [137] is proposed by leveraging the only feature extractor of a pre-trained source model. The crucial idea of LogME is to capture the input-output relationship of target data using the features learned by a source model and a linear transformation. Furthermore, TransRate [138] exploits the coding rate to estimate the transferability scores of any intermediate layer within the pre-trained source model.

2.1.2 Non-IID Transferability

The IID assumption is often violated in real scenarios, e.g., connected nodes in graphs [139], word occurrence in texts [94, 140], temporal observations in time series [141], etc. To bridge this gap, non-IID transferability explores the connections between transfer performance and distribution shifts across domains, considering a relaxed assumption that samples within each domain can be interdependent.

Graphs capture the rich structural relations among data. Graph neural networks (GNNs) [56, 96, 98] have been introduced to encode the nodes within a graph into low-dimensional vec-

tor representations. Recently, the transferability of GNNs across graphs has been studied in node classification/regression tasks from the following aspects: transferability assessment and enhancement. Transferability assessment of GNNs aims to understand whether GNNs trained on a source graph can perform well on a relevant target graph. It is proven [41, 142] that spectral GNNs [97] can be transferable across graphs when the source and target graphs are drawn from the same graphon. Besides, assuming that the k -hop ego-graphs are independent and identically drawn, recent work [54] analyzes the transferability of a well-designed GNN model based on the local structural differences across graphs. In contrast, transferability enhancement of GNNs focuses on designing advanced training strategies of GNNs over both source and target graphs. The key idea is to explicitly mitigate the distribution shifts across graphs using discrepancy minimization [143, 144, 145] or reweighing [146], thereby improving the generalization performance of GNNs.

In addition to graphs, text mining, and time-series analysis may suffer from the distribution shifts induced by non-IID samples. For example, neural machine translation aims to translate a sentence from multiple source languages into a sentence in the target language [147, 148]. It exploits modern neural networks to learn language-invariant word embeddings from source and target languages [149]. However, recent work [150] points out the potential limitations of this language-invariant representation learning framework for multi-source neural machine translation. That is, the translation error can be lower bounded by the sentence distribution difference from different languages. More recently, due to the great success of large language models (LLMs) [72, 151, 152], parameter-efficient transfer learning [153] has been introduced to adapt LLMs to downstream tasks by updating as few parameters as possible. Similarly, cross-domain time-series forecasting [95, 141, 154] and classification [155, 156, 157] have been studied by learning the domain-invariant time-series representations via modern neural networks (e.g., transformers [158]). Although these studies have demonstrated promising performance in various transfer learning tasks, theoretical understanding of knowledge transferability in text and time-series analysis remains limited.

2.2 KNOWLEDGE TRUSTWORTHINESS

The trustworthiness of machine learning models has gained significant attention in various high-stake application domains, such as autonomous vehicles [159], medical diagnosis [21], and financial risk management [160]. This thesis focuses on studying the knowledge trustworthiness in the context of transfer learning. Compared to standard trustworthy machine learning, this thesis discusses whether the transferred knowledge can be trusted by the source and target users, whereas trustworthy machine learning investigates how a user can trust a

model trained on their own data.

2.2.1 Adversarial Robustness

It has been observed [64, 65] that machine learning models can be easily fooled by adversarial examples that are perceptibly indistinguishable with respect to clean inputs. Assuming that training and test data follow the same distribution, the adversarial vulnerability of machine learning models has been explored by carefully designing poisoning and evasion attacks [161]. In particular, poisoning attacks manipulate the training process by injecting carefully crafted examples, with the goals of either reducing the overall predictive performance of a learning algorithm [162] or controlling the model behavior on some specific test examples without degrading the overall predictive performance [163]. In contrast, evasion attacks fool a trained model by adjusting malicious samples during the testing phase [65, 164].

Many efforts have been devoted to understanding the adversarial vulnerability of transfer learning models [165, 166, 167, 168]. Specifically, to fool a trained transfer learning model, evasion attacks generate adversarial samples in the target domain, by either manipulating the prediction loss [166, 167] or minimizing the feature representation dissimilarity between adversarial and clean samples from different classes [165]. In addition, poisoning attacks [168] have demonstrated the vulnerability of transfer learning during training, assuming that the source and target domains share identical input and output spaces and the target domain is only associated with unlabeled data (i.e., unsupervised domain adaptation [2]). Later on, to improve the adversarial robustness of transfer learning models, it is shown [169, 170, 171, 172] that the robustness of an adversarially trained source model can be transferred to the target domain.

Furthermore, the connection between knowledge transferability and adversarial robustness has been studied. It is empirically demonstrated [173, 174] that adversarially robust models can transfer better than their standard-trained counterparts. This improved transferability is explained by the improved feature representation learning [175] or the regularization effect of adversarial training [176].

2.2.2 Privacy

Privacy is another concern when transferring knowledge from a source domain with private data to a target domain. To protect the privacy-sensitive data of local clients, privacy-preserving federated learning is proposed [105, 106]. It empowers multiple clients to collaborate in training machine learning models under the coordination of a central server. From

the perspective of transfer learning, the crucial idea behind federated learning is to transfer knowledge from diverse source clients to a target client. Traditional federated learning aims to learn a single global model from all clients [81, 177, 178, 179], assuming that all the client data are independent and identically distributed (IID). However, it is found [180, 181] that under data heterogeneity among clients, a single global model may not work well for all clients. This motivates the paradigm of personalized federated learning [182] where a personalized model is learned for each client.

In recent years, various personalized federated learning frameworks have been proposed, including parameter decoupling [183, 184, 185], model interpolation [186], clustering [108, 109, 187], multi-task learning [188], meta-learning [189], etc. From the perspective of knowledge transferability, these frameworks mainly follow two parameter sharing mechanisms. One is to use a global model to encode common knowledge shared by all clients, and then regularize the personalized model of each client with this global model [186, 190, 191]. The other one is to capture complex relations among individual clients, and the relation would guide the parameter sharing among clients [192, 193, 194, 195]. However, most existing algorithms focus on improving the overall performance of federated learning systems. Little effort has been devoted to studying the negative transfer [26] in the context of personalized federated learning.

2.2.3 Fairness

Fairness [60] involves eliminating discrimination when training machine learning models. In the legal domain, potential discrimination is defined as disparate treatment (triggered by intentionally treating an individual differently) and disparate impact (triggered by negatively affecting members of a protected group) [78]. Motivated by this definition, different measures of algorithmic fairness have been proposed in machine learning communities, e.g., individual fairness [75], group fairness [76], counterfactual fairness [80], performance fairness [81], etc. To be specific, individual fairness [75, 196] maintains that similar individuals should be treated similarly irrespective of sensitive attributes. Group fairness [77, 197] ensures statistical parity among different protected groups. Counterfactual fairness [80] aims to remove the causal effects between sensitive attributes and model performance. Performance fairness [81, 190] is introduced in personalized federated learning to measure the variance of model performance among different devices/clients.

Previous research [46, 198, 199, 200, 201, 202, 203] has studied the impact of algorithmic fairness on transfer learning. A fundamental question is whether a model fairly trained in the source domain will also be fair in the target domain. To answer this question, [198] reveals

the connection between individual fairness and knowledge transferability. This work shows that individual fairness regularization can improve the transferability of machine learning models, and existing domain adaptation techniques can lead to individual fairness. Besides, assuming the target domain contains a few labeled samples with sensitive attributes, [200] provides the generalization bounds of group fairness in the target domain in terms of fairness-aware distribution discrepancy between source and target domains. When the target domain has only unlabeled samples with sensitive attributes, [46] proposes a fair consistency regularization to ensure group fairness in the target domain. Furthermore, [203] studies a more general learning scenario where the target domain is associated with unlabeled samples with missing sensitive attributes. It develops a fairness-guided sample reweighting approach to mitigate the distribution shifts across domains.

2.2.4 Transparency

Transparency helps non-experts understand the decision-making process of a machine learning model and the confidence level of the model in making decisions [61]. For example, interpretability and explainability have recently been studied to enhance transparency, by designing a simpler and more interpretable model [82, 204] or providing post hoc explanations for existing black-box models [205]. As a complementary metric of transparency, uncertainty quantification [206] illustrates the prediction confidence of a trained model.

There are two major questions behind transparent transfer learning: what is being transferred in transfer learning, and how to quantify the uncertainty of transfer learning models. To answer the first question, prior studies [4, 6] empirically show that in image classification tasks, the success of transfer learning can be attributed to the feature reuse (e.g., Gabor and color blob features learned by the low layers of deep neural networks) and the low-level data statistics. [207] further discusses the connection between fine-tuned neural layers and the type of distribution shifts. To answer the second question, prior work [83] demonstrates that the quality of uncertainty consistently degrades with increasing distribution shifts across domains. Later on, [208] leverages importance weighting to improve calibration under distribution shifts between source and target domains. [209] proposes a post-hoc calibration approach with lower bias and variance of the estimated calibration errors to improve the uncertainty quantification under covariate shift assumption.

CHAPTER 3: IID KNOWLEDGE TRANSFERABILITY

This chapter introduces our work on transfer learning scenarios where the samples in each domain are independent and identically distributed (IID). More specifically, we study the knowledge transferability of transfer learning in both static and dynamic scenarios. Static transfer learning holds that the source and target domains have static data distributions, whereas the data distribution of the source or target domain is evolving over time for dynamic transfer learning. To be specific, Section 3.1 focuses on static transfer learning with a single source domain [18], while Section 3.2 extends static transfer learning to scenarios involving multiple source domains [13]. Furthermore, this thesis explores two learning scenarios in dynamic transfer learning: closed-set targets and open-set targets. Particularly, Section 3.3 studies dynamic transfer learning with closed-set targets indicating that the target domain shares the same group of class labels as the source domain at any time stamp [8]. In contrast, dynamic transfer learning with open-set targets in Section 3.4 assumes that novel unknown classes may appear in the target domain [10].

Notation. The main notation used in this chapter is summarized in Table 3.1. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input feature space and output label space, respectively. Note that if the source and target domains have different output spaces (see Section 3.4), we let $\mathcal{Y}^s$ and $\mathcal{Y}^t$ denote the source output space and target source output space, respectively. Generally, we denote $\mathcal{L}(\cdot, \cdot) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ by the loss function and $d(\cdot, \cdot)$ by the distribution discrepancy across domains and $\mathcal{H}$ be a hypothesis space. For any $h \in \mathcal{H}$, the expected prediction error is defined as $\mathcal{E}(h) = \mathbb{E}_{(x,y) \sim \mathbb{P}}[\mathcal{L}(h(x), y)]$ for any $h \in \mathcal{H}$, and its empirical estimate is given by $\hat{\mathcal{E}}(h) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(h(x_i), y_i)$ given n samples.

Following [32], we assume that the joint probability distribution $\mathbb{P}$ of any domain is drawn from a probability distribution space $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$. For static transfer learning, we denote $\mathcal{D}^s, \mathcal{D}^t$ to be the source and target domains associated with probability distributions $\mathbb{P}^s, \mathbb{P}^t \in \mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$, respectively. Let $\mathbb{P}_X^s, \mathbb{P}_X^t$ denote the marginal distributions of the source and target domains over $\mathcal{X}$, respectively. There are n^s labeled samples $\{(x_i^s, y_i^s)\}_{i=1}^{n^s}$ from the source domain and n^t unlabeled samples $\{x_j^t\}_{j=1}^{n^t}$ from the target domain. We also assume that the target domain might have some labeled samples $\{(x_i^t, y_i^t)\}_{i=1}^{m^t}$ where $0 \leq m^t \ll n^t$. When there are multiple source domains, we use $\mathcal{D}_k^s$ to denote the domain k ($k = 1, 2, \dots$) associated with n_k^s labeled samples $\{x_{ik}^s\}_{i=1}^{n_k^s}$ drawn from $\mathbb{P}_k^s$. For dynamic transfer learning, we denote $\mathcal{D}_k^s, \mathcal{D}_k^t$ to be the source and target domains³ at the k^{th} time stamp associated with probability distributions $\mathbb{P}_k^s, \mathbb{P}_k^t$ over $\mathcal{X} \times \mathcal{Y}$, respectively. At the k^{th} time stamp, there are n_k^s

³With a slight abuse of notation, we use $\mathcal{D}_k^s$ to denote (i) the k^{th} source domain for static transfer learning with multiple sources, and (ii) the source domain k^{th} time stamp for dynamic transfer learning.

Notation	Definition
$\mathcal{X}, \mathcal{Y}$	Input space, output space
$\mathcal{D}^s, \mathcal{D}^t$	Source and target domains
$\mathbb{P}^s, \mathbb{P}^t$	Source and target joint distributions over $\mathcal{X} \times \mathcal{Y}$
$\mathbb{P}_X^s, \mathbb{P}_X^t$	Source and target marginal distributions over $\mathcal{X}$
$\{x_i^s, y_i^s\}_{i=1}^{n^s}$	n^s labeled source samples
$\{x_j^t, y_j^t\}_{j=1}^{m^t}$	m^t labeled target samples
$\{x_j^t\}_{j=1}^{n^t}$	n^t unlabeled target samples ($n^t \gg m^t$)
$\mathcal{D}_k^s$	The k^{th} source domain or the source domain at the k^{th} time stamp
$\mathcal{D}_k^t$	The k^{th} target domain or the target domain at the k^{th} time stamp
$\{x_{ik}^s, y_{ik}^s\}_{i=1}^{n_k^s}$	n_k^s labeled source samples within $\mathcal{D}_k^s$
$\{x_{jk}^t\}_{j=1}^{n_k^t}$	n_k^t unlabeled target samples within $\mathcal{D}_k^t$
$\mathcal{L}(\cdot, \cdot)$	Loss function
$\mathcal{E}(\cdot), \hat{\mathcal{E}}(\cdot)$	Expected prediction error and empirical prediction error
$f(\cdot; \theta)$	Prediction function with parameters θ
$\mathcal{K}_{\text{NNGP}}(\cdot, \cdot)$	NNGP kernel
$\tilde{\mathcal{K}}_{\text{NNGP}}(\cdot, \cdot)$	Distribution-informed NNGP kernel
$\mathcal{K}_{\text{NTK}}(\cdot, \cdot)$	Neural tangent kernel
$\tilde{\mathcal{K}}_{\text{NTK}}(\cdot, \cdot)$	Distribution-informed neural tangent kernel
$\mathcal{K}_{\mathcal{D} \mathcal{X}}(\cdot, \cdot)$	Distribution kernel
$\mathbb{I}$	Identity matrix

Table 3.1: Notation used in Chapter 3

labeled samples $\{(x_{ik}^s, y_{ik}^s)\}_{i=1}^{n_k^s}$ from the source domain and n_k^t unlabeled samples $\{x_{jk}^t\}_{j=1}^{n_k^t}$ from the target domain.

3.1 STATIC TRANSFER LEARNING WITH SINGLE SOURCE

In this section, we study the knowledge transfer from a static source domain with adequate label information to a relevant static target domain with little label information [18].

3.1.1 Problem Definition

Following [2, 38], we formally define the problem of static transfer learning with a single source domain as follows.

Problem 3.1. Given a static source domain $\mathcal{D}^s$ with labeled samples $\{(x_i^s, y_i^s)\}_{i=1}^{n^s} \sim \mathbb{P}^s$ and a static target domain $\mathcal{D}^t$ with unlabeled samples $\{x_j^t\}_{j=1}^{n^t} \sim \mathbb{P}_X^t$ and a few labeled samples

$\{(x_j^t, y_j^t)\}_{j=1}^{m^t} \sim \mathbb{P}^t$ ($m^t \ll n^s$), *static transfer learning* aims to learn a prediction function for the target domain $\mathcal{D}^t$ by leveraging the knowledge from the source domain.

We consider three common assumptions in this problem setting [18]. First, the source and target domains have the same input and output spaces. Second, the samples in each domain are independent and identically distributed (IID). Third, the data distribution of the source/target domain is static. Thus, the major challenge behind static transfer learning lies in the distribution shifts across domains.

Notably, our work [18] focuses on the regression tasks in the static transfer learning setting. It has been revealed in prior work [89] that there is a gap between transfer learning regression and transfer learning classification problems. One commonly used loss function of regression-based transfer learning approaches is mean square error (MSE), whereas classification-based approaches [9, 88] often use cross-entropy loss with softmax. Specifically, softmax changes the feature scales (i.e., Frobenius norm of feature matrix $\|H\|_F$ where H is the hidden feature learned by the deep neural network), and the change of feature scales might lead to the degradation of transfer accuracy [89].

3.1.2 Distribution-Informed Neural Networks

Without loss of generality, we consider an L -layer fully-connected neural network $f(\cdot; \theta)$, which maps the input $x \in \mathcal{X} \subset \mathbb{R}^{d_{in}}$ into the output $y \in \mathcal{Y} \subset \mathbb{R}$. Here θ denotes all the trainable parameters of $f(\cdot; \theta)$ and d_{in} denotes the dimensionality of input data. Nevertheless, it is noticed that a standard neural network $f(\cdot; \theta)$ might not effectively generalize the input-output relationship across different domains. In the following, we provide a motivation example to explain this observation. Figure 3.1(a) shows that in the context of a single domain, e.g., x_1, x_2 are drawn from the same domain, x_1 and x_2 have very similar ground-truth output values if $x_2 \approx x_3$. This input-output relationship within a single domain can be captured by a standard neural network. But in the context of multiple domains shown in Figure 3.1(b), e.g., x_2 is drawn from a source domain while x_3 is from a target domain, x_2 and x_3 might have significantly different ground-truth output values when $x_2 \approx x_3$. This indicates that using $f(\cdot; \theta) : \mathcal{X} \rightarrow \mathcal{Y}$, the input-output relationship learned from the source domain may not generalize well to the target domain. This is because the samples x_2 and x_3 are associated with different domain information, i.e., different sampling distributions. This observation thus motivates us to design a novel distribution-informed neural network (DINO) [18], by studying the distribution-aware input-output relationship from $\mathcal{X} \times \mathcal{P}$ to $\mathcal{Y}$ in different domains.

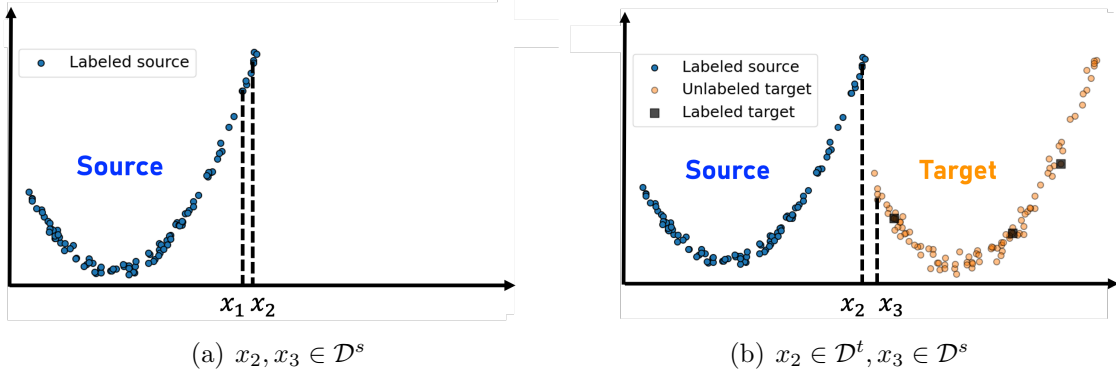


Figure 3.1: A motivating example to compare standard neural networks $f(\cdot)$ with the proposed distribution-informed neural networks $\tilde{f}(\cdot)$. (a) A single domain with two samples $x_1, x_2 \sim \mathbb{P}^s$. A standard neural network holds that $f(x_1) \approx f(x_2)$ if $x_1 \approx x_2$. (b) Two different domains with $x_2 \sim \mathbb{P}^s, x_3 \sim \mathbb{P}^t$. The learned standard neural network learned from the source domain may not generalize well in the target domain, i.e., $f(x_2) \approx f(x_3)$ while $y_2 \neq y_3$ if $x_2 \approx x_3$. In contrast, distribution-informed neural network holds that $\tilde{f}(x_2, \mathbb{P}^s) \neq \tilde{f}(x_3, \mathbb{P}^t)$ if $x_2 \approx x_3$ and $\mathbb{P}^t \neq \mathbb{P}^s$.

Generally, by taking both a sample $x \in \mathcal{X}$ and its associated probability $\mathbb{P} \in \mathcal{P}$ as random variables [32], Bayes' theorem tells us that $\Pr(x, \mathbb{P}) = \Pr(x) \cdot \Pr(\mathbb{P}|x)$ where $\Pr(\cdot)$ denotes the probability of an observed event. The event involving x can be described by observing its representation $f(x)$; given x , the event involving $\mathbb{P}$ corresponds to the input-oriented distribution representation $\Phi_x(\mathbb{P})$. Inspired by this observation, we define a novel distribution-informed neural network (DINO) by learning both the input representation and the input-oriented distribution representation.

$$\tilde{f}(x, \mathbb{P}) \equiv f(x; \theta) \cdot g(\mathbb{P}|x; w_g) = (\phi_{\theta < L}(x)^T w) \cdot (\Phi_x(\mathbb{P})^T w_g) = w^T (\phi_{\theta < L}(x) \Phi_x(\mathbb{P})^T) w_g \quad (3.1)$$

In this case, the first part $f(x; \theta) = \phi_{\theta < L}(x)^T w$ is a standard fully-connected neural network, where $\theta < L$ is the vector of parameters in the first $L - 1$ layers and w is the parameter of the output layer (we assume that the output layer has no bias). It learns the input representation (i.e., $\phi_{\theta < L}(x)$) from the input x . The second part $g_{w_g}(\mathbb{P}|x) = \Phi_x(\mathbb{P})^T w_g$ aims to explicitly learn the input-oriented distribution representation (i.e., $\Phi_x(\mathbb{P})$), where w_g is the parameter. Here, given finite basis sample $\tilde{x}_1, \dots, \tilde{x}_n \sim \mathbb{P}$, the input-oriented distribution representation $\Phi_x(\mathbb{P})$ can be formally defined as follows.

$$\Phi_x(\mathbb{P}) = \sum_{i=1}^n \beta_{x, \tilde{x}_i} \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}} \quad (3.2)$$

where $\beta_{x, \tilde{x}_i} \in \mathbb{R}$ is the coefficient of $\Phi_x(\mathbb{P})$ in the space spanned by $\langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}}$ and indicates the similarity of input sample x and basis sample $\tilde{x}_i$. Here $\langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}}$ denotes the feature map of $\tilde{x}_i$ in the NNGP kernel space [210] induced by infinitely-wide $f(\cdot; \theta)$. Therefore, $\Phi_x(\mathbb{P})$ represents the distribution representation of $\mathbb{P}$ when a sample x is observed in the domain associated with sampling distribution $\mathbb{P}$.

3.1.3 Instantiated Algorithms

By analyzing the distribution-informed neural network (DINO) at initialization and under gradient descent training, our work introduces two instantiated transfer learning algorithms DINO-INIT and DINO-TRAIN for regression tasks [18].

DINO at Initialization. Inspired by the connections between neural networks and Gaussian processes [210, 211, 212, 213], we show that the proposed distribution-informed neural network $\tilde{f}(x, \mathbb{P})$ in Eq. (3.1) is equivalent to a transferable Gaussian process.

Lemma 3.1. Assume that all the parameters of distribution-informed neural network $\tilde{f}(\cdot)$ are initialized as standard normal variables, when the network width goes to infinity, the output function of $\tilde{f}(x)$ at initialization is a Gaussian process, i.e., $\tilde{f}(\cdot) \sim \mathcal{N}(0, \tilde{\mathcal{K}}_{\text{NNGP}})$ with

$$\tilde{\mathcal{K}}_{\text{NNGP}}((x, \mathbb{P}), (x', \mathbb{P}')) = \mathcal{K}_{\text{NNGP}}(x, x') \cdot \mathcal{K}_{\mathcal{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') \quad (3.3)$$

where $\mathcal{K}_{\text{NNGP}}(\cdot, \cdot)$ is the NNGP kernel induced by the neural network $f(\cdot; \theta)$ over the input space $\mathcal{X}$, and $\mathcal{K}_{\mathcal{P}|\mathcal{X}}(\cdot, \cdot)$ is a distribution kernel identifying the similarity of two input-oriented distributions $\mathbb{P}$ and $\mathbb{P}'$ over distribution space $\mathcal{P}$, i.e.,

$$\mathcal{K}_{\mathcal{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = \sum_{i=1}^n \sum_{j=1}^{n'} \beta_{x, \tilde{x}_i} \beta_{x', \tilde{x}'_j} \mathcal{K}_{\text{NNGP}}(\tilde{x}_i, \tilde{x}'_j) \quad (3.4)$$

where n (n') is the number of basis samples sampled from the distribution $\mathbb{P}$ ($\mathbb{P}'$).

Proof. It has been shown [210] that the output function of a fully-connected neural network $f(\cdot; \theta)$ is an iid centered Gaussian process with zero mean and variance $\mathcal{K}_{\text{NNGP}}(x, x')$ (i.e., NNGP kernel). Using definition of distribution-informed neural network $\tilde{f}(x, \mathbb{P})$ in Eq (3.1), we have $\mathbb{E}[\tilde{f}(x, \mathbb{P})] = 0$ (due to $\mathbb{E}[w] = 0$), and $\tilde{\mathcal{K}}_{\text{NNGP}}((x, \mathbb{P}), (x', \mathbb{P}')) = \mathbb{E}[\phi(x)^T \phi(x')] \cdot \Phi_x(\mathbb{P})^T \Phi_{x'}(\mathbb{P}')$ (due to $\mathbb{E}[w^T w] = 1$). Using the definition of $\Phi_x(\mathbb{P})$ in Eq. (3.2), we have $\mathcal{K}_{\mathcal{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = \sum_{i=1}^n \sum_{j=1}^{n'} \beta_{x, \tilde{x}_i} \beta_{x', \tilde{x}'_j} \mathcal{K}_{\text{NNGP}}(\tilde{x}_i, \tilde{x}'_j)$. QED.

This lemma motivates us to propose a transferable Gaussian process regression algorithm named DINO-INIT. The crucial idea is to define the prior Gaussian process $\tilde{f} \sim \mathcal{GP}(0, \tilde{\mathcal{K}}_{\text{NNGP}})$

with the transfer kernel $\tilde{\mathcal{K}}_{\text{NNGP}}$. Following [214], we consider a noisy model $y_i^r = \tilde{f}(x_i^r, \mathbb{P}^r) + \epsilon_i^r$ ($r \in \{s, t\}$), where the noise ϵ_i^r follows a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma_r^2)$. Then the mean $\bar{\mu}(x_*^t)$ and covariance $\bar{\Sigma}(x_*^t, x_*^t)$ of the posterior distribution on a testing target sample x_*^t can be exactly expressed as follows.

$$\begin{aligned}\bar{\mu}(x_*^t) &= \tilde{\mathcal{K}}_{\text{NNGP}}(x_*^t, \mathbf{X}) \mathbf{C}^{-1} \mathbf{y} \\ \bar{\Sigma}(x_*^t, x_*^t) &= \tilde{\mathcal{K}}_{\text{NNGP}}(x_*^t, x_*^t) - \tilde{\mathcal{K}}_{\text{NNGP}}(x_*^t, \mathbf{X}) \mathbf{C}^{-1} \tilde{\mathcal{K}}_{\text{NNGP}}(x_*^t, \mathbf{X})^T\end{aligned}\tag{3.5}$$

where $\mathbf{C} = \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}, \mathbf{X}) + \begin{bmatrix} \sigma_s^2 \mathbb{I}_{n^s} & \mathbf{0} \\ \mathbf{0} & \sigma_t^2 \mathbb{I}_{m^t} \end{bmatrix}$, and $\mathbf{X} = \{(x_i^r, \mathbb{P}^r)\}$ ($r \in \{s, t\}$) denotes all the labeled training samples from source and target domains, and $\mathbf{y} = [y_1^s, \dots, y_{n^s}^s, y_1^t, \dots, y_{m^t}^t]^T \in \mathbb{R}^{n^s+m^t}$ by the corresponding ground-truth output values. For notation simplicity, we represent the distribution-informed kernel $\tilde{\mathcal{K}}_{\text{NNGP}}((x, \mathbb{P}), (x', \mathbb{P}'))$ as $\tilde{\mathcal{K}}_{\text{NNGP}}(x, x')$.

The proposed DINO-INIT algorithm involves the following parameters: the coefficient β_{x, x_i} in the kernel function $\tilde{\mathcal{K}}_{\text{NNGP}}$ and the noise variance σ_s, σ_t . Following [90], we would optimize these parameters by maximizing the conditional likelihood $p(\mathbf{y}_l^t | \mathbf{X}_l^t, \mathbf{X}^s, \mathbf{y}^s)$, where $\mathbf{X}^s = \{(x_i^s, \mathbb{P}^s)\}_{i=1}^{n^s}$, $\mathbf{y}^s = [y_1^s, \dots, y_{n^s}^s]^T$ and $\mathbf{X}_l^t = \{(x_j^t, \mathbb{P}^t)\}_{j=1}^{m^t}$, $\mathbf{y}_l^t = [y_1^t, \dots, y_{m^t}^t]^T$. It can be seen that $p(\mathbf{y}_l^t | \mathbf{X}_l^t, \mathbf{X}^s, \mathbf{y}^s)$ is also a Gaussian process, i.e.,

$$\begin{aligned}p(\mathbf{y}_l^t | \mathbf{X}_l^t, \mathbf{X}^s, \mathbf{y}^s) &= \mathcal{N} \left(\mathbf{K}_{ts} (\mathbf{K}_{ss} + \sigma_s^2 \mathbb{I}_{n^s})^{-1} \mathbf{y}^s, \right. \\ &\quad \left. (\mathbf{K}_{tt} + \sigma_t^2 \mathbb{I}_{m^t}) - \mathbf{K}_{ts} (\mathbf{K}_{ss} + \sigma_s^2 \mathbb{I}_{n^s})^{-1} \mathbf{K}_{st} \right)\end{aligned}\tag{3.6}$$

where $\mathbf{K}_{ss} \equiv \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}^s, \mathbf{X}^s)$ represents the kernel matrix w.r.t. labeled source samples, $\mathbf{K}_{tt} \equiv \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}_l^t, \mathbf{X}_l^t)$ represents the kernel matrix w.r.t. labeled target samples, and $\mathbf{K}_{st} = (\mathbf{K}_{ts})^T \equiv \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}^s, \mathbf{X}_l^t)$ represents the kernel matrix across domains.

Note that it can only optimize the coefficient $\beta_{x, \tilde{x}_i}$ of Eq. (3.2) for training (source or target) samples. The coefficient of testing target samples x_*^t is unknown. To solve this problem, in this work [18], we formulate the estimate of $\beta_{x^r, \tilde{x}_i^r}$ ($r \in \{s, t\}$) as $\beta_{x, \tilde{x}_i^r} = [x^r \circ \tilde{x}_i^r]^T w^r$, where w^r is a domain-specific parameter vector and $\circ$ denotes vector concatenation. As shown in Eq. (3.2), the input-oriented distribution representation can be learned from the domain-specific vector w^r and the basis samples $\tilde{x}_1^r, \dots, \tilde{x}_{n^r}^r$ from domain r .

DINO under Gradient Descent Training. We also consider that the model parameters of the distribution-informed neural network are updated by gradient descent training. Using the distribution-informed neural network, we propose a novel DINO-TRAIN algorithm with

the following objective function.

$$\mathcal{J}(\theta) = \frac{\alpha}{n^s} \sum_{i=1}^{n^s} \left(\tilde{f}(x_i^s, \mathbb{P}^s) - y_i^s \right)^2 + \frac{1-\alpha}{m^t} \sum_{j=1}^{m^t} \left(\tilde{f}(x_j^t, \mathbb{P}^t) - y_j^t \right)^2 + \mu \text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t) \quad (3.7)$$

where $\alpha \in (0, 1)$ and $\mu \geq 0$ are hyperparameters to balance different terms. The first two terms are standard supervised learning losses over the labeled source and target samples, and the third one is the empirical maximum mean discrepancy (MMD) [115] in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_{DA}$ induced by the neural tangent kernel [91] of our distribution-informed neural network, i.e.,

$$\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t) = \left\| \frac{1}{n^s} \sum_{i=1}^{n^s} \nabla_{\theta} \tilde{f}(x_i^s, \mathbb{P}^s) - \frac{1}{n^t + m^t} \sum_{j=1}^{n^t + m^t} \nabla_{\theta} \tilde{f}(x_j^t, \mathbb{P}^t) \right\|_{\mathcal{H}_{DA}}^2 \quad (3.8)$$

In the proposed **DINO-TRAIN** algorithm, we measure the distribution shift using the training dynamics of the distribution-informed neural network over the source and target samples. This is fundamentally different from previous works [31, 53, 88] associated with a two-stage domain discrepancy measurement. That is, previous works require dividing the neural network architecture into two parts: feature extractor and label predictor. In this regard, they first learn the feature representation in hidden neural layers and then measure the domain discrepancy in the latent feature space. In the second stage, they usually require additional modules, e.g., auxiliary neural networks [88, 121] or pre-defined distribution distances [9, 49, 215]. Compared to previous works, our discrepancy measure of Eq. (3.8) can not only unify the deep transfer learning framework with neural networks, but also enable the theoretical convergence and generalization analysis in the following.

Before analyzing the dynamics of the proposed distribution-informed neural network under Eq. (3.7), we introduce the following assumption.

Assumption 3.1. Given a set of labeled samples $\mathbf{X}$, we assume $\lambda_{\min}(\mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) > 0$ and $\lambda_{\min}(\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbf{X}, \mathbf{X})) > 0$, where $\mathcal{K}_{\text{NTK}}(x, x') = \langle \nabla_{\theta} f(x; \theta), \nabla_{\theta} f(x'; \theta) \rangle$ is standard neural tangent kernel (NTK) induced by $f(\cdot; \theta)$ with infinite width, and $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbf{X}, \mathbf{X})$ is the input-oriented distribution kernel⁴ (see Eq. (3.4)). Here, $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue.

The following theorem shows the convergence of the proposed distribution-informed neural network $\tilde{f}(\cdot)$ under gradient descent.

⁴We let $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(x, x') \equiv \mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ for brevity.

Theorem 3.1. For any coefficient $\beta_{x, \tilde{x}_i}$ of the input-oriented distribution representation, there exists $\eta^* \in \mathbb{R}_+$ such that for the infinitely-wide distribution-informed neural network $\tilde{f}(\cdot)$ trained under gradient flow with learning rate $\eta < \eta^*$, the predicted value $\tilde{f}_t(x_*^t)$ over test target data x_*^t at the time step t is

$$\tilde{f}_t(x_*^t) = \tilde{f}_0(x_*^t) - \tilde{\mathcal{K}}_{\text{NTK}}(x_*^t, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \left(\mathbb{I} - e^{-\eta \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \tilde{\mathbf{C}}_t} \right) \left(\tilde{f}_0(\mathbf{X}) - \mathbf{y} \right) \quad (3.9)$$

where $\tilde{\mathcal{K}}_{\text{NTK}}(\cdot, \cdot)$ is a distribution-informed neural tangent kernel (NTK), i.e., $\tilde{\mathcal{K}}_{\text{NTK}}(x, x') = \mathcal{K}_{\text{NTK}}(x, x') \cdot \mathcal{K}_{\mathcal{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ and $\tilde{\mathbf{C}} = \text{diag}\{\underbrace{\alpha/n^s, \dots, \alpha/n^s}_{n^s}, \underbrace{(1-\alpha)/m^t, \dots, (1-\alpha)/m^t}_{m^t}\}$ is a diagonal matrix. Moreover, under Assumption 3.1, when the network width goes to infinity, $\lim_{t \rightarrow \infty} \tilde{f}_t(x_*^t)$ converges to a Gaussian process with the mean $\mu(x_*^t)$ and the covariance $\Sigma(x_*^t, x_*^t)$ as follows.

$$\begin{aligned} \mu(x_*^t) &= \tilde{\mathcal{K}}_{\text{NTK}}(x_*^t, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \mathbf{y} \\ \Sigma(x_*^t, x_*^t) &= \tilde{\mathcal{K}}_{\text{NNGP}}(x_*^t, x_*^t) + \tilde{\mathcal{K}}_{\text{NTK}}(x_*^t, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NNGP}} \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, x_*^t) \\ &\quad - \left(\tilde{\mathcal{K}}_{\text{NTK}}(x_*^t, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}, x_*^t) + h.c. \right) \end{aligned} \quad (3.10)$$

where “+h.c.” means “plus the Hermitian conjugate”.

Proof. The objective function of Eq. (3.7) can be rewritten as follows.

$$\mathcal{J}(\theta) = \left\| \mathbf{C} \tilde{f}(\mathbf{X}) - \mathbf{C} \mathbf{y} \right\|_2^2 + \mu \text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t) \quad (3.11)$$

where $\tilde{f}(\mathbf{X}) = \text{vec}(\{\tilde{f}(x_i^r, \mathbb{P}^r)\}_{i=1}^{n^s+m^t})$ ($r \in \{s, t\}$), and

$$\mathbf{C} = \text{diag}\{\underbrace{\sqrt{\alpha/n^s}, \dots, \sqrt{\alpha/n^s}}_{n^s}, \underbrace{\sqrt{(1-\alpha)/m^t}, \dots, \sqrt{(1-\alpha)/m^t}}_{m^t}\} \quad (3.12)$$

is a constant diagonal matrix. The neural tangent kernel [91] of the proposed distribution-informed neural network $\tilde{f}(\cdot)$ can be defined as

$$\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) = \lim_{\tilde{d} \rightarrow \infty} \nabla_{\theta_0} \tilde{f}(\mathbf{X}) \nabla_{\theta_0} \tilde{f}(\mathbf{X})^T \quad (3.13)$$

where $\tilde{d}$ denotes the layer width of $\tilde{f}(\cdot)$. Moreover, based on the definition of $\tilde{f}(\cdot)$, we have

$$\lim_{\tilde{d} \rightarrow \infty} \left\langle \nabla_{\theta_0} \tilde{f}(x, \mathbb{P}), \nabla_{\theta_0} \tilde{f}(x', \mathbb{P}') \right\rangle = \mathcal{K}_{\text{NTK}}(x, x') \cdot \mathcal{K}_{\mathcal{P}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') \quad (3.14)$$

$$\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) = \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \odot \mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbf{X}, \mathbf{X}) \quad (3.15)$$

where $\mathcal{K}_{\text{NTK}}(x, x') = \lim_{\tilde{d} \rightarrow \infty} \langle \nabla_{\theta_0} f(x), \nabla_{\theta_0} f(x') \rangle$ is standard neural tangent kernel [91] induced by $f(\cdot; \theta)$ (i.e., the first component of Eq. (3.1)) with infinite neural width.

It has been shown [210] that a wide neural network can be approximated by a linearized neural network (given by its first-order Taylor expansion).

$$f_t^{\text{lin}}(x) = f_0(x) + \nabla_{\theta} f_0(x) (\theta_t - \theta_0) \quad (3.16)$$

and $f_t(x) - f_t^{\text{lin}}(x) = \mathcal{O}(1/\sqrt{\tilde{d}}) \rightarrow 0$ ($\tilde{d} \rightarrow \infty$).

Similarly, we have $\tilde{f}_t(\mathbf{X}) = \tilde{f}_0(\mathbf{X}) + \nabla_{\theta} \tilde{f}_0(\mathbf{X}) (\theta_t - \theta_0) + \mathcal{O}(1/\sqrt{\tilde{d}})$. Using the linearized function $\tilde{f}_t^{\text{lin}}(\mathbf{X}) = \tilde{f}_0(\mathbf{X}) + \nabla_{\theta} \tilde{f}_0(\mathbf{X}) (\theta_t - \theta_0)$, we have the dynamics of linearized neural network as follows.

$$\dot{\theta}_t = -\eta \nabla_{\theta} \mathcal{J}(\theta) = -\eta \nabla_{\theta} \tilde{f}_t^{\text{lin}}(\mathbf{X})^T \mathbf{C}^T \mathbf{C} \left(\tilde{f}_t^{\text{lin}}(\mathbf{X}) - \mathbf{y} \right) \quad (3.17)$$

$$= -\eta \nabla_{\theta} \tilde{f}_0(\mathbf{X})^T \mathbf{C}^T \mathbf{C} \left(\tilde{f}_0(\mathbf{X}) + \nabla_{\theta} \tilde{f}_0(\mathbf{X}) (\theta_t - \theta_0) - \mathbf{y} \right) \quad (3.18)$$

The ordinary differential equations (ODEs) have a closed-form solution. We have

$$\theta_t = \theta_0 - \nabla_{\theta} \tilde{f}_0(\mathbf{X})^T \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \left(\mathbb{I} - e^{-\eta \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \mathbf{C}^T \mathbf{C} t} \right) \left(\tilde{f}_0(\mathbf{X}) - \mathbf{y} \right) \quad (3.19)$$

Then given random initialization θ_0 , the predictions of $\tilde{f}_t$ over training samples X and any testing sample x_*^t are

$$\tilde{f}_t(\mathbf{X}) \approx \tilde{f}_t^{\text{lin}}(\mathbf{X}) = \tilde{f}_0(\mathbf{X}) - \left(\mathbb{I} - e^{-\eta \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \mathbf{C}^T \mathbf{C} t} \right) \left(\tilde{f}_0(\mathbf{X}) - \mathbf{y} \right) \quad (3.20)$$

$$\tilde{f}_t(x_*^t) \approx \tilde{f}_t^{\text{lin}}(x_*^t) = \tilde{f}_0(x_*^t) - \tilde{\mathcal{K}}_{\text{NTK}}(x_*^t, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \left(\mathbb{I} - e^{-\eta \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \mathbf{C}^T \mathbf{C} t} \right) \left(\tilde{f}_0(\mathbf{X}) - \mathbf{y} \right) \quad (3.21)$$

with up to an error of $\mathcal{O}(1/\sqrt{\tilde{d}})$.

Here the minimum eigenvalue

$$\begin{aligned} \lambda_{\min}(\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \mathbf{C}^T \mathbf{C}) &\geq \lambda_{\min}(\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) \lambda_{\min}(\mathbf{C}^T \mathbf{C}) \\ &= \lambda_{\min}(\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) \cdot \min\{\alpha/n^s, (1-\alpha)/m^t\} \end{aligned} \quad (3.22)$$

Following [216], using $\lambda_{\min}(\mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) > 0$ and $\lambda_{\min}(\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbf{X}, \mathbf{X})) > 0$ in Assump-

tion 3.1, we have $\lambda_{\min}(\tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) \geq 0$. When $\alpha \in (0, 1)$, the following holds

$$\lim_{t \rightarrow \infty} \lim_{\tilde{d} \rightarrow \infty} \tilde{f}_t(x_*) = \tilde{f}_0(x_*) - \tilde{\mathcal{K}}_{\text{NTK}}(x_*, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} (\tilde{f}_0(\mathbf{X}) - \mathbf{y}) \quad (3.23)$$

Over random initialization of θ_0 , $\tilde{f}_t(x_*)$ converges to a Gaussian distribution, as it is a linear transformation of $\tilde{f}_0(x_*)$ associated with Gaussian distribution. Using $\mathbb{E}[\tilde{f}_0(x_*)] = 0$ and $\mathbb{E}[\tilde{f}_0(x_*) \tilde{f}_0(x_*)] = \tilde{\mathcal{K}}_{\text{NNGP}}(x_*, x_*)$, we have the following results:

$$\begin{aligned} \mu(x_*) &= \tilde{\mathcal{K}}_{\text{NTK}}(x_*, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \mathbf{y} \\ \Sigma(x_*, x_*) &= \tilde{\mathcal{K}}_{\text{NNGP}}(x_*, x_*) + \tilde{\mathcal{K}}_{\text{NTK}}(x_*, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NNGP}} \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, x_*) \\ &\quad - \left(\tilde{\mathcal{K}}_{\text{NTK}}(x_*, \mathbf{X}) \tilde{\mathcal{K}}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}, x_*) + h.c. \right) \end{aligned} \quad (3.24)$$

which completes the proof. QED.

In addition, we derive the generalization bounds of DINO-TRAIN in the following theorem.

Theorem 3.2. Assume for any training sample $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we have $(\tilde{f}(x, \mathbb{P}) - y)^2 \leq M_0$ for some constant $M_0 \geq 0$. Let $\tilde{\mathcal{F}}$ be the hypothesis space induced by infinitely wide neural networks $\tilde{f}$. Then, for any $\tilde{f} \in \tilde{\mathcal{F}}$ and $\delta > 0$, with probability at least $1 - \delta$, the expected error in the target domain can be bounded as follows.

$$\begin{aligned} \mathcal{E}^t(\tilde{f}) &\leq \frac{\alpha}{n^s} \sum_{i=1}^{n^s} \left(\tilde{f}(x_i^s, \mathbb{P}^s) - y_i^s \right)^2 + \frac{1 - \alpha}{m^t} \sum_{j=1}^{m^t} \left(\tilde{f}(x_j^t, \mathbb{P}^t) - y_j^t \right)^2 \\ &\quad + 8\alpha M_0 \text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}(\mathbb{P}^s, \mathbb{P}^t) + \Omega \end{aligned} \quad (3.25)$$

where $\Omega = 2\alpha \mathfrak{R}_{n^s}(\mathcal{H}^s) + 2(1 - \alpha) \mathfrak{R}_{m^t}(\mathcal{H}^t) + M \sqrt{\frac{(n^s + m^t) \log(1/\delta)}{2}}$, $\mathcal{H}_r = \{(x, y) \rightarrow (\tilde{f}(x^r, \mathbb{P}^r) - y^r)^2 : \tilde{f} \in \tilde{\mathcal{F}}\}$ is a set of functions ($r \in \{s, t\}$), $\mathfrak{R}_{n^r}(\mathcal{H}^r)$ is the Rademacher complexity of $\mathcal{H}^r$ given n^r samples, and $M = \max\{\alpha M_0/n^s, (1 - \alpha) M_0/m^t\}$.

Proof. We start by denoting $\Psi(\mathbf{X}) = \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \mathcal{E}^t(\tilde{f}) - \alpha \hat{\mathcal{E}}^s(\tilde{f}) - (1 - \alpha) \hat{\mathcal{E}}^t(\tilde{f})$, where $\hat{\mathcal{E}}^s(\tilde{f}) = \frac{1}{n^s} \sum_{i=1}^{n^s} \left(\tilde{f}(x_i^s, \mathbb{P}^s) - y_i^s \right)^2$ and $\hat{\mathcal{E}}^t(\tilde{f}) = \frac{1}{m^t} \sum_{j=1}^{m^t} \left(\tilde{f}(x_j^t, \mathbb{P}^t) - y_j^t \right)^2$. When one point of $\mathbf{X}$ is changed, $\Psi(\mathbf{X})$ will change at most $M = \max\{\alpha M_0/n^s, (1 - \alpha) M_0/m^t\}$. Using McDiarmid's inequality, it holds that

$$\Pr[\Psi(\mathbf{X}) - \mathbb{E}[\Psi(\mathbf{X})] > \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{(n^s + m^t)M^2}\right) \quad (3.26)$$

Thus, for any $\delta > 0$, with probability at least $1 - \delta$, we have

$$\Psi(\mathbf{X}) \leq \alpha \hat{\mathcal{E}}^s(\tilde{f}) + (1 - \alpha) \hat{\mathcal{E}}^t(\tilde{f}) + \mathbb{E}[\Psi(\mathbf{X})] + M \sqrt{\frac{(n^s + m^t) \log(1/\delta)}{2}} \quad (3.27)$$

Moreover,

$$\begin{aligned} \mathbb{E}[\Psi(X)] &= \mathbb{E} \left[\sup_{\tilde{f} \in \tilde{\mathcal{F}}} \alpha \cdot \hat{\mathcal{E}}^s(\tilde{f}) + (1 - \alpha) \cdot \hat{\mathcal{E}}^t(\tilde{f}) - \mathcal{E}^t(\tilde{f}) \right] \\ &\leq \alpha \cdot \mathbb{E} \left[\sup_{\tilde{f} \in \tilde{\mathcal{F}}} \hat{\mathcal{E}}^s(\tilde{f}) - \mathcal{E}^t(\tilde{f}) \right] + (1 - \alpha) \cdot \mathbb{E} \left[\sup_{\tilde{f} \in \tilde{\mathcal{F}}} \hat{\mathcal{E}}^t(\tilde{f}) - \mathcal{E}^t(\tilde{f}) \right] \\ &\leq 2\alpha \mathfrak{R}_{n^s}(\mathcal{H}^s) + 2(1 - \alpha) \mathfrak{R}_{m^t}(\mathcal{H}^t) + \alpha \cdot \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \mathcal{E}^s(\tilde{f}) - \mathcal{E}^t(\tilde{f}) \end{aligned} \quad (3.28)$$

When $(\tilde{f}(x, \mathbb{P}) - y)_2 \leq M_0$ for any training example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, Lemma 23 of [114] shows that $\left| (\tilde{f}(x, \mathbb{P}) - y)^2 - (\tilde{f}'(x, \mathbb{P}) - y)^2 \right| \leq 2M_0 \left| \tilde{f}(x, \mathbb{P}) - \tilde{f}'(x, \mathbb{P}) \right|$. Therefore, we have $(\tilde{f}(x, \mathbb{P}) - y)^2 \leq 2M_0 \left| \tilde{f}(x, \mathbb{P}) - y \right|$. It has been shown in [91] that an infinitely wide neural network would behave as its linearization around the initialization and achieve zero training loss under MSE loss. Thus, there exists a small perturbation $\Delta\theta$ over parameters for each example (x, y) such that $\tilde{f}_{\theta+\Delta\theta}(x, \mathbb{P}) = y$.

$$\begin{aligned} \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \mathcal{E}^s(\tilde{f}) - \mathcal{E}^t(\tilde{f}) &\leq \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \left| \mathbb{E}_{\mathbb{P}^s} \left(\tilde{f}(x^s, \mathbb{P}^s) - y^s \right)^2 - \mathbb{E}_{\mathbb{P}^t} \left(\tilde{f}(x^t, \mathbb{P}^t) - y^t \right)^2 \right| \\ &\leq 4M_0 \cdot \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \left| \mathbb{E}_{\mathbb{P}^s} \left| \tilde{f}(x^s, \mathbb{P}^s) - y^s \right| - \mathbb{E}_{\mathbb{P}^t} \left| \tilde{f}(x^t, \mathbb{P}^t) - y^t \right| \right| \\ &\leq 8M_0 \cdot \sup_{\tilde{f} \in \tilde{\mathcal{F}}} \left| \mathbb{E}_{\mathbb{P}^s} \left[\tilde{f}(x^s, \mathbb{P}^s) - y^s \right] - \mathbb{E}_{\mathbb{P}^t} \left[\tilde{f}(x^t, \mathbb{P}^t) - y^t \right] \right| \\ &\leq 8M_0 \cdot \sup_{\Delta\theta \in \mathcal{H}_{DA}, \tilde{f} \in \tilde{\mathcal{F}}} \left| \mathbb{E}_{\mathbb{P}^s} \left[\tilde{f}_{\theta}(x^s, \mathbb{P}^s) - \tilde{f}_{\theta+\Delta\theta}(x^s, \mathbb{P}^s) \right] - \mathbb{E}_{\mathbb{P}^t} \left[\tilde{f}(x^t, \mathbb{P}^t) - \tilde{f}_{\theta+\Delta\theta}(x^t, \mathbb{P}^t) \right] \right| \\ &= 8M_0 \cdot \sup_{\Delta\theta \in \mathcal{H}_{DA}, \tilde{f} \in \tilde{\mathcal{F}}} \left| \mathbb{E}_{\mathbb{P}^s} \left[\nabla_{\theta} \tilde{f}_{\theta}(x^s, \mathbb{P}^s) \Delta\theta \right] - \mathbb{E}_{\mathbb{P}^t} \left[\nabla_{\theta} \tilde{f}(x^t, \mathbb{P}^t) \Delta\theta \right] \right| \\ &= 8M_0 \cdot \text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}(\mathbb{P}^s, \mathbb{P}^t) \end{aligned} \quad (3.29)$$

which $\mathcal{H}_{DA}$ is the RKHS induced by kernel $\tilde{\mathcal{K}}_{\text{NTK}}$.

QED.

3.1.4 Connections to Existing Transfer Learning Frameworks

Transferable Gaussian Processes. One special case of distribution-informed neural network at initialization is that when the coefficient $\beta_{x,x_i} = 1/n\|\bar{\Phi}_x(\mathbb{P})\|_{\mathcal{K}_{\text{NNGP}}}$ and $\bar{\Phi}_x(\mathbb{P}) = \frac{1}{n} \sum_{i=1}^n \langle \cdot, x_i \rangle_{\mathcal{K}_{\text{NNGP}}}$, it can be seen that $\Phi_x(\mathbb{P}) = \frac{1}{n\bar{\Phi}_x(\mathbb{P})} \sum_{i=1}^n \langle \cdot, x_i \rangle_{\mathcal{K}_{\text{NNGP}}}$ is the normalized mean mapping [115] of data distribution $\mathbb{P}$ in the NNGP kernel space. Then, the distribution kernel $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ in Eq. (3.4) can be explained as the inner product of the mean mappings of $\mathbb{P}$ and $\mathbb{P}'$. Moreover, using the definition of maximum mean discrepancy (MMD) [115], it can be shown that $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = \frac{c - \text{MMD}_{\mathcal{K}_{\text{NNGP}}}^2(\mathbb{P}, \mathbb{P}')}{2\bar{\Phi}_x(\mathbb{P})\bar{\Phi}_x(\mathbb{P}')}$, where $\text{MMD}_{\mathcal{K}_{\text{NNGP}}}(\cdot)$ denotes the MMD in a NNGP kernel space, and $c = \frac{1}{n^2} \sum_{i,j=1}^n \mathcal{K}_{\text{NNGP}}(x_i, x_j) + \frac{1}{n'^2} \sum_{i,j=1}^{n'} \mathcal{K}_{\text{NNGP}}(x'_i, x'_j)$. It implies that $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ is negatively correlated with the popular MMD estimator. MMD measures the distribution distance using the difference of mean mappings of distributions, whereas $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x')$ measures the distribution similarity via the inner product of the mean mappings of the distributions $\mathbb{P}$ and $\mathbb{P}'$ in the NNGP kernel space. The following corollary shows that the existing transferable Gaussian process [90] can be explained as implicitly learning the distribution-informed representation in the kernel space.

Corollary 3.1. With the assumptions in Lemma 3.1, when $\beta_{x,\tilde{x}_i} = 1/\|\sum_{i=1}^n \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}}\|$, the Gaussian process induced by $\tilde{f}(x)$ at the initialization would be equivalent to the transferable Gaussian process in [90] over NNKP kernel.

Proof. The Gaussian process in [90] employs the following adaptive transfer kernel.

$$\mathcal{K}'(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} \mathcal{K}(\mathbf{X}^s, \mathbf{X}^s) & \tau \cdot \mathcal{K}(\mathbf{X}^s, \mathbf{X}^t) \\ \tau \cdot \mathcal{K}(\mathbf{X}^t, \mathbf{X}^s) & \mathcal{K}(\mathbf{X}^t, \mathbf{X}^t) \end{bmatrix} \quad (3.30)$$

where $\tau \in [0, 1]$ is a trainable parameter explicitly indicating the relatedness of the source and target domains, and $\mathcal{K}(\cdot, \cdot)$ is a base kernel.

When the coefficient $\beta_{x,\tilde{x}_i} = 1/\|\sum_{i=1}^n \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}}\|$, if $\mathbb{P} = \mathbb{P}'$, $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = 1$, otherwise $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}'|x, x') = \frac{1}{\|\sum_{i=1}^n \langle \cdot, \tilde{x}_i \rangle_{\mathcal{K}_{\text{NNGP}}}\| \cdot \|\sum_{j=1}^{n'} \langle \cdot, \tilde{x}'_j \rangle_{\mathcal{K}_{\text{NNGP}}}\|} \sum_{i=1}^n \sum_{j=1}^{n'} \mathcal{K}_{\text{NNGP}}(\tilde{x}_i, \tilde{x}'_j)$. In this case, the adaptive transfer kernel $\tilde{\mathcal{K}}_{\text{NNGP}}(\mathbf{X}, \mathbf{X})$ induced by $\tilde{f}(x)$ at the initialization would be equivalent to [90], by setting the base kernel $\mathcal{K}(\cdot, \cdot)$ as NNGP kernel $\mathcal{K}_{\text{NNGP}}(\cdot, \cdot)$ and the transfer kernel parameter τ as $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}^s, \mathbb{P}^t|x^s, x^t)$ for any inputs x^s, x^t . QED.

Sample Reweighting: Sample reweighting is another popular technique for addressing the distribution shift problem in transfer learning [217, 218, 219]. The core idea is to correct the difference between distributions by multiplying the prediction loss of each source example by

a non-negative weight. The objective function of sample reweighting techniques in transfer learning can be written as follows.

$$\mathcal{J}_{RW}(\theta) = \frac{\alpha}{n^s} \sum_{i=1}^{n^s} w_i^s (f(x_i^s; \theta) - y_i^s)_2^2 + \frac{1-\alpha}{m^t} \sum_{j=1}^{m^t} (f(x_j^t; \theta) - y_j^t)_2^2 \quad (3.31)$$

where $w^s(x, y) = \frac{\mathbb{P}^t(x, y)}{\mathbb{P}^s(x, y)} \geq 0$ denotes the importance of every source example. In practice, $w^s(x, y)$ can be estimated using kernel mean matching [217] or adversarial learning [26, 220]. The following corollary states the connection between our framework of Eq. (3.7) and the sample reweighting techniques of Eq. (3.31) in the function space.

Corollary 3.2. With the assumption in Theorem 3.1, the learned prediction function of our framework in Eq. (3.7) based on the distribution-informed neural network $\tilde{f}(\cdot)$ can recover the learned prediction function of sample reweighting in Eq. (3.31) in the function space.

Proof. We start by deriving the learned prediction function of sample reweighting under infinitely wide neural network $f(\cdot; \theta)$ in Eq. (3.31).

$$\mathcal{J}_{RW}(\theta) = \frac{\alpha}{n^s} \sum_{i=1}^{n^s} w_i^s (f(x_i^s; \theta) - y_i^s)_2^2 + \frac{1-\alpha}{m^t} \sum_{j=1}^{m^t} (f(x_j^t; \theta) - y_j^t)_2^2 = \|\mathbf{CB}f(\mathbf{X}) - \mathbf{CB}\mathbf{y}\|_2^2 \quad (3.32)$$

where $\mathbf{C} = \text{diag}\{\underbrace{\sqrt{\alpha/n^s}, \dots, \sqrt{\alpha/n^s}}_{n^s}, \underbrace{\sqrt{(1-\alpha)/m^t}, \dots, \sqrt{(1-\alpha)/m^t}}_{m^t}\}$ is a diagonal matrix, and $\mathbf{B} = \text{diag}\{\underbrace{\sqrt{w_1^s}, \dots, \sqrt{w_{n^s}^s}}_{n^s}, \underbrace{1, \dots, 1}_{m^t}\}$. We have the dynamics of the linearized neural network as follows ($\tilde{\mathbf{B}} = \mathbf{CB}$ for brevity).

$$\begin{aligned} \dot{\theta}_t &= -\eta \nabla_{\theta} \mathcal{L}_{RW}(\theta) = -\eta \nabla_{\theta} f_t^{\text{lin}}(\mathbf{X})^T \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} (f_t^{\text{lin}}(\mathbf{X}) - \mathbf{y}) \\ &= -\eta \nabla_{\theta} f_0(\mathbf{X})^T \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} (f_0(\mathbf{X}) + \nabla_{\theta_0} f_0(\mathbf{X}) (\theta_t - \theta_0) - \mathbf{y}) \end{aligned} \quad (3.33)$$

Then the ODE has a closed-form solution as

$$\theta_t = \theta_0 - \nabla_{\theta} f_0(\mathbf{X})^T \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \left(\mathbb{I} - e^{-\eta \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} t} \right) (f_0(\mathbf{X}) - \mathbf{y}) \quad (3.34)$$

Then given random initialization θ_0 , the predictions of this neural network over training

samples $\mathbf{X}$ and testing sample x_*^t are

$$f_t(x_*^t) \approx f_t^{\text{lin}}(x_*^t) = f_0(x_*^t) - \mathcal{K}_{\text{NTK}}(x_*^t, \mathbf{X}) \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} \left(\mathbb{I} - e^{-\eta \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} t} \right) (f_0(\mathbf{X}) - \mathbf{y}) \quad (3.35)$$

with up to an error of $\mathcal{O}(1/\sqrt{\tilde{d}})$. Therefore, if $w_i^s > 0$ for all $i = 1, \dots, n^s$, it can be seen that $\lambda_{\min}(\mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X}) \tilde{\mathbf{B}}^T \tilde{\mathbf{B}}) \geq \lambda_{\min}(\mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})) \lambda_{\min}(\tilde{\mathbf{B}}^T \tilde{\mathbf{B}}) > 0$. In this case, we have $\lim_{t \rightarrow \infty} \lim_{\tilde{d} \rightarrow \infty} f_t(x_*^t) = f_0(x_*^t) - \mathcal{K}_{\text{NTK}}(x_*^t, \mathbf{X}) \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} (f_0(\mathbf{X}) - \mathbf{y})$. If $w_i^s = 0$ for some i , i.e., some source samples are ignored during training, the objective function $\mathcal{J}_{RW}$ can be rewritten as the one depending only on the labeled source samples with $w_i^s > 0$. As a result, we have $\lim_{t \rightarrow \infty} \lim_{\tilde{d} \rightarrow \infty} f_t(x_*^t) = f_0(x_*^t) - \mathcal{K}_{\text{NTK}}(x_*^t, \mathbf{X}') \mathcal{K}_{\text{NTK}}(\mathbf{X}', \mathbf{X}')^{-1} (f_0(\mathbf{X}') - \mathbf{y}')$, where $\mathbf{X}' = \mathbf{X} \setminus \{x_i^s | w_i^s = 0\}$ and $\mathbf{y}' = \mathbf{y} \setminus \{y_i^s | w_i^s = 0\}$ denote the training samples after removing some source samples with $w_i^s = 0$.

In the following, we consider a special case of the input-oriented distribution representation function $g(\mathbb{P}|x; w_g)$ in Eq. (3.1). That is, $g(\mathbb{P}^s | x_i^s; w_g) = \mathbb{1}[w_i^s > 0]$ for any i and $g(\mathbb{P}^t | x_j^t; w_g) = 1$ for any j , where $\mathbb{1}[w_i^s] = 1$ if $w_i^s > 0$, $\mathbb{1}[w_i^s] = 0$ otherwise. In this case, we can show that if $w_i^s > 0$ for all $i = 1, \dots, n^s$, $\mathcal{K}_{\mathcal{D}|\mathcal{X}}(\mathbb{P}, \mathbb{P}' | x, x') = 1$ for any x, x' from source and target domains. Then $\tilde{\mathcal{K}}_{\text{NTK}}(x, x') = \mathcal{K}_{\text{NTK}}(x, x')$ for any x, x' in Theorem 3.1. It holds that $\lim_{t \rightarrow \infty} \lim_{\tilde{d} \rightarrow \infty} \tilde{f}_t(x_*^t) = \tilde{f}_0(x_*^t) - \mathcal{K}_{\text{NTK}}(x_*^t, \mathbf{X}) \mathcal{K}_{\text{NTK}}(\mathbf{X}, \mathbf{X})^{-1} (\tilde{f}_0(\mathbf{X}) - \mathbf{y})$. With the input-oriented constant distribution representation, we have $\tilde{f}_0((x, \mathbb{P})) = f_0(x)$. Thus, the learned prediction functions of the proposed DINO-TRAIN and sample reweighting are identical in the function space. Similarly, if $w_i^s = 0$ for some i , the objective function $\mathcal{J}(\theta)$ can be rewritten as the one depending only on the labeled source samples with $w_i^s > 0$. Then it holds that $\lim_{t \rightarrow \infty} \lim_{\tilde{d} \rightarrow \infty} \tilde{f}_t(x_*^t) = \tilde{f}_0(x_*^t) - \mathcal{K}_{\text{NTK}}(x_*^t, \mathbf{X}') \mathcal{K}_{\text{NTK}}(\mathbf{X}', \mathbf{X}')^{-1} (\tilde{f}_0(\mathbf{X}') - \mathbf{y}')$, which recovers the prediction function of sample reweighting. QED.

Invariant Representation Learning. Transfer learning could discover and transfer the common knowledge shared by the source and target domains. To this end, domain-invariant representation learning maps the source and target samples into a new feature space such that the distribution discrepancy between source and target domains can be explicitly minimized [9, 53, 88]. The objective function of this framework can be summarized as follows.

$$\mathcal{J}_{IRL}(\theta) = \frac{\alpha}{n^s} \sum_{i=1}^{n^s} (f(x_i^s; \theta) - y_i^s)^2 + \frac{1 - \alpha}{m^t} \sum_{j=1}^{m^t} (f(x_j^t; \theta) - y_j^t)^2 + \mu \cdot d(\mathbb{P}^s, \mathbb{P}^t) \quad (3.36)$$

where $d(\mathbb{P}^s, \mathbb{P}^t)$ denotes the domain discrepancy measure in the hidden feature space learned by a deep neural network $f(\cdot; \theta)$. We show that our DINO-TRAIN framework can naturally de-

Methods	C $\rightarrow$ N	C $\rightarrow$ S	N $\rightarrow$ C	N $\rightarrow$ S	S $\rightarrow$ C	S $\rightarrow$ N	Avg.
NNGP	2.041 $\pm$ 0.001	1.823 $\pm$ 0.001	0.445 $\pm$ 0.002	0.624 $\pm$ 0.001	0.197 $\pm$ 0.002	0.459 $\pm$ 0.002	0.932
NTKGP	1.345 $\pm$ 0.002	1.227 $\pm$ 0.000	0.323 $\pm$ 0.002	0.529 $\pm$ 0.004	0.248 $\pm$ 0.001	0.425 $\pm$ 0.002	0.683
AT-GP	0.194 $\pm$ 0.005	0.259 $\pm$ 0.002	0.104$\pm$0.001	0.252 $\pm$ 0.005	0.118 $\pm$ 0.003	0.189 $\pm$ 0.006	0.186
TL-NTK	0.164 $\pm$ 0.001	0.231$\pm$0.000	0.124 $\pm$ 0.005	0.242 $\pm$ 0.002	0.125 $\pm$ 0.001	0.197 $\pm$ 0.004	0.181
DINO-INIT	0.128 $\pm$ 0.001	0.233 $\pm$ 0.003	0.114 $\pm$ 0.002	0.227$\pm$0.002	0.112$\pm$0.001	0.181$\pm$0.005	0.166
DINO-TRAIN	0.127$\pm$0.002	0.240 $\pm$ 0.003	0.127 $\pm$ 0.000	0.243 $\pm$ 0.000	0.128 $\pm$ 0.001	0.194 $\pm$ 0.001	0.177

Table 3.2: Results on dSprites

Methods	RL $\rightarrow$ RC	RL $\rightarrow$ T	RC $\rightarrow$ RL	RC $\rightarrow$ T	T $\rightarrow$ RL	T $\rightarrow$ RC	Avg.
NNGP	0.313 $\pm$ 0.001	0.438 $\pm$ 0.004	0.356 $\pm$ 0.005	0.515 $\pm$ 0.008	0.367 $\pm$ 0.001	0.324 $\pm$ 0.004	0.386
NTKGP	0.396 $\pm$ 0.001	0.365 $\pm$ 0.001	0.200 $\pm$ 0.007	0.390 $\pm$ 0.003	0.390 $\pm$ 0.000	0.354 $\pm$ 0.003	0.349
AT-GP	0.214 $\pm$ 0.011	0.209 $\pm$ 0.002	0.227 $\pm$ 0.010	0.198 $\pm$ 0.002	0.236 $\pm$ 0.000	0.249 $\pm$ 0.000	0.222
TL-NTK	0.206 $\pm$ 0.004	0.200 $\pm$ 0.002	0.213 $\pm$ 0.000	0.197 $\pm$ 0.000	0.226 $\pm$ 0.001	0.218 $\pm$ 0.000	0.210
DINO-INIT	0.204 $\pm$ 0.001	0.185$\pm$0.006	0.207$\pm$0.003	0.182$\pm$0.004	0.218$\pm$0.001	0.212$\pm$0.001	0.201
DINO-TRAIN	0.193$\pm$0.001	0.194 $\pm$ 0.003	0.207$\pm$0.003	0.188 $\pm$ 0.002	0.226 $\pm$ 0.001	0.218 $\pm$ 0.001	0.204

Table 3.3: Results on MPI3D

generate into domain-invariant representation learning when the distribution representation in Eq. (3.2) is shared across all samples.

Corollary 3.3. Under mild conditions, our framework Eq. (3.7) with distribution-informed neural network $\tilde{f}(\cdot)$ can recover the standard domain invariant representation learning in Eq. (3.36), where the domain discrepancy measure $d(\cdot, \cdot)$ is instantiated with MMD in RKHS induced by standard NTK $\mathcal{K}_{\text{NTK}}$.

Proof. If the distribution representation of Eq. (3.2) is shared across samples, for any w_g , $\tilde{f}(x, \mathbb{P}) = a \cdot f(x; \theta)$ where $a = \Phi_x(\mathbb{P})^T w_g \in \mathbb{R}$ is shared by all examples. Then, we have $\tilde{f}(x, \mathbb{P}) = a \cdot f(x; \theta) = a \cdot \phi_{\theta < L}(x)^T w = \phi_{\theta < L}(x)^T (aw)$. Therefore, in this case, $\tilde{f}(\cdot)$ is equivalent to a simple distribution-free neural network. The overall framework Eq. (3.7) would degenerate into the standard domain invariant representation learning in Eq. (3.36). Notice that when using the distribution-free neural network, the framework Eq. (3.36) requires an additional discrepancy minimization regularization to guarantee the success of domain adaptation. In our case, the discrepancy minimization regularization would be given by the MMD in RKHS induced by the NTK $\tilde{\mathcal{K}}_{\text{NTK}}$. Here, based on the definition of $\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t)$ in Eq. (3.8), it holds that $\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t) = a^2 \cdot \text{MMD}_{\mathcal{K}_{\text{NTK}}}^2(\mathbb{P}^s, \mathbb{P}^t)$. QED.

3.1.5 Experiment

The effectiveness of the proposed **DINO-INIT** and **DINO-TRAIN** algorithms is validated. Below, we summarize the experimental setup (e.g., data sets, baselines, and model configuration) and the main observations from the experimental results.

Data Sets. In the experiments, we use the following data sets, including two image data sets for object localization and one data set for agriculture data analysis.

- **dSprites** [221]: It is composed of 737,280 images from three domains: Color (C), Noisy (N), and Scream (S). Following [89], we evaluate all the baselines on six adaptation benchmarks: $C \rightarrow N$, $C \rightarrow S$, $N \rightarrow C$, $N \rightarrow S$, $S \rightarrow C$, and $S \rightarrow N$.
- **MPI3D** [222]: It contains over 3M images from three domains: Toy (T), Realistic (RC), and Real (RL). We evaluate all the baselines on six benchmarks: $T \rightarrow RC$, $T \rightarrow RL$, $RC \rightarrow T$, $RC \rightarrow RL$, $RL \rightarrow T$, and $RL \rightarrow RC$.
- **Plant Phenotyping**: It aims to predict diverse traits (e.g., Nitrogen) of plants related to the plants’ growth using leaf hyperspectral reflectance (i.e., spectral wavelengths 500-2400 nm). Then the input example with spectral wavelengths 500-2400 nm is formulated as a 1901-dimensional feature vector. Here we consider the following two domains [223]: Maize (M) and Maize_UNL (MU), where “M” denotes the collected maize data from Illinois and “MU” denotes the collected maize data from Nebraska. In our case, the task is to predict the Nitrogen content of maize using the leaf hyperspectral reflectance.

Baselines. We use the following Gaussian process baselines, including (1) plain Gaussian processes: NNGP [210] and NTKGP [224]; (2) transferable Gaussian processes: AT-GP [90] and TL-NTK [225].

Model Configuration. In the experiments, our algorithms are implemented based on a L -layer ($L = 6$) fully-connected neural network with the ReLU activation function. In dSprites and MPI3D data sets, the size of input images is $64 \times 64 \times 3$, we simply vectorize the input image to a 12288-dimensional vector. That is, each input image $x \in \mathbb{R}^{12288}$ can be learned by the fully-connected neural network. The NNGP and NTK induced by this neural network can be estimated using the Neural Tangents package [226]. In addition, we set $\alpha = 0.5$ and $\mu = 0.1$ for our **DINO-TRAIN** method.

Main Results. Table 3.2, Table 3.3 and Table 3.4 provide the domain adaptation regression results on dSprites, MPI3D and Plant Phenotyping data sets. Following [89], we report the Mean Absolute Error (MAE) between the predicted outputs and the ground-truth outputs in

Methods	M $\rightarrow$ MU	MU $\rightarrow$ M
NNGP	0.562 $\pm$ 0.001	0.672 $\pm$ 0.010
NTKGP	0.562 $\pm$ 0.004	0.702 $\pm$ 0.010
AT-GP	0.308$\pm$0.006	0.593 $\pm$ 0.025
TL-NTK	0.316 $\pm$ 0.008	0.488 $\pm$ 0.027
DINO-INIT	0.316 $\pm$ 0.007	0.645 $\pm$ 0.017
DINO-TRAIN	0.314 $\pm$ 0.009	0.443$\pm$0.030

Table 3.4: Results on Plant Phenotyping

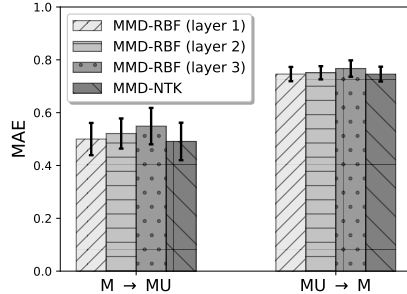


Table 3.5: Discrepancy comparison

the target test set (the best results are indicated in bold). It is observed that our algorithms achieve competitive performance over the Gaussian process baselines. The weakly-trained DINO-INIT slightly outperforms DINO-TRAIN in some cases. This observation is consistent with previous work [227]. The relatively unstable performance of DINO-TRAIN might be caused by the large covariance of the Gaussian process with NTK [224].

Ablation Study. We compare the proposed $\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}(\cdot, \cdot)$ with MMD-RBF (i.e., MMD with standard RBF kernel [115]) for measuring the domain discrepancy. To be specific, we consider a simple 3-layer fully-connected neural network with ReLU. Following [9, 116], we implement the MMD with RBF kernel over the source and target features learned by the first $l = 1, 2, 3$ layers. We denote those approaches as ‘MMD-RBF (layer 1)’, ‘MMD-RBF (layer 2)’ and ‘MMD-RBF (layer 3)’ in Figure 3.5, respectively. For a fair comparison, we use the same neural network architecture to implement the domain adaptation approach with our proposed $\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}(\cdot, \cdot)$, which is denoted as ‘MMD-NTK’ in Figure 3.5. The results on the Plant Phenotyping data set reveal that the selection of feature space affects the model performance when using MMD-RBF, and our proposed $\text{MMD}_{\tilde{\mathcal{K}}_{\text{NTK}}}(\cdot, \cdot)$ outperforms the MMD-RBF.

3.2 STATIC TRANSFER LEARNING WITH MULTIPLE SOURCES

Section 3.1 studies the knowledge transfer across domains when only a single source domain is available [18]. However, in real scenarios, multiple source domains can often be accessed. For instance, when analyzing the semantics of customer reviews for a target product on Amazon [228], it is feasible to leverage knowledge from multiple source products. To this end, in this section, we study the knowledge transfer from multiple static source domains with adequate label information to static target domains with no label information [13].

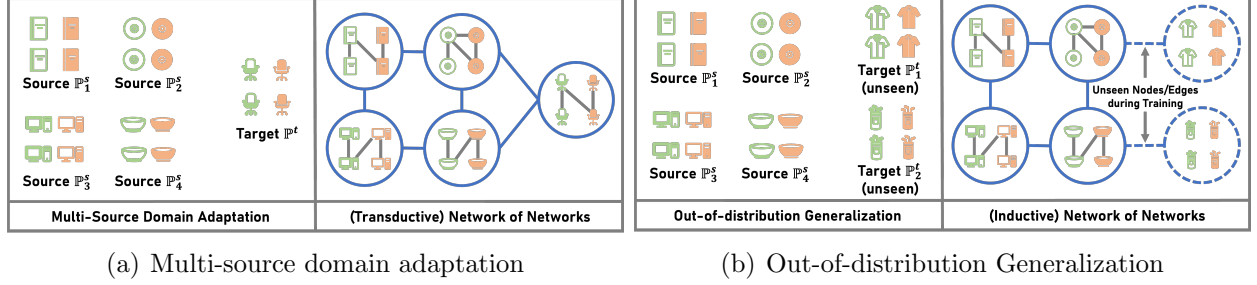


Figure 3.2: Illustration of the network of networks on handling heterogeneous domains (semantic classification on Amazon products [228] is used where green indicates negative review and orange indicates positive review). (a) Multi-source domain adaptation involves multiple labeled source domains and an unlabeled target domain, thus forming a transductive network of networks. (b) For out-of-distribution generalization, target domains are unseen during training, thereby resulting in an inductive network of networks.

3.2.1 Problem Definition

Our work [13] focuses on two research problems: multi-source adaptation [38, 49, 124] and out-of-distribution generalization (a.k.a. domain generalization) [32, 33, 34]. Both research problems involve the knowledge transfer from multiple source domains to a (unseen) target domain. The goal is to learn a prediction function on the target domain using knowledge from relevant source domains. Below, we formally define multi-source domain adaptation in Problem 3.2 [49, 124] and out-of-distribution generalization in Problem 3.3 [32, 34].

Problem 3.2. Given a group of source domains $\{\mathcal{D}_k^s\}_{k=1}^{K-1}$ each with labeled training samples $\{x_{ik}^s, y_{ik}^s\}_{i=1}^{n_k^s} \sim \mathbb{P}_k^s$, and a target domain $\mathcal{D}^t$ with only unlabeled training samples $\{x_i^t\}_{i=1}^{n^t} \sim \mathbb{P}_X^t$, *multi-source domain adaptation* aims to learn a prediction function on the target domain using knowledge from source domains.

Problem 3.3. Given a group of source domains $\{\mathcal{D}_k^s\}_{k=1}^K$ each with labeled training samples $\{x_{ik}^s, y_{ik}^s\}_{i=1}^{n_k^s} \sim \mathbb{P}_k^s$, *out-of-distribution generalization* aims to learn a prediction function from source domains such that this prediction function can be directly applied to unseen target domains.

Similar to Subsection 3.1, we have three assumptions in the aforementioned problem settings. First, all the source and target domains have the same input and output spaces. Second, the samples in each domain are independent and identically distributed (IID). Third, the data distribution of the source/target domain is static. Thus, the major challenge behind multi-source domain adaptation and out-of-distribution generalization lies in the distribution shifts across domains.

Most existing works [49, 112, 124, 229, 230] build a single model to learn the domain-invariant representation from different domains. The invariant representation learned by a domain-agnostic model can be explained as the common knowledge shared by all domains. Nevertheless, it is a strong assumption that all domains share the same model parameters. This is because this assumption underestimates the domain-specific characteristics encoding class separability. Though recent works [231, 232, 233] propose to learn both with-domain specificity and cross-domain commonality, disentangling domain-invariant and domain-specific representations is a nontrivial task. This is because it is challenging to accurately differentiate the domain-invariant representation from the domain-specific representation within samples. The aforementioned frameworks might suffer from the following limitations. First, the connection between the domain relationship and the model (parameters) similarity is under-explored, e.g., similar domains may share similar model parameters [40, 45, 234, 235]. Second, it is not explained how the label information can be adaptively propagated from the source domain to the target domain. Third, little effort has been devoted to understanding the model convergence of previous algorithms [112, 124, 230] based on deep neural networks.

3.2.2 A Generic Framework

We start by introducing a generic framework to characterize the input-output relationship from K different domains.

As illustrated in Figure 3.2, a group of domains can be formulated as a network of networks [236], where each node of the main network represents a domain and each network is formed by domain-specific samples. Specifically, the main network characterizes the relationship among different domains, where each node (blue circles) is a domain, and the edges (solid or dotted blue lines) imply domain similarity. Each domain is further represented by a domain-specific network (e.g., a network within each blue circle), where each node (colored products) is a sample, and the edges (black lines) imply sample similarity. This motivates us to rethink the modeling of data heterogeneity by capturing both sample

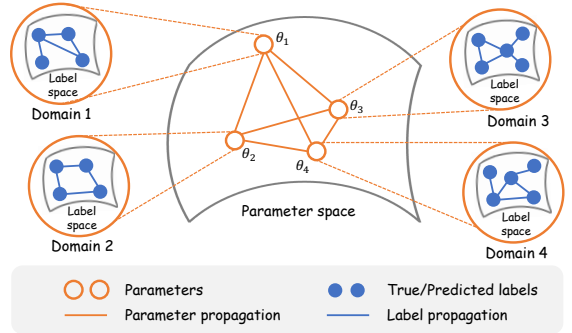


Figure 3.3: Illustration of TENON in information propagation. Label information is propagated in the labeling space within each domain, while parameter information is propagated in the parameter space across domains.

similarity within domains and distribution similarity across domains (illustrated in Figure 3.3). First, in each domain, two samples tend to have similar output values if they are similar in the input space [237, 238]. Second, given a learning function $f(\cdot)$, two domains would be close in the parameter space if they are distributionally similar [40, 234, 239]. To this end, we propose a simple yet generic distributional network of networks (TENON) framework [13]. Given K different domains, we have the following objective function.

$$\begin{aligned}
\min_{\{\theta_k\}_{k=1}^K} & \lambda \sum_{k=1}^K \underbrace{\sum_{i=1}^{n_k} \|f(x_{ik}; \theta_k) - y_{ik}\|_2^2}_{\text{Label consistency within domain}} + \underbrace{\frac{1}{2} \sum_{k=1}^K \sum_{i,j=1}^{n_k} s_{ij}^k \left\| \frac{f(x_{ik}; \theta_k)}{\sqrt{D_{ii}^k}} - \frac{f(x_{jk}; \theta_k)}{\sqrt{D_{jj}^k}} \right\|_2^2}_{\text{Label smoothness within domain}} \\
& + \underbrace{\frac{1}{2} \sum_{k,k'=1}^K d_{kk'} \left\| \frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right\|_F^2}_{\text{Parameter smoothness across domains}}
\end{aligned} \tag{3.37}$$

where each domain is associated with a model $f(\cdot; \theta_k) : \mathcal{X} \rightarrow \mathcal{Y}$ parameterized by θ_k . s_{ij}^k indicates the sample similarity between x_{ik} and x_{jk} within the k -th domain, and $d_{kk'}$ denotes the domain similarity between the k -th domain and the k' -th domain. Here $D_{ii}^k = \sum_{j=1}^{n_k} s_{ij}^k$ and $M_{kk} = \sum_{k'=1}^K d_{kk'}$. θ_k denotes the model parameters within the k -th domain. $\lambda > 0$ is a hyper-parameter to balance different terms.

Following [237], the sample similarity s_{ij}^k can be empirically estimated as follows.

$$s_{ij}^k = \exp(-\sigma \cdot \|x_{ik} - x_{jk}\|^2) \tag{3.38}$$

where $\sigma > 0$ is a hyper-parameter. In addition, a variety of domain discrepancy measures have been proposed to model the heterogeneous domains, e.g., $\mathcal{H}\Delta\mathcal{H}$ -divergence [38], Maximum Mean Discrepancy [9, 115], f -divergence [31], etc. It is flexible to define $d_{kk'}$ in Eq. (3.37) based on existing domain discrepancy measures. In this paper, under the covariate shift assumption [2] (i.e., $\mathbb{P}(y|x)$ is shared for all domains), we use Maximum Mean Discrepancy (MMD) [115] to define the domain similarity $d_{kk'}$ as follows.

$$\text{MMD}(k, k') = \left\| \frac{1}{n_k} \sum_{i=1}^{n_k} \phi(x_{ik}) - \frac{1}{n_{k'}} \sum_{j=1}^{n_{k'}} \phi(x_{jk'}) \right\|_{\mathcal{H}_K}^2 \tag{3.39}$$

$$d_{kk'} = \exp(-\sigma \cdot \text{MMD}(k, k')) \tag{3.40}$$

where $\phi(\cdot) : \mathcal{X} \rightarrow \mathcal{H}_K$ is a kernel mapping from an input space $\mathcal{X}$ to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$.

The intuition behind Eq. (3.37) is explained as follows. The first term captures the consistency of models $\{f(\cdot; \theta_k)\}_{k=1}^K$ with the prior label information. The second term measures the label smoothness within each domain. It implies that input samples have similar prediction values if they are similar in the input space. Furthermore, the third term measures the cross-domain model smoothness in the parameter space. Notably, graph-based parameter smoothness regularization [234, 239] has been studied in multi-task learning. However, compared to previous works, our framework of Eq. (3.37) explicitly reveals the connection between the domain distribution discrepancy and the model (parameters) similarity, i.e., domains have similar model parameters if they are distributionally similar. Furthermore, by incorporating the within-domain label smoothness regularization (i.e., the second term of Eq. (3.37)), **TENON** allows propagating label information from labeled source samples to unlabeled target samples, whereas previous works [234, 239] collaboratively update the model parameters over the labeled samples from all domains.

3.2.3 Theoretical Analysis

The convergence and optimality of **TENON** can be analyzed by considering different instantiations of learning models $\{f(\cdot; \theta_k)\}_{k=1}^K$. In the following, we start with the simple linear regression functions, i.e., $f(x; \theta_k) = \theta_k^T x$ for all $k \in \{1, 2, \dots, K\}$. The following lemma shows the global convergence and optimality of the **TENON** framework.

Lemma 3.2. Given linear models $f(x; \theta_k) = \theta_k^T x$ for $k \in \{1, \dots, K\}$, the objective of Eq. (3.37) can be minimized at

$$\Theta^* = \lambda \hat{\mathbf{X}} \left(\left(\hat{\mathbf{A}} + \lambda \mathbb{I} \right) \hat{\mathbf{X}}^T \hat{\mathbf{X}} + \left(\hat{\mathbf{X}}^T \hat{\mathbf{X}} \right)^{-1} \hat{\mathbf{X}}^T \hat{\mathbf{B}} \hat{\mathbf{X}} \right)^{-1} \mathbf{y} \quad (3.41)$$

where $\Theta = [\theta_1^T, \dots, \theta_K^T]^T$, $\mathbf{X}_k = [x_{1k}, x_{2k}, \dots, x_{n_k k}]$, $\mathbf{y} = [y_{11}, \dots, y_{n_1 1}, \dots, y_{1K}, \dots, y_{n_K K}]^T$ and

$$\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{X}_K \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} \bar{\mathbf{A}}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{A}}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \bar{\mathbf{A}}_K \end{bmatrix} \quad (3.42)$$

$$\hat{\mathbf{B}} = \mathbb{I}_{K d_{in} \times K d_{in}} - \mathbf{B} \otimes \mathbb{I}_{d_{in} \times d_{in}} \quad (3.43)$$

where $\mathbf{A}_k = (\mathbf{D}^k)^{-1/2} \mathcal{S}^k (\mathbf{D}^k)^{-1/2}$ is the normalized sample similarity matrix of domain k with $\bar{\mathbf{A}}_k = \mathbb{I} - \mathbf{A}_k$, and $\mathbf{B} = \mathbf{M}^{-1/2} \mathcal{D} \mathbf{M}^{-1/2}$ is the normalized domain similarity matrix. $\mathcal{S}^k$ is

sample similarity matrix of domain k with the entry $[\mathcal{S}^k]_{ij} = s_{ij}^k$, and $\mathbf{D}^k$ is a diagonal matrix with the entry $[\mathbf{D}^k]_{ii} = D_{ii}^k$. $\mathfrak{D}$ is domain similarity matrix with the entry $[\mathfrak{D}]_{kk'} = d_{kk'}$, and $\mathbf{M}$ is a diagonal matrix with the entry $[\mathbf{M}]_{kk} = M_{kk}$. $\otimes$ denotes the Kronecker product of two matrices. d_{in} is the dimensionality of input samples.

Proof. The objective of Eq. (3.37) can be rewritten as follows.

$$\begin{aligned}
\mathcal{J}(\Theta) &= \frac{1}{2} \sum_{k=1}^K \sum_{i,j=1}^{n_k} s_{ij}^k \left\| \frac{f(x_{ik}; \theta_k)}{\sqrt{D_{ii}^k}} - \frac{f(x_{jk}; \theta_k)}{\sqrt{D_{jj}^k}} \right\|_2^2 + \frac{1}{2} \sum_{k,k'=1}^K d_{kk'} \left\| \frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right\|_2^2 \\
&\quad + \lambda \sum_{k=1}^K \sum_{i=1}^{n_k} \|f(x_{ik}; \theta_k) - y_{ik}\|_F^2 \\
&= \Theta^T \hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T \Theta + \Theta^T \hat{\mathbf{B}} \Theta + \lambda \cdot \left\| \hat{\mathbf{X}}^T \Theta - \mathbf{y} \right\|_2^2
\end{aligned} \tag{3.44}$$

The last equality is derived from the following results. The first term of $\mathcal{J}(\Theta)$ is rewritten as follows.

$$\begin{aligned}
&\sum_{k=1}^K \sum_{i,j=1}^{n_k} s_{ij}^k \left(\frac{f(x_{ik}; \theta_k)}{\sqrt{D_{ii}^k}} - \frac{f(x_{jk}; \theta_k)}{\sqrt{D_{jj}^k}} \right)^2 = \sum_{k=1}^K \sum_{i,j=1}^{n_k} s_{ij}^k \left(\frac{\theta_k^T x_{ik}}{\sqrt{D_{ii}^k}} - \frac{\theta_k^T x_{jk}}{\sqrt{D_{jj}^k}} \right)^2 \\
&= \sum_{k=1}^K \sum_{i,j=1}^{n_k} s_{ij}^k \left(\frac{\theta_k^T x_{ik} (x_{ik})^T \theta_k}{D_{ii}^k} + \frac{\theta_k^T x_{jk} (x_{jk})^T \theta_k}{D_{jj}^k} - 2 \frac{\theta_k^T x_{ik} (x_{jk})^T \theta_k}{\sqrt{D_{ii}^k \cdot D_{jj}^k}} \right) \\
&= \sum_{k=1}^K \left(2 \sum_{i=1}^{n_k} \theta_k^T x_{ik} (x_{ik})^T \theta_k - 2 \sum_{i,j=1}^{n_k} \frac{s_{ij}^k}{\sqrt{D_{ii}^k \cdot D_{jj}^k}} \theta_k^T x_{ik} (x_{jk})^T \theta_k \right) \\
&= \sum_{k=1}^K (2 \theta_k^T \mathbf{X}_k \mathbf{X}_k^T \theta_k - 2 \theta_k^T \mathbf{X}_k \mathbf{A}_k \mathbf{X}_k^T \theta_k) \\
&= 2 \sum_{k=1}^K \theta_k^T \mathbf{X}_k (\mathbf{I}_{n_k \times n_k} - \mathbf{A}_k) \mathbf{X}_k^T \theta_k \\
&= 2 \Theta^T \hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T \Theta
\end{aligned} \tag{3.45}$$

The second term of $\mathcal{J}(\Theta)$ is rewritten as follows.

$$\begin{aligned}
\sum_{k,k'=1}^K d_{kk'} \left\| \frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right\|_2^2 &= \sum_{k,k'=1}^K d_{kk'} \left(\frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right)^T \left(\frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right) \\
&= \sum_{k,k'=1}^K d_{kk'} \left(\frac{\theta_k^T \theta_k}{M_{kk}} + \frac{\theta_{k'}^T \theta_{k'}}{M_{k'k'}} - 2 \frac{\theta_k^T \theta_{k'}}{\sqrt{M_{kk} \cdot M_{k'k'}}} \right) \\
&= 2 \sum_{k=1}^K \theta_k^T \theta_k - 2 \sum_{k,k'=1}^K \frac{d_{kk'}}{\sqrt{M_{kk} \cdot M_{k'k'}}} \theta_k^T \theta_{k'} \\
&= 2\Theta^T \Theta - 2\Theta^T (\mathbf{B} \otimes \mathbf{I}_{d_{in} \times d_{in}}) \Theta = 2\Theta^T \hat{\mathbf{B}} \Theta
\end{aligned} \tag{3.46}$$

The third term of $\mathcal{J}(\Theta)$ can be rewritten as $\sum_{k=1}^K \sum_{i=1}^{n_k} \|f(x_{ik}; \theta_k) - y_{ik}\|_2^2 = \left\| \hat{\mathbf{X}}^T \Theta - \mathbf{y} \right\|_2^2$. Thus, the derivative of $\mathcal{J}(\Theta)$ is given by

$$\begin{aligned}
\frac{\partial \mathcal{J}(\Theta)}{\partial \Theta} &= \frac{\partial \left(\Theta^T \hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T \Theta + \Theta^T \hat{\mathbf{B}} \Theta + \lambda \cdot \left\| \hat{\mathbf{X}}^T \Theta - \mathbf{y} \right\|_2^2 \right)}{\partial \Theta} \\
&= 2 \left(\hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T + \hat{\mathbf{B}} \right) \Theta + 2\lambda \hat{\mathbf{X}} \hat{\mathbf{X}}^T \Theta - 2\lambda \hat{\mathbf{X}} \mathbf{y} \\
&= 2 \left(\hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T + \hat{\mathbf{B}} + \lambda \hat{\mathbf{X}} \hat{\mathbf{X}}^T \right) \Theta - 2\lambda \hat{\mathbf{X}} \mathbf{y}
\end{aligned} \tag{3.47}$$

By setting $\frac{\partial \mathcal{J}(\Theta)}{\partial \Theta} = 0$, the minimizer of $\mathcal{J}(\Theta)$ is obtained at

$$\begin{aligned}
\Theta^* &= \lambda \left(\hat{\mathbf{X}} \hat{\mathbf{A}} \hat{\mathbf{X}}^T + \hat{\mathbf{B}} + \lambda \hat{\mathbf{X}} \hat{\mathbf{X}}^T \right)^{-1} \hat{\mathbf{X}} \mathbf{y} \\
&= \lambda \hat{\mathbf{X}} \left(\hat{\mathbf{A}} \hat{\mathbf{X}}^T \hat{\mathbf{X}} + \left(\hat{\mathbf{X}}^T \hat{\mathbf{X}} \right)^{-1} \left(\hat{\mathbf{X}}^T \hat{\mathbf{B}} \hat{\mathbf{X}} \right) + \lambda \hat{\mathbf{X}}^T \hat{\mathbf{X}} \right)^{-1} \mathbf{y}
\end{aligned} \tag{3.48}$$

which completes the proof. QED.

Next, we instantiate the learning models $\{f(\cdot; \theta_k)\}_{k=1}^K$ with overparameterized neural networks [91, 240]. This allows us to reveal the convergence of **TENON** in Eq. (3.37) with commonly used neural network architectures⁵. For notation simplicity, we will use $f(\cdot; \Theta)$ to denote the overall learning function with $f(x_{ik}; \Theta) = f(x_{ik}; \theta_k)$ for any sample x_{ik} from domain k . It is observed [240] that neural network $f(\cdot; \Theta)$ can be approximated by its linearized version $f^{\text{lin}}(\cdot; \Theta)$, i.e., $\sup_{t \geq 0} \|f_t(x; \Theta) - f_t^{\text{lin}}(x; \Theta)\| = \mathcal{O}(h^{-\frac{1}{2}})$ where h is the width of neural networks. $f_t(\cdot; \Theta)$ denotes the model at time step t , and $f^{\text{lin}}(\cdot; \Theta)$ is given by the first order Taylor expansion of $f(\cdot; \Theta)$: $f_t^{\text{lin}}(x; \Theta) = f_0(x; \Theta) + \nabla f_0(x; \Theta) (\Theta_t - \Theta_0)$ Inspired

⁵Following [240], θ_k denotes the vectorized parameters of the neural network model within domain k here.

by this observation, we generalize the results of Lemma 3.2 by instantiating $\{f(\cdot; \theta_k)\}_{k=1}^K$ with neural networks. The following theorem shows the global convergence of TENON under gradient descent when the layer width of $\{f(\cdot; \theta_k)\}_{k=1}^K$ goes to infinity.

Theorem 3.3. Let $\mathbf{X}$ denote all training samples. In the limit of layer width, the model parameters Θ in the objective of Eq. (3.37) converges to

$$\lim_{t \rightarrow \infty} \Theta_t = -\nabla_{\Theta} f_0(\mathbf{X})^T \mathbf{K}_{\text{NTK}}^{-1} \Gamma^{-1} (\Omega - \lambda \mathbf{y}) + \Theta_0 \quad (3.49)$$

where t is the time step of training process, Θ_0 denotes the initialized parameters, and $f_0(\mathbf{X}) = \text{vec}(f_0(x_{ik}; \theta_k) | i \in \{1, 2, \dots, n_k\}, k \in \{1, 2, \dots, K\})$ is model output with initialized parameters. Moreover, the prediction function $f(\cdot; \theta_k)$ of Eq. (3.37) for any testing sample x_{*k} within domain k converges to

$$\lim_{t \rightarrow \infty} f_t(x_{*k}; \theta_k) = \lambda \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \mathbf{K}_{\text{NTK}}^{-1} \Gamma^{-1} \mathbf{y} + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \mathbf{K}_{\text{NTK}}^{-1} \Gamma^{-1} \Omega \quad (3.50)$$

where

$$\Gamma = \hat{\mathbf{A}} + \lambda \mathbb{I} + \mathbf{K}_{\text{NTK}}^{-1} \nabla_{\Theta} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\Theta} f_0(\mathbf{X})^T \mathbf{K}_{\text{NTK}}^{-1} \quad (3.51)$$

$$\Omega = \mathbf{K}_{\text{NTK}}^{-1} \nabla_{\Theta} f_0(\mathbf{X}) \hat{\mathbf{B}} \Theta_0 + (\hat{\mathbf{A}} + \lambda \mathbb{I}) f_0(\mathbf{X}) \quad (3.52)$$

$$\mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) = [0, \dots, 0, \underbrace{\omega_1^k, \dots, \omega_{n_k}^k}_{\text{Within domain } k}, 0, \dots, 0] \quad (3.53)$$

and $\mathbf{K}_{\text{NTK}} = \text{diag}(\mathbf{K}_{11}, \mathbf{K}_{22}, \dots, \mathbf{K}_{KK})$. $\mathbf{K}_{kk}$ is a neural tangent kernel [91, 224] matrix within domain k , i.e., its entry is given by $[\mathbf{K}_{kk}]_{ij} = \langle \nabla_{\theta_k} f_0(x_{ik}; \theta_k), \nabla_{\theta_k} f_0(x_{jk}; \theta_k) \rangle$. $\omega_i^k = \langle \nabla_{\theta_k} f_0(x_{*k}; \theta_k), \nabla_{\theta_k} f_0(x_{ik}; \theta_k) \rangle$.

Proof. Following [240], we consider the following linearized neural network

$$f_t^{\text{lin}}(x) = f_0(x) + (\nabla_{\theta_0} f_0(x))^T (w_t) \quad (3.54)$$

where $w_t = \theta_t - \theta_0$ is the parameter change from the initial values. Let $\mathbf{W}_t = \Theta_t - \Theta_0$ be the change of parameters from the initial values. Based on continuous time gradient

descent [240], the evolution of the parameters can be expressed as

$$\begin{aligned}
\dot{\mathbf{W}}_t &= -\frac{\eta}{2} \nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta}) \\
&= -\frac{\eta}{2} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \nabla_{f_t^{\text{lin}}(\mathbf{X})} \left(f_t^{\text{lin}}(\mathbf{X})^T \hat{\mathbf{A}} f_t^{\text{lin}}(\mathbf{X}) + \lambda (f_t^{\text{lin}}(\mathbf{X}) - \mathbf{y})^T (f_t^{\text{lin}}(\mathbf{X}) - \mathbf{y}) \right) - \eta \left(2\hat{\mathbf{B}}\boldsymbol{\Theta}_t \right) \\
&= -\frac{\eta}{2} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\hat{\mathbf{A}} f_t^{\text{lin}}(\mathbf{X}) + \lambda (f_t^{\text{lin}}(\mathbf{X}) - \mathbf{y}) \right) - \eta \hat{\mathbf{B}}\boldsymbol{\Theta}_t \\
&= -\eta \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\hat{\mathbf{A}} f_0(\mathbf{X}) + \hat{\mathbf{A}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \mathbf{W}_t + \lambda (f_0(\mathbf{X}) + \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \mathbf{W}_t - \mathbf{y}) \right) - \eta \hat{\mathbf{B}}(\mathbf{W}_t + \boldsymbol{\Theta}_0) \\
&= -\eta \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\left(\hat{\mathbf{A}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) + \lambda \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \right) \mathbf{W}_t + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) - \lambda \mathbf{y} \right) - \eta \hat{\mathbf{B}}(\mathbf{W}_t + \boldsymbol{\Theta}_0) \\
&= -\eta \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\left(\hat{\mathbf{A}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) + \lambda \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \right) \mathbf{W}_t + (\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T)^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \mathbf{W}_t \right. \\
&\quad \left. + (\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T)^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) - \lambda \mathbf{y} \right)
\end{aligned} \tag{3.55}$$

In this case, the ODE has a closed-form solution below.

$$\begin{aligned}
\boldsymbol{\Theta}_t &= -\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\hat{\mathbf{A}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T + \lambda \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right. \\
&\quad \left. + (\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T)^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbb{I} - \exp \left\{ -\eta \left(\hat{\mathbf{A}} \mathbf{K}_{\text{NTK}} + \lambda \mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right) t \right\} \right) \\
&\quad \cdot \left((\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T)^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) - \lambda \mathbf{y} \right) + \boldsymbol{\Theta}_0 \\
&= -\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\hat{\mathbf{A}} \mathbf{K}_{\text{NTK}} + \lambda \mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbb{I} - \exp \left\{ -\eta \left(\hat{\mathbf{A}} \mathbf{K}_{\text{NTK}} + \lambda \mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right) t \right\} \right) \\
&\quad \cdot \left(\mathbf{K}_{\text{NTK}}^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) - \lambda \mathbf{y} \right) + \boldsymbol{\Theta}_0
\end{aligned} \tag{3.56}$$

Thus,

$$\begin{aligned}
\lim_{t \rightarrow \infty} \boldsymbol{\Theta}_t &= -\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \left(\hat{\mathbf{A}} \mathbf{K}_{\text{NTK}} + \lambda \mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbf{K}_{\text{NTK}}^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) - \lambda \mathbf{y} \right) + \boldsymbol{\Theta}_0
\end{aligned} \tag{3.57}$$

For an arbitrary point x_{*k} , the predicted value is given by

$$\begin{aligned}
& f_t^{\text{lin}}(x_{*k}; \theta_k) \\
&= \lambda \cdot \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbb{I} - \exp \left\{ -\eta \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right) \right\} \right) \mathbf{y} \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbb{I} - \exp \left\{ -\eta \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right) \right\} \right) \\
&\quad \cdot \left(\mathbf{K}_{\text{NTK}}^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) \right)
\end{aligned} \tag{3.58}$$

Thus, it holds that

$$\begin{aligned}
& \lim_{t \rightarrow \infty} f_t(x_{*k}; \theta_k) = \lim_{t \rightarrow \infty} f_t^{\text{lin}}(x_{*k}; \theta_k) \\
&= \lambda \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \mathbf{y} \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \left(\nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X})^T \right) \right)^{-1} \\
&\quad \cdot \left(\mathbf{K}_{\text{NTK}}^{-1} \nabla_{\boldsymbol{\Theta}} f_0(\mathbf{X}) \hat{\mathbf{B}} \boldsymbol{\Theta}_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) \right)
\end{aligned} \tag{3.59}$$

which completes the proof. QED.

Furthermore, the following corollary shows the connection between the proposed TENON framework and standard label propagation [237, 238].

Corollary 3.4. In the special case where $d_{kk'} = 0$ ($k \neq k'$), with the same conditions as Theorem 3.3, for any $k \in \{1, \dots, K\}$, the predicted values of $f(\cdot; \theta_k)$ in Eq. (3.37) over the training samples $\mathbf{X}_k$ within domain k converge to

$$\lim_{t \rightarrow \infty} f_t(\mathbf{X}_k; \theta_k) = (1 - \alpha) (\mathbb{I} - \alpha \mathbf{A}_k)^{-1} \mathbf{y}_k \tag{3.60}$$

where $f_t(\mathbf{X}_k; \theta_k) = [f_t(x_{1k}; \theta_k), f_t(x_{2k}; \theta_k), \dots, f_t(x_{n_k k}; \theta_k)]^T$, $\mathbf{y}_k = [y_{1k}, y_{2k}, \dots, y_{n_k k}]^T$, and $\alpha = \frac{1}{\lambda+1}$. Furthermore, for any testing sample x_{*k} , it holds

$$\begin{aligned}
\lim_{t \rightarrow \infty} f_t(x_{*k}; \theta_k) &= (1 - \alpha) \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} (\mathbb{I} - \alpha \mathbf{A}_k)^{-1} \mathbf{y}_k \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} f_0(\mathbf{X}_k)
\end{aligned} \tag{3.61}$$

Proof. In this case, it holds that $\hat{\mathbf{B}} = \mathbf{0}$. Then for any sample x_{*k} , it holds

$$\begin{aligned}
& f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} + \mathbf{K}_{\text{NTK}}^{-1} \nabla_{\Theta} f_0(\mathbf{X}) \hat{\mathbf{B}} \nabla_{\Theta} f_0(\mathbf{X})^T \right)^{-1} \\
& \quad \cdot \left(\mathbf{K}_{\text{NTK}}^{-1} \nabla_{\Theta} f_0(\mathbf{X}) \hat{\mathbf{B}} \Theta_0 + \hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) \right) \\
&= f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} \right)^{-1} \left(\hat{\mathbf{A}} f_0(\mathbf{X}) + \lambda f_0(\mathbf{X}) \right) \\
&= f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \mathbf{K}_{\text{NTK}}^{-1} f_0(\mathbf{X}) \\
&= f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} f_0(\mathbf{X}_k)
\end{aligned} \tag{3.62}$$

Thus,

$$\begin{aligned}
\lim_{t \rightarrow \infty} f_t(x_{*k}; \theta_k) &= \lambda \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}) \left(\hat{\mathbf{A}}\mathbf{K}_{\text{NTK}} + \lambda\mathbf{K}_{\text{NTK}} \right)^{-1} \mathbf{y} \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} f_0(\mathbf{X}_k) \\
&= \lambda [\mathbf{0}, \dots, \mathbf{0}, \mathbf{K}(x_{*k}, \mathbf{X}_k), \mathbf{0}, \dots, \mathbf{0}] \left(\left(\hat{\mathbf{A}} + \lambda \mathbb{I} \right) \mathbf{K}_{\text{NTK}} \right)^{-1} \mathbf{y} \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} f_0(\mathbf{X}_k) \\
&= \lambda \mathbf{K}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} (\lambda \mathbb{I} + \mathbb{I} - \mathbf{A}_k)^{-1} \mathbf{y}_k \\
&\quad + f_0(x_{*k}; \theta_k) - \mathbf{K}_{\text{NTK}}(x_{*k}, \mathbf{X}_k) \mathbf{K}_{kk}^{-1} f_0(\mathbf{X}_k)
\end{aligned} \tag{3.63}$$

For training samples $\mathbf{X}_k$ within domain k , the following holds

$$\lim_{t \rightarrow \infty} f_t(\mathbf{X}_k; \theta_k) = \frac{\lambda}{\lambda + 1} \left(\mathbb{I} - \frac{1}{\lambda + 1} \mathbf{A}_k \right)^{-1} \mathbf{y}_k \tag{3.64}$$

which completes the proof. QED.

It can be seen from Corollary 3.4 that when all domains are irrelevant (i.e., $d_{kk'} = 0$ for any $k \neq k'$), the objective of TENON is equivalent to standard label propagation [237, 238] on each domain and no knowledge is shared across domains. Furthermore, previous label propagation approaches focus on transductive semi-supervised learning, where labels are inferred for a set of unlabeled training samples (shown in Eq. (3.60)), whereas Corollary 3.4 provides a feasible solution for inductive semi-supervised learning, where the labels can be inferred for new unseen testing samples (shown in Eq. (3.61)).

3.2.4 Instantiated Algorithms

We provide two instantiated algorithms of TENON for domain adaptation (TENON-DA) and out-of-distribution generalization (TENON-ODG). The crucial idea is to formulate domain adaptation and out-of-distribution generalization as the transductive network of networks and the inductive network of networks [236], respectively.

Multi-Source Domain Adaptation. We study the problem of multi-source domain adaptation via a network of networks. As shown in Figure 3.2(a), each domain (source or target domain) is formulated as a node in the main network, and samples within each domain form a domain-specific network. Thus, domain adaptation aims to propagate the label information (1) from source domains to the target domain (domain-level propagation) and (2) from labeled samples to unlabeled samples (sample-level propagation). To this end, we instantiate the proposed TENON framework (denoted as TENON-DA) below.

Given $K-1$ source domains $\{\mathcal{D}_k^s\}_{k=1}^{K-1}$ each with labeled training samples $\{x_{ik}, y_{ik}\}_{i=1}^{n_k^s} \sim \mathbb{P}_k^s$, and a target domain $\mathcal{D}^t$ with only unlabeled training samples $\{x_j^t\}_{j=1}^{n^t} \sim \mathbb{P}_X^t$, the objective function of TENON-DA is directly given by Eq. (3.37). Here the class label y_{ik} ($k = 1, \dots, K-1$) of source training sample x_{ik} is represented as a one-hot vector, and the pseudo label y_i^t of unlabeled target training sample x_i^t is initialized as a zero vector. We can represent $x_j^t = x_{jK}$ and $y_j^t = y_{jK}$ in the objective function of Eq. (3.37). Based on Theorem 3.3, we can obtain the closed-form solution of TENON-DA as follows. Suppose $f_0(\cdot; \Theta) = 0$, $\Theta_0 = \mathbf{0}$, the predicted class labels of target training samples are given by

$$\mathbf{y}_{\text{pred}}^t = [\mathbf{y}_{\text{pred}}]_{-n^t}; \quad \text{where} \quad \mathbf{y}_{\text{pred}} = \lambda \mathbf{K}_{\text{NTK}}^{-1} \mathbf{\Gamma}^{-1} \mathbf{y} \quad (3.65)$$

where $[\mathbf{y}_{\text{pred}}]_{-n^t}$ denotes the last n^t rows of predicted output values $\mathbf{y}^*$. Moreover, for any new target testing sample x_*^t , the predicted class label via TENON-DA is

$$y_*^t = \lambda \mathbf{K}_{\text{NTK}}(x_*^t, \mathbf{X}) \mathbf{K}_{\text{NTK}}^{-1} \mathbf{\Gamma}^{-1} \mathbf{y} = \mathbf{K}(x_*^t, \mathbf{X}^t) \mathbf{y}_{\text{pred}}^t \quad (3.66)$$

where $\mathbf{K}(x_*^t, \mathbf{X}^t) = [\mathbf{K}_{\text{NTK}}(x_*^t, x_1^t), \dots, \mathbf{K}_{\text{NTK}}(x_*^t, x_{n^t}^t)]$. TENON-DA is a non-parametric multi-source domain adaptation approach. As illustrated in Algorithm 3.1, we can pre-compute the propagated labels $\mathbf{y}_{\text{pred}}^t$ for unlabeled target training samples. Then, the class label of any testing target sample is inferred using the propagated labels $\mathbf{y}_{\text{pred}}^t$ and the neural tangent kernel $\mathbf{K}(x_*^t, \mathbf{X}^t)$ between this testing sample and the target training samples.

Out-of-distribution Generalization. We study the out-of-distribution generalization [35] via an inductive network of networks. As illustrated in Figure 3.2(b), all source (training) domains can be used to construct a network of networks. Since the target (testing) domains

Algorithm 3.1: TENON-DA

Input: $(K - 1)$ source domains $\{\mathcal{D}_k^s\}_{k=1}^{K-1}$, a target domain $\mathcal{D}^t$;

Output: Predicted output values of target samples.

———— Training Stage (Pre-computing) —————

Calculate all sample similarity s_{ij}^k and domain similarity $d_{kk'}$;

foreach $k = 1, \dots, K$ **do**

 Calculate block neural tangent kernel $\mathbf{K}_{kk}$;

 Calculate inverse matrix $\mathbf{K}_{kk}^{-1}$;

foreach $k' = k + 1, \dots, K$ **do**

 Calculate block neural tangent kernel $\mathbf{K}_{kk'}$;

end

end

Calculate $\mathbf{\Gamma K}_{\text{NTK}}$;

Calculate $\mathbf{y}_{\text{pred}} = \lambda \mathbf{K}_{\text{NTK}}^{-1} \mathbf{\Gamma}^{-1} \mathbf{y} = \lambda (\mathbf{\Gamma K}_{\text{NTK}})^{-1} \mathbf{y}$;

Obtain target propagated labels $\mathbf{y}_{\text{pred}}^t = [\mathbf{y}_{\text{pred}}]_{-n^t}$;

———— Inference Stage —————

foreach x_*^t from target domain **do**

 Calculate neural tangent kernel $\mathbf{K}(x_*^t, \mathbf{X}^t) = [\mathbf{K}_{\text{NTK}}(x_*^t, x_1^t), \dots, \mathbf{K}_{\text{NTK}}(x_*^t, x_{n^t}^t)]$;

 Calculate $y_*^t = \mathbf{K}(x_*^t, \mathbf{X}^t) \mathbf{y}_{\text{pred}}^t$.

end

Algorithm 3.2: TENON-OOD

Input: K source (training) domains $\{\mathcal{D}_k^s\}_{k=1}^K$;

Output: Predicted output values of target samples.

———— Training Stage (Pre-computing) —————

Calculate $\mathbf{\Gamma K}_{\text{NTK}}$ (same procedures as Algorithm 3.1);

Calculate $\mathbf{y}_{\text{pred}} = \lambda \mathbf{K}_{\text{NTK}}^{-1} \mathbf{\Gamma}^{-1} \mathbf{y} = \lambda (\mathbf{\Gamma K}_{\text{NTK}})^{-1} \mathbf{y}$;

———— Inference Stage —————

foreach x_*^t from target (testing) domain **do**

 Calculate neural tangent kernel $\mathbf{\Phi}(x_*^t, \mathbf{X})$;

 Calculate $y_*^t = \frac{1}{K} \mathbf{\Phi}(x_*^t, \mathbf{X}) \mathbf{y}_{\text{pred}}$.

end

are only available during the testing phase, they will be added to the main network as new nodes after model training. Therefore, out-of-distribution generalization can be considered as an inductive distributional learning problem, given the formulated network of networks. To solve this problem, we instantiate the proposed TENON framework (denoted as TENON-OOD) with the following training and inference stages (see Algorithm 3.2).

- *Training Stage:* Given K source (training) domains $\{\mathcal{D}_k^s\}_{k=1}^K$ each with labeled training samples $\{x_{ik}, y_{ik}\}_{i=1}^{n_k^s} \sim \mathbb{P}_k^s$, the objective function of TENON-OOD during training can be

directly given by Eq. (3.37). Thus, based on Theorem 3.3, we obtain the closed-form solution for model parameters $\{\theta_k\}_{k=1}^K$ over training domains.

$$\Theta^* = \lambda \nabla_{\Theta} f_0(\mathbf{X})^T \mathbf{K}_{\text{NTK}}^{-1} \Gamma^{-1} \mathbf{y} \quad (3.67)$$

where $\Theta^* = [\theta_1^{*T}, \dots, \theta_K^{*T}]^T$. Here θ_k^* denotes the optimized model parameters within domain k .

- *Inference Stage:* In the inference stage, we can learn the model parameters θ_{K+1}^* for a new target (testing) domain $\mathcal{D}^t$ as follows. For standard out-of-distribution generalization, no prior knowledge regarding the target (testing) domain is available before model inference. In this case, we assume that the new target (testing) domain can be considered as a new (domain) node for the previously derived network of networks. The edge weight⁶ between this new node and previous nodes within the main network is set as 1. Considering the objective function $\min_{\theta_{K+1}} \sum_{k,k'=1}^{K+1} d_{kk'} \left\| \frac{\theta_k}{\sqrt{M_{kk}}} - \frac{\theta_{k'}}{\sqrt{M_{k'k'}}} \right\|_2^2$, we obtain the closed-form solution $\theta_{K+1}^* = \frac{1}{K} \sum_{k=1}^K \theta_k^*$, and thus the predicted class label of any testing sample x_*^t is given by

$$y_*^t = \frac{\lambda}{K} \Phi(x_*^t, \mathbf{X}) \mathbf{K}_{\text{NTK}}^{-1} \Gamma^{-1} \mathbf{y} \quad (3.68)$$

$$\Phi(x_*^t, \mathbf{X}) = [\mathbf{K}_{\text{NTK}}(x_*^t, x_{11}^s), \dots, \mathbf{K}_{\text{NTK}}(x_*^t, x_{n_{s1}}^s), \dots, \mathbf{K}_{\text{NTK}}(x_*^t, x_{1K}^s), \dots, \mathbf{K}_{\text{NTK}}(x_*^t, x_{n_{sK}}^s)] \quad (3.69)$$

It can be seen from Algorithms 3.1 & 3.2 that the time complexity of **TENON-DA** and **TENON-OOD** is determined by the calculation of neural tangent kernel (NTK) of any pair of training samples and the inversion of the propagation matrix $\Gamma \mathbf{K}_{\text{NTK}}$. The time complexity of calculating NTK over all domains is $\mathcal{O}(\bar{n}^2)$ [241], where $\bar{n} = \sum_{k=1}^K n_k^s$ denotes the number of all training samples. The inversion of $\Gamma \mathbf{K}_{\text{NTK}}$ requires $\mathcal{O}(\bar{n}^3)$. Following [235], we can use the conjugate gradient method to solve the linear system $(\Gamma \mathbf{K}_{\text{NTK}}) \mathbf{y}^* = \lambda \mathbf{y}$, in order to estimate the propagated labels $\mathbf{y}^*$. This allows us to reduce the time complexity from $\mathcal{O}(\bar{n}^3)$ to $\mathcal{O}(b\bar{n}^2)$, where b is the number of iterations. In this case, we term the variants of **TENON-DA** and **TENON-OOD** algorithms with conjugate gradient as **TENON-DA-Fast** and **TENON-TENON-OOD-Fast**, respectively.

⁶Without prior knowledge regarding the unseen target domain, we assume that the unseen target domain is equally similar to all source domains. In this case, only the parameter smoothness regularization (i.e., the third term in Eq. (3.37)) will be available to optimize the model parameters of this unseen target domain.

3.2.5 Experiment

We provide the experimental results to verify the effectiveness of the proposed TENON framework for multi-source domain adaptation and out-of-distribution generalization.

Data Sets. In the experiments, we use the following data sets.

- Amazon Review [228]: It contains positive and negative product reviews from four different domains: Books, DVD, Electronics, and Kitchen. Following [112, 124], we use top-5000 frequent unigrams/bigrams to extract the bag-of-words features for Amazon reviews. Each review is associated with a binary label indicating positive or negative sentiment.
- CityCam [242]: CityCam is a large-scale web camera data set. It contains images captured by several cameras in different city locations. Following [220], we use images from four cameras (with IDs: 253, 495, 511, and 572). Each image has a 2048-dimensional feature vector extracted from the pre-trained ResNet50 [69]. Specifically, we consider a binary classification task based on the number of vehicles within the camera images, i.e., whether there are at least 10 cars in an image.
- HuffPost [243]: HuffPost contains news headlines associated with news categories collected from the Huffington Post from 2012 to 2018. Following [244], we use pre-trained DistilBERT [245] to extract a 768-dimensional feature vector for each new headline. The task is to identify the news tags of article headlines.
- ArXiv [246]: ArXiv provides metadata of arXiv preprints from 2007 to 2023. As illustrated in [244], each preprint consists of a paper title and its corresponding primary categories. The paper title can further be represented as a 768-dimensional feature vector using pre-trained DistilBERT [245]. The task of ArXiv is to predict the primary category of arXiv pre-prints from their paper titles.
- CivilComments [12]: CivilComments contains 8 demographic identities: male, female, LGBTQ, Christian, Muslim, other religions, Black, or White. Each identity is considered as a single domain. The task is to predict whether a text comment is toxic.

Baselines. We use the following domain adaptation baselines, including (1) semi-supervised learning: LabelProp [237, 247], and (2) Domain adaptation: DANN [88], MDAN [124], M3SAD [49], DARN [112], and GRDA [248]. In addition, we use the following out-of-distribution generalization baselines: ERM, DANN [88], IRM [230], SD [249], Fish [250], and EQRM [251].

Model	Amazon Review				CityCam			
	Books	DVD	Electronics	Kitchen	253	495	511	572
LabelProp	0.7078 $\pm$ 0.0040	0.7294 $\pm$ 0.0093	0.7699 $\pm$ 0.0079	0.7799 $\pm$ 0.0082	0.6741 $\pm$ 0.1360	0.6312 $\pm$ 0.0470	0.6753 $\pm$ 0.0888	0.7560 $\pm$ 0.0564
DANN	0.6958 $\pm$ 0.0157	0.7229 $\pm$ 0.0031	0.7818$\pm$0.0053	0.7879 $\pm$ 0.0072	0.7804 $\pm$ 0.0415	0.6716 $\pm$ 0.0499	0.8498$\pm$0.0136	0.7563 $\pm$ 0.0344
MDAN	0.7196 $\pm$ 0.0095	0.7432 $\pm$ 0.0205	0.7744 $\pm$ 0.0121	0.7869 $\pm$ 0.0156	0.8007 $\pm$ 0.0579	0.6685 $\pm$ 0.0339	0.8222 $\pm$ 0.0227	0.7280 $\pm$ 0.0261
M3SAD	0.7019 $\pm$ 0.0232	0.7251 $\pm$ 0.0210	0.7753 $\pm$ 0.0117	0.7893 $\pm$ 0.0134	0.8064 $\pm$ 0.0581	0.6175 $\pm$ 0.0286	0.7970 $\pm$ 0.0357	0.7621 $\pm$ 0.0531
DARN	0.7175 $\pm$ 0.0126	0.7412 $\pm$ 0.0180	0.7703 $\pm$ 0.0119	0.7888 $\pm$ 0.0145	0.8243 $\pm$ 0.0392	0.6795 $\pm$ 0.0321	0.8271 $\pm$ 0.0161	0.7547 $\pm$ 0.0385
GRDA	0.7110 $\pm$ 0.0099	0.7294 $\pm$ 0.0110	0.7714 $\pm$ 0.0091	0.7884 $\pm$ 0.0056	0.7949 $\pm$ 0.0751	0.6698 $\pm$ 0.0340	0.8254 $\pm$ 0.0196	0.7495 $\pm$ 0.0389
TENON-DA-Fast	0.7241$\pm$0.0146	0.7499 $\pm$ 0.0101	0.7713 $\pm$ 0.0093	0.7898$\pm$0.0101	0.8359$\pm$0.0146	0.7145$\pm$0.0122	0.7918 $\pm$ 0.0104	0.7857 $\pm$ 0.0176
TENON-DA	0.7238 $\pm$ 0.0135	0.7503$\pm$0.0094	0.7763 $\pm$ 0.0065	0.7851 $\pm$ 0.0037	0.8350 $\pm$ 0.0156	0.7143 $\pm$ 0.0134	0.7932 $\pm$ 0.0097	0.7859$\pm$0.0182

Table 3.6: Multi-source domain adaptation on Amazon review and CityCam data sets

Model	2013	2014	2015	2016	2017	2018	Avg.
LabelProp	0.5312 $\pm$ 0.0109	0.2942 $\pm$ 0.0095	0.2641 $\pm$ 0.0202	0.3535 $\pm$ 0.0237	0.3900 $\pm$ 0.0145	0.5055 $\pm$ 0.0165	0.3897
DANN	0.4171 $\pm$ 0.0293	0.3510 $\pm$ 0.0234	0.3847 $\pm$ 0.0423	0.4468 $\pm$ 0.0205	0.4568 $\pm$ 0.0352	0.5490 $\pm$ 0.0254	0.4342
MDAN	0.4171 $\pm$ 0.0293	0.3365 $\pm$ 0.0481	0.3757 $\pm$ 0.0546	0.4424 $\pm$ 0.0207	0.4392 $\pm$ 0.0267	0.5019 $\pm$ 0.0322	0.4188
M3SAD	0.4077 $\pm$ 0.0290	0.3490 $\pm$ 0.0439	0.4055 $\pm$ 0.0380	0.4468 $\pm$ 0.0246	0.4455 $\pm$ 0.0281	0.4879 $\pm$ 0.0398	0.4237
DARN	0.4171 $\pm$ 0.0293	0.3944 $\pm$ 0.0140	0.3902 $\pm$ 0.0375	0.4553 $\pm$ 0.0205	0.4951$\pm$0.0227	0.5601 $\pm$ 0.0106	0.4520
GRDA	0.4324 $\pm$ 0.0330	0.3671 $\pm$ 0.0234	0.3539 $\pm$ 0.0255	0.4534 $\pm$ 0.0150	0.4520 $\pm$ 0.0153	0.4987 $\pm$ 0.0137	0.4262
TENON-DA-Fast	0.5850 $\pm$ 0.0110	0.5028$\pm$0.0183	0.4573 $\pm$ 0.0275	0.4995$\pm$0.0129	0.4556 $\pm$ 0.0295	0.5201$\pm$0.0080	0.5033
TENON-DA	0.5851$\pm$0.0110	0.5028$\pm$0.0183	0.4575$\pm$0.0273	0.4987 $\pm$ 0.0131	0.4651 $\pm$ 0.0132	0.5198 $\pm$ 0.0080	0.5048

Table 3.7: Multi-source domain adaptation on the Hoffpost data set (“Avg.” indicates the average accuracy over all target domains)

Model	2009	2011	2013	2015	2017	2019	2021	Avg.
LabelProp	0.7006 $\pm$ 0.0267	0.6938 $\pm$ 0.0060	0.7186 $\pm$ 0.0063	0.6780 $\pm$ 0.0056	0.6717 $\pm$ 0.0130	0.6857 $\pm$ 0.0085	0.6741 $\pm$ 0.0099	0.6889
DANN	0.7483 $\pm$ 0.0122	0.6943 $\pm$ 0.0387	0.7187 $\pm$ 0.0304	0.7283$\pm$0.0092	0.7183 $\pm$ 0.0081	0.7293 $\pm$ 0.0256	0.7213 $\pm$ 0.0350	0.7226
MDAN	0.7483 $\pm$ 0.0122	0.7161 $\pm$ 0.0075	0.6251 $\pm$ 0.0819	0.6593 $\pm$ 0.0535	0.6748 $\pm$ 0.0254	0.6944 $\pm$ 0.0410	0.7139 $\pm$ 0.0223	0.6903
M3SAD	0.5533 $\pm$ 0.0099	0.6026 $\pm$ 0.1134	0.6845 $\pm$ 0.0637	0.7151 $\pm$ 0.0200	0.6980 $\pm$ 0.0176	0.7384 $\pm$ 0.0117	0.7473 $\pm$ 0.0174	0.6770
DARN	0.7483 $\pm$ 0.0122	0.7228$\pm$0.0110	0.7123 $\pm$ 0.0302	0.7177 $\pm$ 0.0196	0.7126 $\pm$ 0.0151	0.7456$\pm$0.0054	0.7471 $\pm$ 0.0140	0.7295
GRDA	0.7414 $\pm$ 0.0219	0.7186 $\pm$ 0.0041	0.6662 $\pm$ 0.0482	0.6956 $\pm$ 0.0385	0.6777 $\pm$ 0.0215	0.7035 $\pm$ 0.0286	0.7366 $\pm$ 0.0103	0.7056
TENON-DA-Fast	0.7619 $\pm$ 0.0098	0.7161 $\pm$ 0.0064	0.7415 $\pm$ 0.0071	0.7155 $\pm$ 0.0070	0.7216 $\pm$ 0.0034	0.7336 $\pm$ 0.0076	0.7297 $\pm$ 0.0124	0.7314
TENON-DA	0.7621$\pm$0.0100	0.7157 $\pm$ 0.0063	0.7420$\pm$0.0067	0.7195 $\pm$ 0.0053	0.7241$\pm$0.0014	0.7403 $\pm$ 0.0078	0.7477$\pm$0.0096	0.7359

Table 3.8: Multi-source domain adaptation on the ArXiv data set (“Avg.” indicates the average accuracy over all target domains)

Model Configuration. Following [112], we use a 3-layer multi-layer perceptron (MLP) to instantiate the prediction function for all baselines. Then we implement our proposed algorithms using the NTK [252] induced by a 3-layer MLP with infinite width. The classification accuracy is used as the evaluation metric in the experiments. In addition, we set $\sigma = 2, \lambda = 1$ in our experiments.

Main Results. Tables 3.6-3.8 provide the performance comparison between TENON-DA and baselines on various data sets (the best results are indicated in bold). All the experiments are repeated five times and then we report the mean and standard deviation of classification accuracies. For each run, we randomly select 200 samples from each domain as the training

Model	Male	Female	LGBTQ	Christian	Muslim	Others	Black	White	Avg
ERM	0.6859 $\pm$ 0.0091	0.6428 $\pm$ 0.0186	0.6796 $\pm$ 0.0226	0.7058 $\pm$ 0.0248	0.7396 $\pm$ 0.0258	0.7024 $\pm$ 0.0074	0.7118 $\pm$ 0.0169	0.6796 $\pm$ 0.0137	0.6934
DANN	0.6850 $\pm$ 0.0096	0.6453 $\pm$ 0.0249	0.6794 $\pm$ 0.0202	0.7160 $\pm$ 0.0156	0.7340 $\pm$ 0.0207	0.6984 $\pm$ 0.0058	0.6872 $\pm$ 0.0280	0.6776 $\pm$ 0.0145	0.6904
IRM	0.6848 $\pm$ 0.0087	0.6432 $\pm$ 0.0199	0.6862 $\pm$ 0.0186	0.7071 $\pm$ 0.0240	0.7416 $\pm$ 0.0195	0.7028 $\pm$ 0.0083	0.7121 $\pm$ 0.0192	0.6837 $\pm$ 0.0121	0.6951
SD	0.6777 $\pm$ 0.0119	0.6449 $\pm$ 0.0189	0.6847 $\pm$ 0.0216	0.7068 $\pm$ 0.0250	0.7464 $\pm$ 0.0152	0.7031$\pm$0.0088	0.7092 $\pm$ 0.0192	0.6784 $\pm$ 0.0105	0.6939
Fish	0.6882 $\pm$ 0.0055	0.6650 $\pm$ 0.0059	0.6793 $\pm$ 0.0079	0.7363$\pm$0.0195	0.7512 $\pm$ 0.0117	0.6944 $\pm$ 0.0125	0.7219 $\pm$ 0.0071	0.6912$\pm$0.0105	0.7034
EQRM	0.6882 $\pm$ 0.0094	0.6603 $\pm$ 0.0034	0.6830 $\pm$ 0.0176	0.7237 $\pm$ 0.0269	0.7517 $\pm$ 0.0063	0.6969 $\pm$ 0.0083	0.7269 $\pm$ 0.0067	0.6899 $\pm$ 0.0096	0.7025
TENON-ODD-Fast	0.6922$\pm$0.0062	0.6678 $\pm$ 0.0045	0.6968 $\pm$ 0.0074	0.7236 $\pm$ 0.0059	0.7502 $\pm$ 0.0062	0.7006 $\pm$ 0.0110	0.7037 $\pm$ 0.0096	0.6813 $\pm$ 0.0077	0.7020
TENON-ODD	0.6909 $\pm$ 0.0082	0.6702$\pm$0.0030	0.7044$\pm$0.0063	0.7217 $\pm$ 0.0086	0.7620$\pm$0.0066	0.7022 $\pm$ 0.0089	0.7294$\pm$0.0055	0.6909 $\pm$ 0.0048	0.7089

Table 3.9: Out-of-distribution generalization on CivilComments (“Avg.” indicates the average accuracy over all testing domains)

samples and others as the testing samples. Specifically, for Amazon Review and CityCam data sets, following [112], we take one domain (e.g., “Books”) as the target domain, and others domains (e.g., “DVD”, “Electronics” and “Kitchen”) as source domains. In contrast, Hoffpost and ArXiv data sets [244] contain evolving domains where the data distribution is changing over time. In this case, we take one specific time stamp as the target domain and all historical time stamps as source domains. We have the following observations from Tables 3.6-3.8. (1) LabelProp considers propagating the label information within a single graph. It does not capture the data heterogeneity among different domains, thus leading to sub-optimal performance in domain adaptation. (2) Compared to multi-source domain adaptation baselines, our proposed non-parametric TENON-DA algorithm can achieve superior performance in most cases. This observation verifies the effectiveness of TENON-DA in handling heterogeneous data across domains. (3) TENON-DA-Fast achieves comparable performance with TENON-DA. Furthermore, Figure 3.4(b) shows that TENON-DA-Fast significantly reduces the running time compared to TENON-DA.

Table 3.9 shows the results of TENON-ODD on the CivilComments data set (the best results are indicated in bold). In this case, we take one domain (e.g., “Male”) as the unseen testing target domain and others (e.g., “Female”, “LGBTQ”, “Christian”, “Muslim”, “Others”, “Black”, and “White”) as source training domains. Similarly, it is observed that TENON-ODD outperforms baselines for out-of-distribution generalization.

Ablation Study. Figure 3.4 compares the TENON-DA/TENON-ODD algorithms with their approximation solutions introduced in Subsection 3.2.4. It can be seen that with only 10 iterations, TENON-DA-Fast/TENON-ODD-Fast based on conjugate gradient can efficiently achieve similar performance with their counterparts.

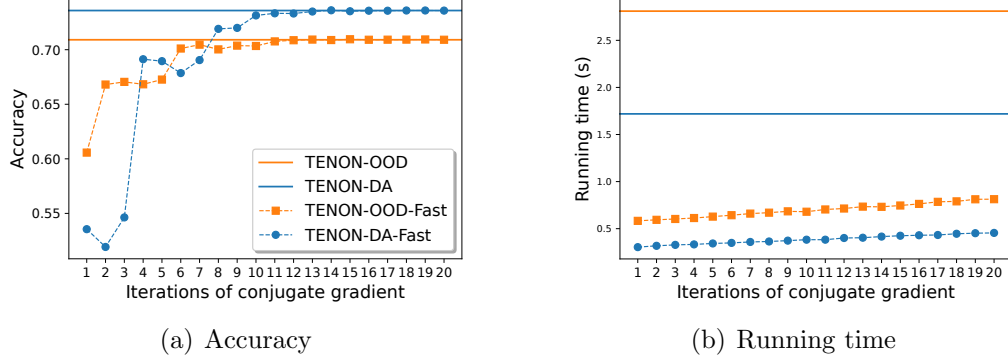


Figure 3.4: Analysis of conjugate gradient

3.3 DYNAMIC TRANSFER LEARNING WITH CLOSED-SET TARGETS

In contrast to Section 3.1 and Section 3.2, this section focuses on studying dynamic transfer learning with closed-set targets [8, 87]. In this problem setting, the data distribution of the source or target domain can change over time, and the source and target domains share the same output space (i.e., the same group of class labels in classification problems).

3.3.1 Problem Definition

Formally, we define the problem of dynamic transfer learning with closed-set targets [8, 87] as follows (shown in Figure 3.5).

Problem 3.4. Given a labeled dynamic source domain $\{\mathcal{D}_k^s\}_{k=1}^N$ and an unlabeled dynamic target domain $\{\mathcal{D}_k^t\}_{k=1}^N$, *dynamic transfer learning with closed-set targets* aims to learn the prediction function for the newest target domain $\mathcal{D}_{N+1}^t$ by leveraging the knowledge from historical source and target domains.

To understand the knowledge transferability across time-evolving domains, we consider the following assumptions. (1) The source and target domains are related at the initial time stamp. (2) The data distributions of both source and target domains are continuously changing over time. (3) The source and target domains have the same input and output spaces for all time stamps. Under these assumptions, we derive the generalization bounds of dynamic transfer learning, followed by a unified meta-learning framework (L2E) [8].

Prior work [40, 253, 254] has studied dynamic transfer learning with a static source domain and a time evolving target domain. These studies demonstrated that the prediction function on the newest target domain can be learned by aggregating the knowledge from the labeled source data and historical unlabeled target data. However, when both source and target

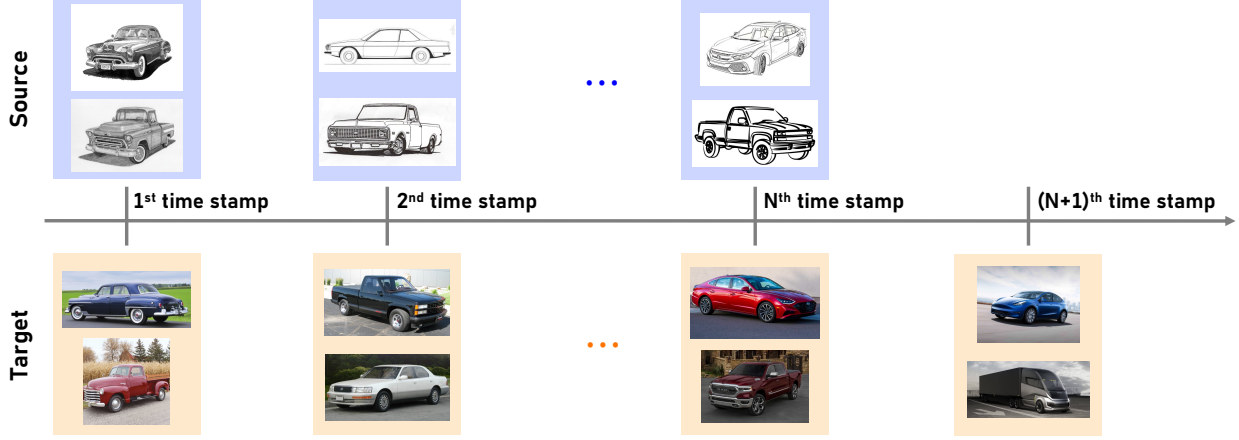


Figure 3.5: Illustration of dynamic transfer learning with closed-set targets. Both source (sketch vehicle images) and target (real-world vehicle images) domains contain two categories of vehicle images: truck and sedan. However, the design styles of vehicles change over time.

domains are evolving over time, prior work might yield sub-optimal solutions due to the under-explored source knowledge.

3.3.2 Theoretical Analysis

We start by deriving the generalization error bound for dynamic transfer learning. Following [38], we consider a binary classification problem with $\mathcal{Y} \in \{0, 1\}$ for simplicity. Before deriving the generalization error bounds, we first introduce several basic concepts below.

Definition 3.1. (L^1 -divergence [38]) The L^1 -divergence between two domains with sampling distributions $\mathbb{P}^s$ and $\mathbb{P}^t$ over $\mathcal{X}$ is defined as follows.

$$d_1(\mathbb{P}^s, \mathbb{P}^t) := 2 \sup_{Q \in \mathcal{Q}} |\Pr_{\mathbb{P}^s}[Q] - \Pr_{\mathbb{P}^t}[Q]| \quad (3.70)$$

where $\mathcal{Q}$ is the set of all the possible measurable subsets under $\mathbb{P}^s$ and $\mathbb{P}^t$.

Definition 3.2. (μ -admissibility [255]) Let $\mathcal{H}$ be the hypothesis class where a hypothesis is a function $h : \mathcal{X} \rightarrow \mathcal{Y}$. A loss function $\mathcal{L}(\cdot, \cdot)$ is μ -admissible if there exists $\mu > 0$ such that for all $x \in \mathcal{X}$, $y, y' \in \mathcal{Y}$ and $h, h' \in \mathcal{H}$, the following inequalities hold.

$$|\mathcal{L}(h'(x), y) - \mathcal{L}(h(x), y)| \leq \mu |h'(x) - h(x)| \quad (3.71)$$

$$|\mathcal{L}(h(x), y') - \mathcal{L}(h(x), y)| \leq \mu |y' - y| \quad (3.72)$$

For dynamic transfer learning, the following theorem states that the expected error of the newest target domain can be bounded in terms of historical source and target knowledge.

Theorem 3.4. Assume that the loss function $\mathcal{L}(\cdot, \cdot)$ is μ -admissible and obeys the triangle inequality. Given a class of functions $\mathcal{H}_{\mathcal{L}} = \{(x, y) \mapsto \mathcal{L}(h(x), y) : h \in \mathcal{H}\}$, for any $\delta > 0$ and $h \in \mathcal{H}$, with probability at least $1 - \delta$, the expected error $\mathcal{E}_{N+1}^t$ for the newest target domain $\mathcal{D}_{N+1}^t$ is bounded by

$$\mathcal{E}_{N+1}^t(h) \leq \frac{1}{2N} \sum_{k=1}^N \left(\hat{\mathcal{E}}_k^s(h) + \hat{\mathcal{E}}_k^t(h) \right) + \frac{N+2}{2} \left(\tilde{d} + \tilde{\lambda} \right) + \tilde{\mathfrak{R}}(\mathcal{H}_{\mathcal{L}}) + \frac{\mu}{N} \sqrt{\frac{\log \frac{1}{\delta}}{2\tilde{m}}} \quad (3.73)$$

where $\tilde{d} = \mu \cdot \max \left\{ \max_{1 \leq k \leq N-1} d_1(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s), d_1(\mathbb{P}_1^s, \mathbb{P}_1^t), \max_{1 \leq k \leq N} d_1(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t) \right\}$, $\tilde{\lambda} = \mu \cdot \max \left\{ \max_{1 \leq k \leq N-1} \lambda_*(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s), \lambda_*(\mathbb{P}_1^s, \mathbb{P}_1^t), \max_{1 \leq k \leq N} \lambda_*(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t) \right\}$ and λ_* measures the difference of the labeling functions across domains and across time stamps, i.e., $\lambda_*(\mathbb{P}_1^s, \mathbb{P}_1^t) = \min \{ \mathbb{E}_{\mathbb{P}_1^s} [|f_1^s(x) - f_1^t(x)|], \mathbb{E}_{\mathbb{P}_1^t} [|f_1^s(x) - f_1^t(x)|] \}$. $\tilde{\mathfrak{R}}(\mathcal{H}_{\mathcal{L}})$ is a Rademacher complexity term and $\tilde{m} = \sum_{k=1}^N (n_k^s + n_k^t)$ is the number of samples from historical source and target domains.

Proof. Following [256], we consider a set of samples $\mathcal{B} = \left(\{(x_{i1}^s, y_{i1}^s)\}_{i=1}^{m_1^s}, \dots, \{(x_{iN}^s, y_{iN}^s)\}_{i=1}^{m_N^s}, \{(x_{i1}^t, y_{i1}^t)\}_{i=1}^{m_1^t}, \dots, \{(x_{iN}^t, y_{iN}^t)\}_{i=1}^{m_N^t} \right)$ drawn from the product distribution $\mathbb{P}(x, y) = \mathbb{P}_1^s(x, y)^{m_1^s} \otimes \dots \otimes \mathbb{P}_N^s(x, y)^{m_N^s} \otimes \mathbb{P}_1^t(x, y)^{m_1^t} \otimes \dots \otimes \mathbb{P}_N^t(x, y)^{m_N^t}$. Then we define a function g over $\mathcal{B}$ as follows.

$$g(\mathcal{B}) = \sup_{h \in \mathcal{H}} \mathcal{E}_{N+1}^t(h) - \frac{1}{2N} \sum_{k=1}^N \left(\hat{\mathcal{E}}_k^s(h) + \hat{\mathcal{E}}_k^t(h) \right) \quad (3.74)$$

where $\hat{\mathcal{E}}_j^s(h) = \frac{1}{m_j^s} \sum_{i=1}^{m_j^s} \mathcal{L}(h(x_{ij}^s), y_{ij}^s)$ and $\hat{\mathcal{E}}_j^t(h) = \frac{1}{m_j^t} \sum_{i=1}^{m_j^t} \mathcal{L}(h(x_{ij}^t), y_{ij}^t)$ for all $j = 1, \dots, N$. Let $\mathcal{B}$ and $\mathcal{B}'$ be two sample sets containing only one different source sample at (x_{ij}^s, y_{ij}^s) and $(\bar{x}_{ij}^s, \bar{y}_{ij}^s)$, then we have

$$|g(\mathcal{B}) - g(\mathcal{B}')| \leq \frac{1}{2m_j^s N} \sup_{h \in \mathcal{H}} |\mathcal{L}(h(x_{ij}^s), y_{ij}^s) - \mathcal{L}(h(\bar{x}_{ij}^s), \bar{y}_{ij}^s)| \leq \frac{2\mu}{2m_j^s N} \leq \frac{\mu}{\tilde{m}N} \quad (3.75)$$

The same result holds for different target samples. Based on McDiarmid's inequality, we have for any $\epsilon > 0$

$$\Pr [g(\mathcal{B}) - \mathbb{E}_{\mathcal{B}} [g(\mathcal{B})] \geq \epsilon] \leq \exp \left(\frac{-2\tilde{m}N^2\epsilon^2}{\mu^2} \right) \quad (3.76)$$

Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds

$$g(\mathcal{B}) \leq \mathbb{E}_{\mathcal{B}} [g(\mathcal{B})] + \frac{\mu}{N} \sqrt{\frac{\log \frac{1}{\delta}}{2\tilde{m}}} \quad (3.77)$$

In addition, for any $h \in \mathcal{H}$, we have

$$\begin{aligned}
\mathcal{E}_1^t(h) &= \mathcal{E}_1^t(h) + \mathcal{E}_1^s(h) - \mathcal{E}_1^s(h) + \mathcal{E}_1^s(h, f_1^t) - \mathcal{E}_1^s(h, f_1^t) \\
&\leq \mathcal{E}_1^s(h) + |\mathcal{E}_1^s(h, f_1^t) - \mathcal{E}_1^s(h, f_1^s)| + |\mathcal{E}_1^t(h, f_1^t) - \mathcal{E}_1^s(h, f_1^t)| \\
&\leq \mathcal{E}_1^s(h) + \mathbb{E}_{\mathbb{P}_1^s} [|\mathcal{L}(h(x_1^s), f_1^t(x_1^s)) - \mathcal{L}(h(x_1^s), f_1^s(x_1^s))|] + |\mathcal{E}_1^t(h, f_1^t) - \mathcal{E}_1^s(h, f_1^t)| \\
&\leq \mathcal{E}_1^s(h) + \mu \mathbb{E}_{\mathbb{P}_1^s} [|f_1^t(x_1^s) - f_1^s(x_1^s)|] + \int |\mathbb{P}_1^t(x) - \mathbb{P}_1^s(x)| |\mathcal{L}(h(x), f_1^t(x))| dx \\
&\leq \mathcal{E}_1^s(h) + \mu \mathbb{E}_{\mathbb{P}_1^s} [|f_1^t(x_1^s) - f_1^s(x_1^s)|] + \mu \int |\mathbb{P}_1^t(x) - \mathbb{P}_1^s(x)| |h(x) - f_1^t(x)| dx \\
&\leq \mathcal{E}_1^s(h) + \mu \mathbb{E}_{\mathbb{P}_1^s} [|f_1^t(x_1^s) - f_1^s(x_1^s)|] + \mu d_1(\mathbb{P}_1^s, \mathbb{P}_1^t)
\end{aligned} \tag{3.78}$$

Then we can obtain

$$\begin{aligned}
\mathcal{E}_1^t(h) &\leq \mathcal{E}_1^s(h) + \mu d_1(\mathbb{P}_1^s, \mathbb{P}_1^t) + \mu \min \{ \mathbb{E}_{\mathbb{P}_1^s} [|f_1^t(x_1^s) - f_1^s(x_1^s)|], \mathbb{E}_{\mathbb{P}_1^t} [|f_1^t(x_1^s) - f_1^s(x_1^s)|] \} \\
&= \mathcal{E}_1^s(h) + \mu d_1(\mathbb{P}_1^s, \mathbb{P}_1^t) + \mu \lambda_*(\mathbb{P}_1^s, \mathbb{P}_1^t)
\end{aligned} \tag{3.79}$$

It can be seen that this holds for the source or target domain across time stamps. Based on the Rademacher bound in Theorem 2 of [111], the following holds

$$\begin{aligned}
\mathbb{E}_{\mathcal{B}} [g(\mathcal{B})] &= \mathbb{E}_{\mathcal{B}} \left[\sup_{h \in \mathcal{H}} \mathcal{E}_{N+1}^t(h) - \frac{1}{2N} \sum_{k=1}^N (\hat{\mathcal{E}}_k^s(h) + \hat{\mathcal{E}}_k^t(h)) \right] \\
&= \frac{1}{2N} \sup_{h \in \mathcal{H}} \left(\sum_{k=1}^N (\mathcal{E}_{N+1}^t(h) - \mathcal{E}_k^t(h)) + \sum_{k=1}^N (\mathcal{E}_{N+1}^t(h) - \mathcal{E}_k^s(h)) \right) \\
&\quad + \mathbb{E}_{\mathcal{B}} \left[\sup_{h \in \mathcal{H}} \frac{1}{2N} \sum_{k=1}^N (\mathcal{E}_k^s(h) - \hat{\mathcal{E}}_k^s(h)) + \frac{1}{2N} \sum_{k=1}^N (\mathcal{E}_k^t(h) - \hat{\mathcal{E}}_k^t(h)) \right] \\
&\leq \frac{1}{2N} \sup_{h \in \mathcal{H}} \left(\sum_{k=1}^N (\mathcal{E}_{N+1}^t(h) - \mathcal{E}_k^t(h)) + \sum_{k=1}^N (\mathcal{E}_1^s(h) - \mathcal{E}_k^s(h)) \right. \\
&\quad \left. + \sum_{k=1}^N (\mathcal{E}_{N+1}^t(h) - \mathcal{E}_1^t(h)) + \sum_{k=1}^N (\mathcal{E}_1^t(h) - \mathcal{E}_1^s(h)) \right) \\
&\quad + \mathbb{E}_{\mathcal{B}} \left[\frac{1}{2N} \sum_{k=1}^N \sup_{h \in \mathcal{H}} (\mathcal{E}_k^s(h) - \hat{\mathcal{E}}_k^s(h)) + \frac{1}{2N} \sum_{k=1}^N \sup_{h \in \mathcal{H}} (\mathcal{E}_k^t(h) - \hat{\mathcal{E}}_k^t(h)) \right] \\
&\leq \frac{1}{2N} \left(\frac{N(N+1)}{2} (\tilde{d} + \tilde{\lambda}) + \frac{N(N-1)}{2} (\tilde{d} + \tilde{\lambda}) + N(\tilde{d} + \tilde{\lambda}) + N(\tilde{d} + \tilde{\lambda}) \right) \\
&\quad + \mathbb{E}_{\mathcal{B}} \left[\frac{1}{2N} \sum_{k=1}^N \mathfrak{R}_{m_k^s}(\mathcal{H}_{\mathcal{L}}) + \frac{1}{2N} \sum_{k=1}^N \mathfrak{R}_{m_k^t}(\mathcal{H}_{\mathcal{L}}) \right] \leq \frac{N+2}{2} (\tilde{d} + \tilde{\lambda}) + \tilde{\mathfrak{R}}(\mathcal{H}_{\mathcal{L}})
\end{aligned} \tag{3.80}$$

where $\mathfrak{R}_n(\mathcal{H}_{\mathcal{L}})$ denotes the Rademacher complexity of the class of functions $\mathcal{H}_{\mathcal{L}} = \{(x, y) \mapsto$

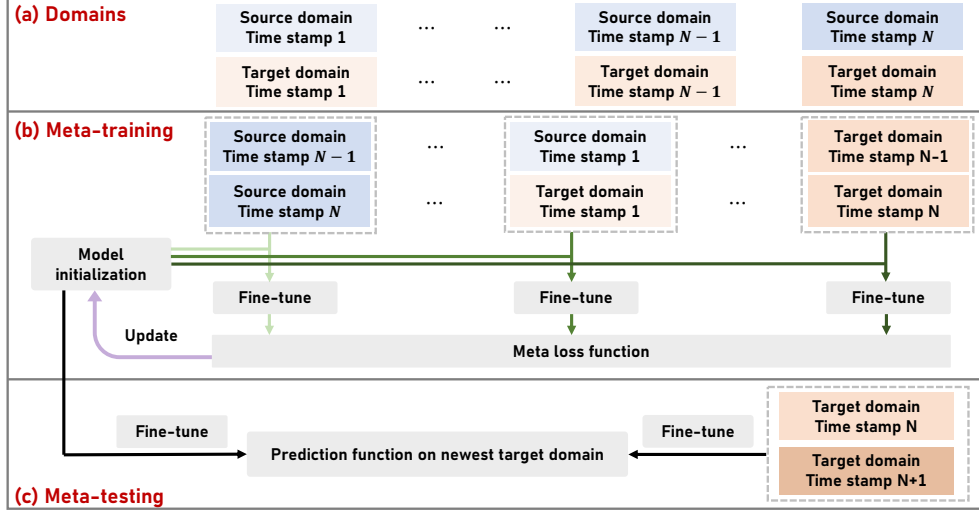


Figure 3.6: Illustration of the proposed L2E framework

$\mathcal{L}(h(x), y) : h \in \mathcal{H}$ over all samples of size n , and $\tilde{\mathfrak{R}}(\mathcal{H}_{\mathcal{L}}) = \frac{1}{2N} \sum_{k=1}^N (\mathfrak{R}_{n_k^s}(\mathcal{H}_{\mathcal{L}}) + \mathfrak{R}_{n_k^t}(\mathcal{H}_{\mathcal{L}}))$. Therefore, for any $h \in \mathcal{H}$,

$$\begin{aligned}
\mathcal{E}_{N+1}(h) &\leq \frac{1}{2N} \sum_{j=1}^N \left(\hat{\mathcal{E}}_k^s(h) + \hat{\mathcal{E}}_k^t(h) \right) + \mathbb{E}_{\mathcal{B}}[g(\mathcal{B})] + \frac{\mu}{N} \sqrt{\frac{\log \frac{1}{\delta}}{2\tilde{m}}} \\
&\leq \frac{1}{2N} \sum_{j=1}^N \left(\hat{\mathcal{E}}_k^s(h) + \hat{\mathcal{E}}_k^t(h) \right) + \frac{N+2}{2} \left(\tilde{d} + \tilde{\lambda} \right) + \tilde{\mathfrak{R}}(\mathcal{H}_{\mathcal{L}}) + \frac{\mu}{N} \sqrt{\frac{\log \frac{1}{\delta}}{2\tilde{m}}}
\end{aligned} \tag{3.81}$$

which completes the proof.

QED.

This theorem reveals that the expected error on the newest target domain is bounded in terms of (i) the empirical errors of historical source and target domains; (ii) the maximum distribution discrepancy across domains and across time stamps; (iii) the maximum labeling difference across domains and across time stamps; and (iv) the average Rademacher complexity of the class of functions $\mathcal{H}_{\mathcal{L}} = \{(x, y) \mapsto \mathcal{L}(h(x), y) : h \in \mathcal{H}\}$. Furthermore, this error bound would become tighter by using the hypothesis-dependent distribution discrepancy measures, e.g., $\mathcal{H}$ -divergence [38], Maximum Mean Discrepancy [115], Margin Disparity Discrepancy [53], f -divergence [31], etc.

3.3.3 A Meta-Learning Framework

We propose a generic meta-learning framework L2E [8] for transferring knowledge from a dynamic source domain to a dynamic target domain. Following the generalization error

bound in Theorem 3.4, we have the following objective function for learning the predictive function of the newest target domain $\mathcal{D}_{N+1}^t$.

$$\min_{\theta} J(\theta) = \sum_{k=1}^N \left(\hat{\mathcal{E}}_k^s(\theta) + \hat{\mathcal{E}}_k^t(\theta) \right) + \gamma \left(d(\mathbb{P}_1^s, \mathbb{P}_1^t; \theta) + \sum_{k=1}^{N-1} d(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s; \theta) + \sum_{k=1}^N d(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t; \theta) \right) \quad (3.82)$$

where $\hat{\mathcal{E}}_k^s(\theta)$ denotes the empirical prediction loss of the source domain at the k^{th} time stamp. We would like to point out that the error bound of Theorem 3.4 is derived in terms of the maximum distribution discrepancy across domains and across time stamps. But there is no prior knowledge regarding the time stamp with the maximum distribution discrepancy. In our framework, we would minimize all the distribution discrepancies across domains and across time stamps. Then, we describe three key components of our meta-learning framework L2E, including constructing meta-pairs of tasks, meta-training, and meta-testing (illustrated in Figure 3.6).

Meta-Pairs of Tasks: With the assumption that the source and target domains are continuously evolving over time, the framework of Eq. (3.82) can be rewritten as a set of pair-wise standard transfer learning tasks.

$$\begin{aligned} \min_{\theta} J(\theta) = & \sum_{k=1}^{N-1} \left(\hat{\mathcal{E}}_{k+1}^s(\theta) + \gamma \cdot d(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s; \theta) \right) + \left(\hat{\mathcal{E}}_1^s(\theta) + \gamma \cdot d(\mathbb{P}_1^s, \mathbb{P}_1^t; \theta) \right) \\ & + \sum_{k=1}^N \left(\hat{\mathcal{E}}_k^t(\theta) + \gamma \cdot d(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t; \theta) \right) \end{aligned} \quad (3.83)$$

This observation motivates us to construct the meta-pair of tasks using consecutive domain data, and learn the optimal model initialization with these meta-pairs of tasks. Different from previous work [40, 254, 257] with only static source domain, we argue that the evolution pattern of the source domain can also help improve the performance of L2E on the newest target domain.

Meta-Training: Let $\zeta_k(\theta) = \hat{\mathcal{E}}_k^t(\theta) + \gamma \cdot d(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t; \theta)$, $\zeta_0(\theta) = \hat{\mathcal{E}}_1^s(\theta) + \gamma \cdot d(\mathbb{P}_1^s, \mathbb{P}_1^t; \theta)$ and $\zeta_{-k}(\theta) = \hat{\mathcal{E}}_{k+1}^s(\theta) + \gamma \cdot d(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s; \theta)$. Then Eq.(3.83) has a simplified expression $\min_{\theta} J(\theta) = \sum_{k=1-N}^N \zeta_k(\theta)$, where ζ_k denotes the objective function of standard transfer learning across domains and across time stamps. Moreover, it can be formulated as a meta-learning problem [258]. To be more specific, we randomly split the training data from every historical source or target domain into one training set and one validation set. Let ζ_k^{tr} (ζ_k^{val}) be the objective function of ζ_k on the training (validation) set. The model initialization $\tilde{\theta}_N^*$

can be learned as follows.

$$\tilde{\theta}_N^* \leftarrow \arg \min_{\theta} \sum_{k=1-N}^{N-1} \zeta_k^{val}(M_k(\theta)) \quad \text{where} \quad M_k(\theta) \leftarrow \theta - \eta \nabla_{\theta} \zeta_k^{tr}(\theta) \quad (3.84)$$

where $M_k : \theta \rightarrow \theta_k$ is a mapping function which learns the optimal domain-specific parameter θ_k from model initialization θ . Following model-agnostic meta-learning (MAML) [258], $M_k(\theta)$ can be instantiated by one or a few gradient descent updates. Here η is the learning rate. In addition, the training samples of the historical target domain are not labeled. Thus, we propose to sequentially learn the predictive function for every historical target task and generate the pseudo-labels of unlabeled examples as follows.

$$\tilde{\theta}_{k-1}^* \leftarrow \arg \min_{\theta} \sum_{j=1-N}^{k-2} \zeta_j^{val}(M_j(\theta)) \quad \text{and} \quad \hat{y}_{ik}^t \leftarrow p\left(y|x_{ik}^t; M_{k-1}(\tilde{\theta}_{k-1}^*)\right) \quad (3.85)$$

where $\hat{y}_{ik}^t$ is the predicted pseudo-label of input example x_{ik}^t from the historical target task $\mathcal{D}_k^t$ ($k = 1, \dots, N$). Notice that the training examples with incorrect pseudo-labels might lead to the accumulation of misclassification errors on the new target tasks over time. To mitigate this issue, we propose to select those examples with high prediction confidence. Specifically, we estimate the entropy of the predicted class probability of target examples, and then choose the top $p\%$ with the lowest entropy values.

Meta-Testing: The optimal parameters θ_{N+1} on the newest target domain $\mathcal{D}_{N+1}^t$ could be obtained as follows.

$$\theta_{N+1} = M_N(\tilde{\theta}_N^*) \leftarrow \tilde{\theta}_N^* - \eta \nabla_{\tilde{\theta}_N^*} \zeta_N(\tilde{\theta}_N^*) \quad (3.86)$$

where $\tilde{\theta}_N^*$ is the optimized model initialization learned in the meta-training phase.

The intuition of this meta-learning framework L2E can be explained as follows. The evolution of dynamic source and target domains can be represented as a sequential knowledge transfer process across time stamps. But from the perspective of transfer learning [2], it would be an asymmetric knowledge transfer process for every time stamp, with the goal of maximizing the prediction performance on the newest domain. That explains why continuous knowledge transfer might have the issue of catastrophic forgetting [253, 254]. However, the continuous evolution of source and target domains indicates that there might exist some common knowledge transferred across all time stamps. For instance, no matter how the fashion of clothes images changes over time, it follows the basic designs (e.g., shape) for different types of clothes. This common knowledge can be captured by the prior model

initialization in our meta-learning framework. It then enables fast adaptation of the newest target domain with only a few model updates.

3.3.4 Experiment

In the experiments, we evaluate the effectiveness of the proposed L2E framework for dynamic transfer learning with closed-set targets.

Data Sets. We used the following three image data sets: Office-31 [259] (with three domains: Amazon (A), Webcam (W) and DSLR (D)), Image-CLEF [123] (with four domains: B, C, I and P) and Caltran [257]. For Office-31 and Image-CLEF, there are 5 time stamps in the source domain and 6 time stamps in the target domain. Caltran contains the real-time images captured by a camera at an intersection for several weeks. Notice that the first two data sets consist of only static domains. Thus, we generate the time evolving domains by gradually adding random noise and rotation to the original images. Take Office-31 (Amazon $\rightarrow$ Webcam) as an example, on one hand, we generate the dynamic labeled source domain $\mathcal{D}_k^s$ ($k = 1, 2, \dots, 5$) at the k^{th} time stamp from Amazon by rotating the original images with degree O_d (i.e., $O_d = -30 \cdot (k - 1)$) and adding the random Gaussian noise. On the other hand, we generate the dynamic unlabeled target domain $\mathcal{D}_k^t$ ($k = 1, 2, \dots, 6$) at the k^{th} time stamp from Webcam by rotating the original images with degree O'_d and adding the random salt&pepper noise with magnitude O'_n , i.e., $O'_d = 15 \cdot (k - 1)$ and $O'_n = 0.05 \cdot (k - 1)$. The goal of dynamic transfer learning is to predict the class labels for the newest target domain $\mathcal{D}_6^t$ using historical source and target knowledge. Caltran is a real-world image data set captured by a camera at an intersection for several weeks, so we consider that one domain consists of the images from a single day. In this case, following [257], we take the first day’s images as the labeled source domain, and others as a dynamic unlabeled target domain.

Baselines. The comparison baselines are given below. (1) Static adaptation: SourceOnly that trains only on the source task, DANN [88], and MDD [53]. (2) Multi-source adaptation: MDAN [124], M3SDA [49], and DARN [112]. (3) Continuous adaptation: CUA [253], TransLATE [39], and GST [40]. Here, we merge all the labeled source data into a large one, and then transfer its knowledge to the newest target task for static adaptation methods. For a fair comparison, all the multi-source and continuous adaptation methods used both historical source and target data for knowledge transfer, and the target selection strategy for choosing high-quality target examples with pseudo-labels.

Model Configuration. We adopted the ResNet-18 [69] pre-trained on ImageNet as the base network for feature extraction with an added 256-dimension bottleneck layer between the *res5c* and *fc* layers. The domain discrepancy $d(\cdot, \cdot)$ is instantiated with JS-divergence [88].

Method	A $\rightarrow$ W		W $\rightarrow$ D		I $\rightarrow$ C		B $\rightarrow$ P	
	Acc	H-Acc	Acc	H-Acc	Acc	H-Acc	Acc	H-Acc
SourceOnly	0.17 $\pm$ 0.02	0.35 $\pm$ 0.02	0.78 $\pm$ 0.02	0.88 $\pm$ 0.03	0.26 $\pm$ 0.01	0.51 $\pm$ 0.01	0.24 $\pm$ 0.00	0.43 $\pm$ 0.01
DANN	0.32 $\pm$ 0.03	0.45 $\pm$ 0.04	0.84 $\pm$ 0.01	0.90 $\pm$ 0.01	0.36 $\pm$ 0.00	0.58 $\pm$ 0.01	0.27 $\pm$ 0.01	0.43 $\pm$ 0.00
MDD	0.32 $\pm$ 0.01	0.45 $\pm$ 0.01	0.85 $\pm$ 0.02	0.91 $\pm$ 0.01	0.41 $\pm$ 0.01	0.62 $\pm$ 0.01	0.28 $\pm$ 0.03	0.42 $\pm$ 0.01
MDAN	0.49 $\pm$ 0.03	0.57 $\pm$ 0.01	0.88 $\pm$ 0.00	0.92 $\pm$ 0.01	0.62 $\pm$ 0.03	0.77 $\pm$ 0.00	0.37 $\pm$ 0.05	0.51 $\pm$ 0.02
M3SDA	0.49 $\pm$ 0.03	0.57 $\pm$ 0.03	0.81 $\pm$ 0.04	0.86 $\pm$ 0.03	0.56 $\pm$ 0.03	0.74 $\pm$ 0.01	0.39 $\pm$ 0.02	0.52 $\pm$ 0.02
DARN	0.45 $\pm$ 0.02	0.49 $\pm$ 0.03	0.69 $\pm$ 0.03	0.74 $\pm$ 0.03	0.55 $\pm$ 0.02	0.76 $\pm$ 0.02	0.39 $\pm$ 0.02	0.52 $\pm$ 0.01
CUA	0.48 $\pm$ 0.01	0.58$\pm$0.01	0.85 $\pm$ 0.03	0.89 $\pm$ 0.04	0.58 $\pm$ 0.01	0.74 $\pm$ 0.01	0.36 $\pm$ 0.03	0.51 $\pm$ 0.00
TransLATE	0.50 $\pm$ 0.01	0.58$\pm$0.02	0.86 $\pm$ 0.02	0.90 $\pm$ 0.02	0.64 $\pm$ 0.01	0.76 $\pm$ 0.00	0.40 $\pm$ 0.03	0.55 $\pm$ 0.01
GST	0.43 $\pm$ 0.01	0.45 $\pm$ 0.00	0.84 $\pm$ 0.02	0.80 $\pm$ 0.01	0.39 $\pm$ 0.01	0.54 $\pm$ 0.03	0.32 $\pm$ 0.01	0.31 $\pm$ 0.02
L2E	0.52$\pm$0.02	0.58$\pm$0.01	0.89$\pm$0.01	0.95$\pm$0.00	0.66$\pm$0.02	0.80$\pm$0.01	0.44$\pm$0.04	0.57$\pm$0.02

Table 3.10: Dynamic transfer learning accuracy on Office-31 and Image-CLEF

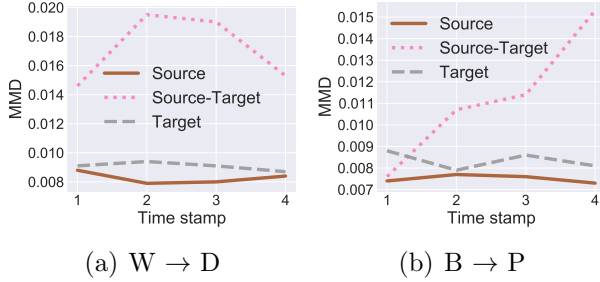


Figure 3.7: Evolvment of domains

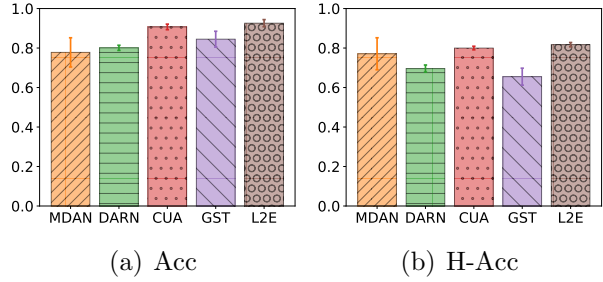


Figure 3.8: Results on Caltran

In addition, we set $\gamma = 0.1$ and $p = 80$ for all the experiments.

Main Results. Figure 3.7 shows the results of domain discrepancy via Maximum Mean Discrepancy (MMD) [115], including (i) the evolution of source domain, i.e., $d(\mathbb{P}_k^s, \mathbb{P}_{k+1}^s)$, (ii) the evolution of target domain, i.e., $d(\mathbb{P}_k^t, \mathbb{P}_{k+1}^t)$, and (iii) the evolution of domain discrepancy, i.e., $d(\mathbb{P}_k^s, \mathbb{P}_k^t)$. This confirms that in both Office-31 and Image-CLEF, the data distributions of source and target domains change smoothly, whereas the relatedness between source and target domains decreases over time. Table 3.10 and Figure 3.8 provide the dynamic transfer learning results where the classification accuracies on the newest target domain (Acc) and all the historical target domains (H-Acc) are reported. We run all the experiments five times and report the mean and standard deviation of classification accuracies (the best results are indicated in bold). It can be observed that: (1) compared to static methods, the multi-source and dynamic transfer learning methods achieve better performance by leveraging the historical knowledge; (2) the proposed L2E framework outperforms state-of-the-art baselines in both the newest target domain and the historical target domains. This indicates that L2E

mitigates the issue of catastrophic forgetting on the historical target domain.

Ablation Study. Table 3.11 reports the results of several variants of L2E on ImageCLEF (B $\rightarrow$ P): (i) L2E w/o source evolution: using only source data at the initial time stamp; (ii) L2E w merged source: merging all source data into a large one; (iii) L2E w/o historical target: transferring the dynamic source domain to the newest target domain directly without historical target knowledge. It is observed that the evolution

Method	Acc	H-Acc
L2E w/o source evolution	0.38 $\pm$ 0.04	0.57 $\pm$ 0.02
L2E w merged source	0.35 $\pm$ 0.07	0.51 $\pm$ 0.02
L2E w/o historical target	0.21 $\pm$ 0.0	0.46 $\pm$ 0.02
L2E w all pairs	0.39 $\pm$ 0.03	0.57 $\pm$ 0.010
L2E	0.44$\pm$0.04	0.57$\pm$0.02

Table 3.11: Ablation study

knowledge from historical source and target domains could indeed improve the performance of L2E in dynamic transfer learning. Besides, we also consider to generate the meta-pairs from any two historical domains (indicated in “L2E w all pairs” in Table 3.11). It does not outperform L2E with only the meta-pairs from consecutive tasks. One explanation is that it might generate the unrelated meta-pairs of tasks [87].

3.4 DYNAMIC TRANSFER LEARNING WITH OPEN-SET TARGETS

This section extends dynamic transfer learning to open-set scenarios involving unknown classes in the target domain. Specifically, we aim to understand the knowledge transferability from a static source domain to a time-evolving target domain with unknown classes [10].

3.4.1 Problem Definition

Formally, we define the problem of dynamic transfer learning with open-set targets [10] as follows (shown in Figure 3.9).

Problem 3.5. Given a labeled source domain $\mathcal{D}^s$ and a time-evolving unlabeled target domain $\{\mathcal{D}_k^t\}_{k=1}^N$ with time stamp k , where $\mathcal{Y}^s \subset \mathcal{Y}_k^t$, *dynamic open-set transfer learning* aims to learn an optimal prediction function for target domain $\mathcal{D}_{N+1}^t$ using the knowledge from the source domain and the historical target domains, such that it can classify the samples from the shared classes in the target domain $\mathcal{D}_{N+1}^t$ correctly, and identify the samples from all the unknown classes as “unknown”.

We assume that the data distribution of the target domain is continuously evolving. Unlike Section 3.3, this assumption involves the data distribution of target samples within known

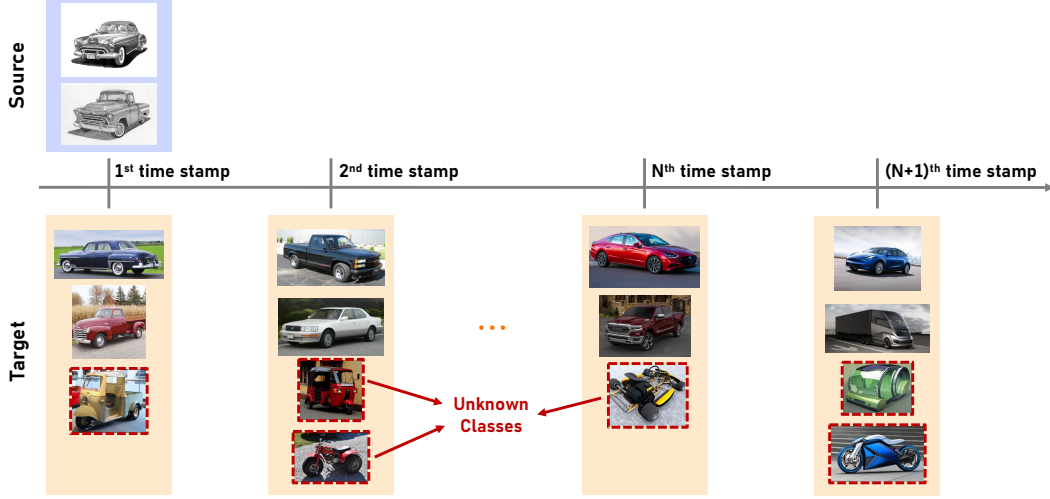


Figure 3.9: Illustration of dynamic transfer learning with open-set targets. Novel unknown classes may appear in the time-evolving target domain.

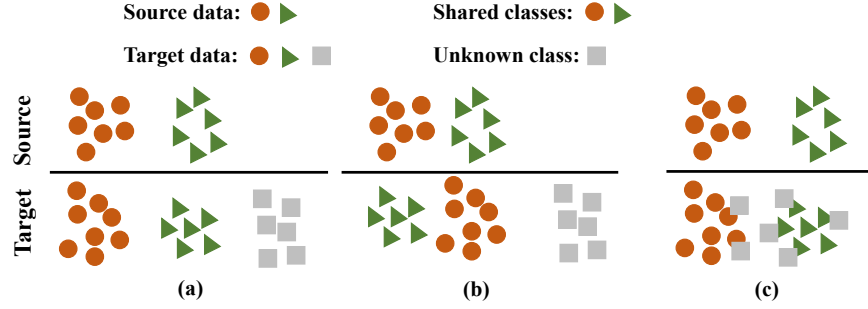


Figure 3.10: Distribution alignment in open-set transfer learning scenarios, including (a) label-informed distribution alignment after identifying the “unknown” class, (b) marginal distribution alignment after identifying the “unknown” class, and (c) distribution alignment without identifying the “unknown” class.

classes (i.e., the classes that appear in the source domain). In other words, the distribution change of the target domain at the adjacent time stamps is upper bounded by a small constant $\aleph > 0$, i.e., $d(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{k+1|\text{known}}^t) \leq \aleph$, $k = 1, \dots, N$ where $\mathbb{P}_{k|\text{known}}^t$ denotes the data distribution of the target domain at the k^{th} time stamp with only known classes. As we will show in Theorem 3.5, this assumption ensures the successful knowledge transfer across domains and time stamps, despite newly appearing samples from unknown classes in the target domain.

3.4.2 Theoretical Analysis

Following [38], we start by studying a binary classification problem with $\mathcal{Y}_s \in \{0, 1\}$. A typical domain discrepancy measure between the source and target domains is as follows.

Definition 3.3. ($\mathcal{H}\Delta\mathcal{H}$ -divergence [38]) For $h, h' \in \mathcal{H}$, let B denote the subset of $\mathcal{X}$ such that $h(x) \neq h'(x)$ for any $x \in B$, the $\mathcal{H}\Delta\mathcal{H}$ -divergence between source and target domains over $\mathcal{X}$ is defined as:

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_X^t) = \sup_{h, h' \in \mathcal{H}} \left| \Pr_{\mathbb{P}_X^s}[B] - \Pr_{\mathbb{P}_X^t}[B] \right| \quad (3.87)$$

However, it cannot be directly applied to measure the domain discrepancy for open-set domain adaptation since novel unknown classes exist in the target domain. In particular, as shown in Figure 3.10(c), distribution alignment with $\mathcal{H}\Delta\mathcal{H}$ -divergence in the presence of the “unknown” target class might lead to non-separable representation of target examples in the latent feature space. This is because in this case, it might enforce the alignment of the target examples of “unknown” class and the source examples of known classes.

One solution proposed in previous work [260, 261] is to minimize the marginal domain discrepancy between source and target domains within the known classes, without using any label information. Nevertheless, it may have two limitations: (i) it does not explicitly identify the “unknown” class, thus can suffer from the same problem of distribution alignment with $\mathcal{H}\Delta\mathcal{H}$ -divergence (Figure 3.10(c)); (ii) marginal distribution alignment for source and target data within the known classes cannot guarantee the success of knowledge transfer. As illustrated in Figure 3.10(b), exact marginal distribution alignment might lead to negative transfer [26], which is consistent with recent theoretical analysis [103].

To solve these problems, we propose a novel discriminant domain discrepancy measure named $\mathcal{OS}$ -divergence in the context of open-set targets. In particular, we argue that a qualified open-set domain discrepancy measure should satisfy the following conditions (see Figure 3.10(a)): (i) identifying the “unknown” class and (ii) measuring the joint distribution shift (w.r.t. both the input samples and the output class labels) of source and target domains within the known classes. Inspired by the Fisher criterion [262] that encourages within-class examples to be close and between-class examples to be separable, we define $\mathcal{OS}$ -divergence as the difference of the within-class discrepancy to the between-class discrepancy.

Definition 3.4. ($\mathcal{OS}$ -divergence) For any $h \in \mathcal{H}$, let $I(h)$ denote the subset of $\mathcal{X}$ such that $x \in I(h) \Leftrightarrow h(x) = 1$ and $\overline{I(h)}$ be the complement of $I(h)$. The $\mathcal{OS}$ -divergence between the

source and target tasks is defined as

$$d_{\mathcal{OS}}(\mathbb{P}^s, \mathbb{P}^t) = d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t) - \rho \cdot d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t) \quad (3.88)$$

where $\mathbb{P}_{\text{known}}^t$ and $\mathbb{P}_{X,\text{unknown}}^t$ denotes the joint distribution (over $\mathcal{X} \times \mathcal{Y}$) and the marginal distribution (over $\mathcal{X}$) of the target domain when only considering the known classes, $\rho > 0$ is a discrepancy coefficient, and $d_{\mathcal{C}}(\cdot, \cdot)$ is a label-informed domain discrepancy measure [39]

$$d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t) = \sup_{h \in \mathcal{H}} \left| \Pr_{\mathbb{P}^s}[\{I(h), y = 1\} \cup \{\overline{I(h)}, y = 0\}] - \Pr_{\mathbb{P}_{\text{known}}^t}[\{I(h), y = 1\} \cup \{\overline{I(h)}, y = 0\}] \right| \quad (3.89)$$

We would like to point out that $\mathcal{OS}$ -divergence can be considered as the first-order Taylor series of $d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t) + \frac{\rho}{d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t)} - 2\rho$, as $d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t) + \frac{\rho}{d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t)} - 2\rho \approx d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t) - \rho \cdot d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t) + O((d_{\mathcal{H}\Delta\mathcal{H}}^2(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t)))$. One intuitive explanation of the $\mathcal{OS}$ -divergence is that minimizing $\mathcal{OS}$ -divergence is equivalent to encouraging the matching of joint distribution across domains within the known classes, and in the meanwhile, distinguishing the marginal distribution between the known and “unknown” classes.

Let $\pi^t = \Pr(y = C + 1)$ be the class-prior probability of the “unknown” class in the target domain, given that there are C known classes in the source domain. Based on $\mathcal{OS}$ -divergence, we derive the generalization error bound for dynamic transfer learning with open-set targets. For notation convenience, we will use $\mathcal{D}_0^t$ to represent the source domain $\mathcal{D}^s$ below.

Theorem 3.5. Let $\mathbb{P}_{k|\text{known}}^t$ be the joint distribution of the target domain over $\mathcal{X} \times \mathcal{Y}$ at the k^{th} time stamp when only considering the known classes. Suppose that the loss function $\mathcal{L}(\cdot, \cdot)$ is bounded, i.e., $|\mathcal{L}(\cdot, \cdot)| \leq M$. For any hypothesis $h \in \mathcal{H}$ and $\sum_{k=0}^N \alpha_k = 1$ where $\alpha_k \geq 0$ ($k = 0, \dots, N$), there exists $\rho > 0$ such that the expected prediction error $\mathcal{E}_{N+1}^t(h)$ of the newest target domain is bounded as:

$$\begin{aligned} \mathcal{E}_{N+1}^t(h) &\leq (1 - \pi_{N+1}^t) \left(\sum_{k=0}^N \alpha_k \mathbb{E}_{(x,y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] + 4M \sum_{k=0}^N \alpha_k d_{\mathcal{OS}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1}^t) \right) \\ &\quad + \Delta_{PU} + (1 - \pi_{N+1}^t) M \left(2 - \pi_{N+1}^t + \sum_{k=0}^N \alpha_k |\pi_{N+1}^t - \pi_k^t| \right) \end{aligned} \quad (3.90)$$

where $\Delta_{PU} = \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t} [\mathcal{L}(h(x), C + 1)] - (1 - \pi_{N+1}^t) \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), C + 1)]$ and π_k^t is the class-prior probability of the “unknown” class in the target domain at the k^{th} time stamp.

Proof. It holds that $\mathbb{P}^t(x) = \pi^t \mathbb{P}^t(x|y = C + 1) + \sum_{c=1}^C \pi_c^t \mathbb{P}^t(x|y = c)$ where π_c^t is the class-prior probability of class c in the target domain. Based on the theory of positive-unlabeled learning [263, 264], the target error can be given by:

$$\begin{aligned} \mathcal{E}_{N+1}^t(h) &= \sum_{c=1}^C \pi_{c,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=c)} [\mathcal{L}(h(x), y = c)] \\ &\quad + \pi_{C+1,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)] \end{aligned} \quad (3.91)$$

where $\pi_{c,k}^t$ is the class-prior probability of class c in the target domain at the k^{th} time stamp and $\pi_{C+1,k}^t \equiv \pi_k^t$. Based on the observation, for any h ,

$$\begin{aligned} &d_{\mathcal{H}\Delta\mathcal{H}}((\mathbb{P}_{N+1}^t(x|y = C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1))) + \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [h(x) = C + 1] \\ &\leq d_{\mathcal{H}\Delta\mathcal{H}}((\mathbb{P}_{N+1}^t(x|y = C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1))) + 1 - \pi_{N+1}^t \leq 2 - \pi_{N+1}^t \end{aligned} \quad (3.92)$$

then, we can have the following results.

$$\begin{aligned}
& \pi_{C+1,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
= & \pi_{C+1,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
& + (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
\leq & \pi_{C+1,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
& + (1 - \pi_{C+1,N+1}^t) M \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [h(x) = C + 1] \\
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
\leq & \pi_{C+1,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
& + (1 - \pi_{C+1,N+1}^t) M (2 - \pi_{C+1,N+1}^t - d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{N+1}^t(x|y = C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1))) \\
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
\leq & \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x)} [\mathcal{L}(h(x), y = C + 1)] \\
& + (1 - \pi_{C+1,N+1}^t) M (2 - \pi_{C+1,N+1}^t - d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{N+1}^t(x|y = C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1))) \\
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
\leq & \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x)} [\mathcal{L}(h(x), y = C + 1)] \\
& + (1 - \pi_{C+1,N+1}^t) M \sum_{k=0}^N \alpha_k \left(2 - \pi_{C+1,N+1}^t + d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_k^t(x|y \neq C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1)) \right. \\
& \left. - d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{N+1}^t(x|y = C + 1), \mathbb{P}_k^t(x|y \neq C + 1)) \right) \\
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
\leq & (1 - \pi_{C+1,N+1}^t) M \sum_{k=0}^N \alpha_k \left(2 - \pi_{C+1,N+1}^t + d_{\mathcal{C}}(\mathbb{P}_k^t|_{\text{known}}, \mathbb{P}_{N+1}^t|_{\text{known}}) \right. \\
& \left. - d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{X,k| \text{known}}^t, \mathbb{P}_{X,N+1| \text{unknown}}^t) \right) \\
& + \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x)} [\mathcal{L}(h(x), y = C + 1)] - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)]
\end{aligned} \tag{3.93}$$

Here we use the following observation.

$$\begin{aligned}
d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_k^t(x|y \neq C + 1), \mathbb{P}_{N+1}^t(x|y \neq C + 1)) &= d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{X,k| \text{known}}^t, \mathbb{P}_{X,N+1| \text{known}}^t) \\
&\leq d_{\mathcal{C}}(\mathbb{P}_k^t|_{\text{known}}, \mathbb{P}_{N+1}^t|_{\text{known}})
\end{aligned} \tag{3.94}$$

Therefore, we have

$$\begin{aligned}
\mathcal{E}_{N+1}^t(h) &\leq \sum_{c=1}^C \pi_{c,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=c)} [\mathcal{L}(h(x), y = c)] \\
&+ (1 - \pi_{C+1,N+1}^t) M \sum_{k=0}^N \alpha_k \left(2 - \pi_{C+1,N+1}^t + d_{\mathcal{C}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1|\text{known}}^t) \right. \\
&\left. - d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{X,k|\text{known}}^t, \mathbb{P}_{X,N+1|\text{unknown}}^t) \right) \\
&+ \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x)} [\mathcal{L}(h(x), y = C + 1)] - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)]
\end{aligned} \tag{3.95}$$

For its first term, we see that it is a target prediction error within the known classes.

$$\begin{aligned}
&\sum_{c=1}^C \pi_{c,N+1}^t \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=c)} [\mathcal{L}(h(x), y = c)] \\
&= (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{(x,y) \sim \mathbb{P}_{N+1|\text{known}}^t} [\mathcal{L}(h(x), y)] \\
&= \sum_{k=0}^N \alpha_k (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{(x,y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] \\
&\quad + \sum_{k=0}^N \alpha_k (1 - \pi_{C+1,N+1}^t) \left(\mathbb{E}_{(x,y) \sim \mathbb{P}_{N+1|\text{known}}^t} [\mathcal{L}(h(x), y)] - \mathbb{E}_{(x,y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] \right) \\
&\leq (1 - \pi_{C+1,N+1}^t) \left(\sum_{k=0}^N \alpha_k \mathbb{E}_{(x,y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] \right. \\
&\quad \left. + M \sum_{k=0}^N \alpha_k \left| \Pr_{\mathbb{P}_{N+1|\text{known}}^t} [h(x) \neq y] - \Pr_{\mathbb{P}_{k|\text{known}}^t} [h(x) \neq y] \right| \right) \\
&\leq (1 - \pi_{C+1,N+1}^t) \left(\sum_{k=0}^N \alpha_k \mathbb{E}_{(x,y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] + M \sum_{k=0}^N \alpha_k d_{\mathcal{C}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1|\text{known}}^t) \right)
\end{aligned} \tag{3.96}$$

In addition, we have

$$\begin{aligned}
& - (1 - \pi_{C+1,N+1}^t) \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)] \\
& = (1 - \pi_{C+1,N+1}^t) \cdot \left(- \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] \right. \\
& \quad \left. + \sum_{k=0}^N \alpha_k (\mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] - \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y \neq C+1)} [\mathcal{L}(h(x), y = C + 1)]) \right) \\
& \leq (1 - \pi_{C+1,N+1}^t) \cdot \left(- \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] \right. \\
& \quad \left. + M \sum_{k=0}^N \alpha_k \cdot \left| \Pr_{\mathbb{P}_{N+1}^t} [h(x) \neq C + 1] - \Pr_{\mathbb{P}_k^t} [h(x) \neq C + 1] \right| \right) \\
& \leq (1 - \pi_{C+1,N+1}^t) \cdot \left(- \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] \right. \\
& \quad \left. + M \sum_{k=0}^N \alpha_k \cdot \left(\left| \Pr_{\mathbb{P}_{N+1}^t} [h(x) \neq y] - \Pr_{\mathbb{P}_k^t} [h(x) \neq y] \right| \right. \right. \\
& \quad \left. \left. + \left| \Pr_{\mathbb{P}_{N+1}^t} [y \neq C + 1] - \Pr_{\mathbb{P}_k^t} [y \neq C + 1] \right| \right) \right) \\
& = (1 - \pi_{C+1,N+1}^t) \cdot \left(- \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] \right. \\
& \quad \left. + M \sum_{k=0}^N \alpha_k \left(\left| \Pr_{\mathbb{P}_{N+1}^t} [h(x) \neq y] - \Pr_{\mathbb{P}_k^t} [h(x) \neq y] \right| + |\pi_{C+1,N+1}^t - \pi_{C+1,k}^t| \right) \right) \\
& \leq -(1 - \pi_{C+1,N+1}^t) \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X,k}^t} [\mathcal{L}(h(x), y = C + 1)] \\
& \quad + (1 - \pi_{C+1,N+1}^t) M \sum_{k=0}^N \alpha_k d_C(\mathbb{P}_k^t, \mathbb{P}_{N+1}^t) + (1 - \pi_{C+1,N+1}^t) M \sum_{k=0}^N \alpha_k |\pi_{N+1}^t - \pi_k^t|
\end{aligned} \tag{3.97}$$

Therefore, the following holds

$$\begin{aligned}
\mathcal{E}_{N+1}^t(h) &\leq \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x)} [\mathcal{L}(h(x), y = C + 1)] \\
&\quad + (1 - \pi_{C+1, N+1}^t) \sum_{k=0}^N \alpha_k \mathbb{E}_{(x, y) \sim \mathbb{P}_{k|\text{known}}^t} [\mathcal{L}(h(x), y)] \\
&\quad + 4(1 - \pi_{C+1, N+1}^t) M \sum_{k=0}^N \alpha_k d_{\mathcal{OS}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1}^t) \\
&\quad - (1 - \pi_{C+1, N+1}^t) \sum_{k=0}^N \alpha_k \mathbb{E}_{x \sim \mathbb{P}_{X, k|\text{known}}^t} [\mathcal{L}(h(x), y = C + 1)] \\
&\quad + (1 - \pi_{C+1, N+1}^t) M \left(2 - \pi_{C+1, N+1}^t + \sum_{k=0}^N \alpha_k |\pi_{N+1}^t - \pi_k^t| \right)
\end{aligned} \tag{3.98}$$

where $d_{\mathcal{OS}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1}^t) = d_{\mathcal{C}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1|\text{known}}^t) - \frac{1}{4} d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{X, k|\text{known}}^t, \mathbb{P}_{X, N+1|\text{unknown}}^t)$.

QED.

Theorem 3.5 shows that the target error $\mathcal{E}_{N+1}^t(h)$ could be upper bounded in terms of the prediction error on the historical domains, the distribution discrepancy across domains, and an open-set risk Δ_{PU} . Furthermore, we provide a lower bound for dynamic transfer learning with open-set targets.

Theorem 3.6. Let $\mathbb{P}_{X, k|\text{known}}^t$ denote the marginal distribution over $\mathcal{X}$ of the target domain at the k^{th} time stamp when only considering the known classes. For a hypothesis $h \in \mathcal{H}$, if $\mathcal{L}(h(x), y) = |h(x) - y|$, $\mathcal{E}^s(h) \equiv \mathcal{E}_0^t(h) = 0$ and $\mathbb{P}_{X, k|\text{known}}^t = \mathbb{P}_{X, k'|\text{known}}^t$ for any $k, k' = 0, 1, \dots, N + 1$, then the expected target error $\mathcal{E}_{N+1}^t(h)$ has the following lower bound.

$$\mathcal{E}_{N+1}^t(h) \geq \pi_{N+1}^t \mathcal{E}_{N+1}^U(h) + \left| \sum_{c=1}^C c (\mathbb{P}_{N+1}^t(y = c) - (1 - \pi_{N+1}^t) \mathbb{P}^s(y = c)) \right| \tag{3.99}$$

where $\mathcal{E}_{N+1}^U(h) = \mathbb{E}_{x \sim \mathbb{P}_{N+1}^t(x|y=C+1)} [\mathcal{L}(h(x), y = C + 1)]$ is the “unknown” prediction error.

Proof. Since $\mathcal{E}^s(h) = 0$, we have

$$0 = \mathcal{E}^s(h) \geq \left| \sum_{c=1}^C \int_x \mathbb{P}^s(x, y = c) (h(x) - c) dx \right| = \left| \int_x \mathbb{P}_X^s(x) h(x) dx - \sum_{c=1}^C c \mathbb{P}^s(y = c) \right| \tag{3.100}$$

Thus, it holds that $\int_x \mathbb{P}_X^s(x) h(x) dx = \sum_{c=1}^C c \mathbb{P}^s(y = c)$ for the source domain. Therefore, we

have

$$\begin{aligned}
\mathcal{E}_{N+1}^t(h) &= \mathbb{E}_{(x,y) \sim \mathbb{P}_{N+1}^t} [\mathcal{L}(h(x), y)] \\
&= \sum_{c=1}^C \int_x \mathbb{P}_{N+1}^t(x, y=c) \mathcal{L}(h(x), y=c) dx + \int_x \mathbb{P}_{N+1}^t(x, y=C+1) \mathcal{L}(h(x), y=C+1) dx \\
&= \sum_{c=1}^C \int_x \mathbb{P}_{N+1}^t(x, y=c) |h(x) - y| dx + \pi_{N+1}^t \mathcal{E}_{N+1}^U(h) \\
&\geq \left| \sum_{c=1}^C \int_x \mathbb{P}_{N+1}^t(x, y=c) (h(x) - c) dx \right| + \pi_{N+1}^t \mathcal{E}_{N+1}^U(h) \\
&= \left| (1 - \pi_{N+1}^t) \int_x \mathbb{P}_{X, N+1 | \text{known}}^t(x) h(x) dx - \sum_{c=1}^C c \mathbb{P}_{N+1}^t(y=c) \right| + \pi_{N+1}^t \mathcal{E}_{N+1}^U(h) \\
&= \left| (1 - \pi_{N+1}^t) \int_x \mathbb{P}_X^s(x) h(x) dx - \sum_{c=1}^C c \mathbb{P}_{N+1}^t(y=c) \right| + \pi_{N+1}^t \mathcal{E}_{N+1}^U(h) \\
&= \left| (1 - \pi_{N+1}^t) \sum_{c=1}^C c \mathbb{P}^s(y=c) - \sum_{c=1}^C c \mathbb{P}_{N+1}^t(y=c) \right| + \pi_{N+1}^t \mathcal{E}_{N+1}^U(h)
\end{aligned} \tag{3.101}$$

which completes the proof. QED.

Theorem 3.6 shows that the target error $\mathcal{E}_{N+1}^t(h)$ can be large, even when the source prediction error is zero and the marginal distribution within C known classes across domains are exactly matched (i.e., discrepancy distance [111] used in previous works [260, 261] is zero).

3.4.3 Proposed Algorithm

The goal of dynamic transfer learning with open-set targets is to classify the target samples within the C known classes correctly, and to identify all the “unknown” samples appearing in the new target domain. To this end, we propose a novel approach named **OuterAdapter** [10], by empirically minimizing the upper bound of target error in Theorem 3.5. The objective function of **OuterAdapter** is formulated as follows.

$$\min_{\theta} \mathcal{O}_{PU}(\theta) + \beta \sum_{k=0}^N \alpha_k d_{\mathcal{OS}}(\mathbb{P}_{k|\text{known}}^t, \mathbb{P}_{N+1}^t; \theta) \tag{3.102}$$

Algorithm 3.3: OuterAdapter

Input: A static source domain $\mathcal{D}^s$ with labeled samples, a time evolving target domain $\{\mathcal{D}_k^t\}_{k=1}^{N+1}$ with unlabeled samples, hyper-parameters ρ, λ, β ;
Output: Prediction function on the newest target domain $\mathcal{D}_{N+1}^t$.
Randomly initialize the model parameters θ ;
foreach k **in** $[0, 1, \dots, N + 1]$ **do**
 while *Stopping criterion is not satisfied* **do**
 Choose labeled samples (x_i^s, y_i^s) from $\mathcal{D}^s$;
 foreach $j \in \{1, \dots, k - 1\}$ **do**
 Choose labeled samples $(x_{ij}^t, \hat{y}_{ij}^t)$ from $\mathcal{D}_j^t$;
 end
 Choose unlabeled samples x_{ik}^t from $\mathcal{D}_k^t$;
 Estimate the probability of x_{ik}^t as “unknown”;
 Choose top- p unlabeled samples as “unknown” according to the estimated probability, and others as known ones;
 Estimate $d_{\mathcal{OS}}$ and α_j ($j = 0, \dots, k$);
 Update parameters θ and a_c using gradient descent;
 end
 Estimate pseudo-label $\hat{y}_{ik}^t$ for x_{ik}^t in $\mathcal{D}_k^t$;
end

and

$$\begin{aligned} \mathcal{O}_{PU}(\theta) = & \sum_{k=0}^N \frac{\lambda \alpha_j}{n_k^t} \sum_{i=1}^{n_k^t} (\mathcal{L}(h(x_{ik}^t), \hat{y}_{ik}^t; \theta) - \mathcal{L}(h(x_{ik}^t), C + 1; \theta)) \\ & + \frac{1}{m_{N+1}^t} \sum_{i=1}^{m_{N+1}^t} \mathcal{L}(h(x_{ik}^t), C + 1; \theta) \end{aligned} \quad (3.103)$$

where n_k^t denotes the number of samples within C known classes and m_k^t is the number of all samples in the target domain at the k^{th} time stamp. $\mathcal{O}_{PU}(\theta)$ is an unbiased positive-unlabeled learning based prediction error parameterized by θ , $\lambda = 1 - \pi_{N+1}^t \geq 0$ is the class-prior probability, and $\beta \geq 0$ is a hyper-parameter to balance different terms. Here θ represents all the trainable parameters in the proposed objective function. $\hat{y}_{ik}^t$ is the pseudo-label of the sample x_{ik}^t where $\hat{y}_{ik}^t = y_{ik}^t$ for labeled source examples and $\hat{y}_{ik}^t$ would be empirically estimated for unlabeled historical target samples.

The overall training process of **OuterAdapter** is illustrated in Algorithm 3.3. It is given the labeled source data and a sequence of unlabeled target data as the input, and outputs the optimal prediction function h for the newest target domain $\mathcal{D}_{N+1}^t$. In the following, we estimate the parameters α_k and $\hat{y}_{ik}^t$ as well as open-set discrepancy measure $d_{\mathcal{OS}}$ involved in the proposed objective function.

- *Estimation of pseudo-label $\hat{y}_{ik}^t$* : We sequentially learn the pseudo-label $\hat{y}_{ik}^t$ for an input sample x_{ik}^t appearing in the target domain $\mathcal{D}_k^t$ as follows. With labeled source data, the prediction function of the first target domain $\mathcal{D}_1^t$ can be learned by applying our **OuterAdapter** algorithm on the labeled source samples (x_i^s, y_i^s) and the unlabeled target samples x_{i1}^t . Then the pseudo-label $\hat{y}_{i1}^t$ of example x_{i1}^t from the first target domain $\mathcal{D}_1^t$ can be obtained. After that, the prediction function of the target domain $\mathcal{D}_2^t$ is learned by applying our **OuterAdapter** algorithm on the labeled samples $(x_i^s, y_i^s)(x_{i1}^t, \hat{y}_{i1}^t)$ and the unlabeled target samples x_{i2}^t for deriving the pseudo-label $\hat{y}_{i2}^t$ of the sample x_{i2}^t . Therefore, the pseudo-label $\hat{y}_{ik}^t$ of historical unlabeled target samples can be sequentially estimated.
- *Estimation of $d_{\mathcal{OS}}$* : Following [38, 39], the $\mathcal{H}\Delta\mathcal{H}$ -divergence and $\mathcal{C}$ -divergence can be estimated from finite samples using the adversarial domain discriminator. However, it is hard to select the “unknown” target samples to estimate $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_X^s, \mathbb{P}_{X,\text{unknown}}^t)$ because we have no prior information regarding the “unknown” class. Thus we choose the target samples that are classified as the “unknown” class with a high prediction probability. $d_{\mathcal{C}}(\mathbb{P}^s, \mathbb{P}_{\text{known}}^t)$ can be estimated in a similar way. In this case, the pseudo-label $\hat{y}_{ik}^t$ of input x_{ik}^t is used to measure the distribution shift across domains.
- *Estimation of α_j* : In our algorithm, α_k indicates the importance of $\mathcal{D}_k^t$ on learning the prediction function for the newest target domain $\mathcal{D}_{N+1}^t$. In this paper, we propose a simple yet effective self-attention scheme to automatically learn the value of α_k : $\alpha_k = \exp(\text{LeakyReLU}(\sum_{c=1}^C a_c^T \bar{x}_{ck}^t))$ and $\alpha_k = \alpha_k / \sum_{k'=0}^N \alpha_{k'}$ where $\cdot^T$ denotes the transpose. Here a_c is a class-specific weight parameter and $\bar{x}_{ck}^t (k = 0, \dots, N+1)$ is the class-specific average feature vector of examples x_{ik}^t within the class c in the domain $\mathcal{D}_k^t$, by using either the real class-label y_{ik}^t (if available) or pseudo-label $\hat{y}_{ik}^t$.

3.4.4 Experiment

In the experiments, we evaluate the effectiveness of the proposed **OuterAdapter** algorithm for dynamic transfer learning with open-set targets.

Data Sets. We use the following public image data sets.

- **Office-Home** [265]: It has 15,500 images of 65 categories from four domains: Art, Clipart, Product, and Real-World. In this data set, we select the first 25 classes in alphabetical order as the known classes and two other classes as “unknown” for the first target domain. In real scenarios, the new unknown classes might appear in the target

domain over time. Thus, we select two more novel classes as “unknown” in the target domain for every time stamp. Besides, the data distribution of the target domain would be time evolving in real scenarios due to the changing visual environment. To this end, we generate a set of time evolving domains by adding the random salt&pepper noise and rotation to the sampled raw images in every time stamp. Specifically, we generate the data of $\mathcal{D}_k^t$ by rotating the original images with degree O_d and adding the random salt&pepper noise with magnitude O_n , i.e., $O_d = 30 \cdot (k - 1)$ and $O_n = 0.05 \cdot (k - 1)$.

- Office-31 [259]: It contains 4,652 images of 31 categories from three domains: Amazon, Webcam, and DSLR. In this data set, we select the first 10 classes in alphabetical order as the known classes and two other classes as “unknown” for the first target domain. Thus, we select two more novel classes as “unknown” in the target domain for every time stamp. We generate a set of time evolving domains by rotating the raw images with degree O_d and adding the random salt&pepper noise with magnitude O_n , i.e., $O_d = 30 \cdot (k - 1)$ and $O_n = 0.05 \cdot (k - 1)$.
- Syn2Real-O [266]: It is a large open-set adaptation benchmark with over 200K images of 12 object categories from two distinct domains: Synthetic and Real. In this case, the source domain (Synthetic) uses the training data from VisDA-17 as the known classes and the target domain (Real) contains the test data from VisDA-17 (known set) as well as 50k images from irrelevant categories of the COCO data set (unknown classes). For the target domain, we choose all the known classes at every time stamp and randomly choose $k\%$ of the entire unknown classes at the k^{th} time stamp. Besides, we simulate the distribution shift of the target domain over time by rotating the raw images with degree O_d and adding the random salt&pepper noise with magnitude O_n , i.e., $O_d = 30 \cdot (k - 1)$ and $O_n = 0.05 \cdot (k - 1)$.

Baselines. We use the following baseline methods: (1) static closed-set adaptation methods: SourceOnly with basic ResNet-50 [69] and DANN [88]; (2) multi-source adaptation methods: MDAN [124] and DARN [112]; (3) dynamic closed-set adaptation methods: CUA [253] and TransLATE [39]; (4) static open-set adaptation methods: OSBP [43] and DAMC [267]; (5) dynamic open-set adaptation methods: it combines the open-set adaptation methods with CUA [253] to leverage the historical target knowledge, i.e., OSBP+CUA and DAMC+CUA. In addition to the proposed **OuterAdapter** algorithm, we also consider its simple variant **OuterAdapter** $_{\mu}$ where $\alpha_k = \frac{\mu^{N-k}}{\sum_{k'=0}^N \mu^{N-k'}}$ for $0 \leq \mu \leq 1$. Here we turn the hyperparameter $\mu \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

Model Configuration. We adopted ResNet-50 [69] pre-trained on ImageNet as the base

Method	$\mathcal{D}_{t_1}$		$\mathcal{D}_{t_2}$		$\mathcal{D}_{t_3}$		$\mathcal{D}_{t_4}$		$\mathcal{D}_{t_5}$		$\mathcal{D}_{t_6}$	
	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*
SourceOnly	0.542 $\pm$ 0.011	0.564 $\pm$ 0.012	0.372 $\pm$ 0.011	0.387 $\pm$ 0.012	0.300 $\pm$ 0.012	0.312 $\pm$ 0.012	0.316 $\pm$ 0.008	0.329 $\pm$ 0.008	0.271 $\pm$ 0.014	0.282 $\pm$ 0.014	0.258 $\pm$ 0.011	0.269 $\pm$ 0.011
DANN	0.599 $\pm$ 0.014	0.622 $\pm$ 0.014	0.434 $\pm$ 0.016	0.452 $\pm$ 0.016	0.329 $\pm$ 0.012	0.342 $\pm$ 0.013	0.340 $\pm$ 0.009	0.353 $\pm$ 0.009	0.285 $\pm$ 0.008	0.296 $\pm$ 0.008	0.299 $\pm$ 0.016	0.311 $\pm$ 0.017
MDAN	0.597 $\pm$ 0.008	0.621 $\pm$ 0.008	0.490 $\pm$ 0.012	0.510 $\pm$ 0.012	0.415 $\pm$ 0.008	0.432 $\pm$ 0.008	0.405 $\pm$ 0.013	0.421 $\pm$ 0.013	0.392 $\pm$ 0.007	0.408 $\pm$ 0.007	0.398 $\pm$ 0.011	0.415 $\pm$ 0.011
DARN	0.601 $\pm$ 0.013	0.625 $\pm$ 0.013	0.480 $\pm$ 0.022	0.499 $\pm$ 0.023	0.412 $\pm$ 0.026	0.429 $\pm$ 0.026	0.403 $\pm$ 0.033	0.419 $\pm$ 0.034	0.389 $\pm$ 0.021	0.404 $\pm$ 0.022	0.382 $\pm$ 0.036	0.397 $\pm$ 0.038
CUA	0.602 $\pm$ 0.008	0.626 $\pm$ 0.008	0.488 $\pm$ 0.021	0.508 $\pm$ 0.022	0.408 $\pm$ 0.028	0.425 $\pm$ 0.029	0.402 $\pm$ 0.026	0.418 $\pm$ 0.027	0.382 $\pm$ 0.019	0.397 $\pm$ 0.020	0.384 $\pm$ 0.020	0.399 $\pm$ 0.020
TransLATE	0.590 $\pm$ 0.007	0.614 $\pm$ 0.007	0.473 $\pm$ 0.014	0.492 $\pm$ 0.015	0.408 $\pm$ 0.022	0.424 $\pm$ 0.023	0.382 $\pm$ 0.015	0.398 $\pm$ 0.015	0.380 $\pm$ 0.017	0.396 $\pm$ 0.017	0.380 $\pm$ 0.018	0.395 $\pm$ 0.018
OSBP	0.649 $\pm$ 0.009	0.674 $\pm$ 0.008	0.474 $\pm$ 0.015	0.489 $\pm$ 0.016	0.355 $\pm$ 0.012	0.365 $\pm$ 0.012	0.370 $\pm$ 0.002	0.382 $\pm$ 0.004	0.303 $\pm$ 0.002	0.310 $\pm$ 0.003	0.313 $\pm$ 0.006	0.319 $\pm$ 0.007
DAMC	0.610 $\pm$ 0.010	0.623 $\pm$ 0.009	0.431 $\pm$ 0.008	0.438 $\pm$ 0.011	0.316 $\pm$ 0.006	0.321 $\pm$ 0.010	0.338 $\pm$ 0.003	0.345 $\pm$ 0.001	0.278 $\pm$ 0.001	0.280 $\pm$ 0.002	0.299 $\pm$ 0.011	0.306 $\pm$ 0.013
OSBP+CUA	0.649 $\pm$ 0.009	0.674 $\pm$ 0.008	0.557 $\pm$ 0.009	0.580 $\pm$ 0.010	0.475 $\pm$ 0.006	0.493 $\pm$ 0.007	0.435 $\pm$ 0.007	0.453 $\pm$ 0.008	0.429 $\pm$ 0.008	0.446 $\pm$ 0.009	0.417 $\pm$ 0.009	0.434 $\pm$ 0.009
DAMC+CUA	0.610 $\pm$ 0.010	0.623 $\pm$ 0.009	0.487 $\pm$ 0.007	0.503 $\pm$ 0.008	0.413 $\pm$ 0.008	0.428 $\pm$ 0.009	0.404 $\pm$ 0.008	0.419 $\pm$ 0.008	0.390 $\pm$ 0.010	0.405 $\pm$ 0.010	0.380 $\pm$ 0.016	0.401 $\pm$ 0.017
OuterAdapter $_{\mu}$	0.652$\pm$0.004	0.678$\pm$0.004	0.590 $\pm$ 0.012	0.614 $\pm$ 0.012	0.507 $\pm$ 0.006	0.527 $\pm$ 0.006	0.487 $\pm$ 0.015	0.506 $\pm$ 0.016	0.440 $\pm$ 0.025	0.457 $\pm$ 0.026	0.425 $\pm$ 0.025	0.442 $\pm$ 0.027
OuterAdapter	0.652$\pm$0.004	0.678$\pm$0.004	0.598$\pm$0.005	0.622$\pm$0.005	0.541$\pm$0.005	0.562$\pm$0.004	0.525$\pm$0.004	0.546$\pm$0.004	0.501$\pm$0.006	0.521$\pm$0.007	0.494$\pm$0.008	0.513$\pm$0.008

Table 3.12: Dynamic open-set transfer learning on Office-Home (Art $\rightarrow$ Clipart)

Method	$\mathcal{D}_{t_1}$		$\mathcal{D}_{t_2}$		$\mathcal{D}_{t_3}$		$\mathcal{D}_{t_4}$		$\mathcal{D}_{t_5}$		$\mathcal{D}_{t_6}$	
	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*
SourceOnly	0.871 $\pm$ 0.009	0.958 $\pm$ 0.009	0.822 $\pm$ 0.007	0.905 $\pm$ 0.008	0.690 $\pm$ 0.003	0.759 $\pm$ 0.003	0.733 $\pm$ 0.027	0.806 $\pm$ 0.031	0.649 $\pm$ 0.009	0.714 $\pm$ 0.011	0.542 $\pm$ 0.008	0.596 $\pm$ 0.009
DANN	0.891 $\pm$ 0.006	0.980 $\pm$ 0.007	0.821 $\pm$ 0.003	0.903 $\pm$ 0.003	0.775 $\pm$ 0.009	0.853 $\pm$ 0.010	0.804 $\pm$ 0.020	0.884 $\pm$ 0.023	0.767 $\pm$ 0.019	0.844 $\pm$ 0.020	0.742 $\pm$ 0.014	0.815 $\pm$ 0.016
MDAN	0.900 $\pm$ 0.005	0.990 $\pm$ 0.005	0.866 $\pm$ 0.017	0.953 $\pm$ 0.019	0.819 $\pm$ 0.013	0.901 $\pm$ 0.019	0.834 $\pm$ 0.011	0.918 $\pm$ 0.012	0.797 $\pm$ 0.016	0.877 $\pm$ 0.017	0.792 $\pm$ 0.014	0.872 $\pm$ 0.016
DARN	0.884 $\pm$ 0.008	0.972 $\pm$ 0.008	0.831 $\pm$ 0.022	0.914 $\pm$ 0.025	0.815 $\pm$ 0.020	0.897 $\pm$ 0.021	0.815 $\pm$ 0.025	0.896 $\pm$ 0.027	0.797 $\pm$ 0.022	0.877 $\pm$ 0.024	0.803 $\pm$ 0.024	0.883 $\pm$ 0.027
CUA	0.879 $\pm$ 0.002	0.967 $\pm$ 0.003	0.850 $\pm$ 0.013	0.935 $\pm$ 0.015	0.832 $\pm$ 0.015	0.915 $\pm$ 0.017	0.834 $\pm$ 0.007	0.918 $\pm$ 0.008	0.836 $\pm$ 0.006	0.919 $\pm$ 0.007	0.834 $\pm$ 0.010	0.917 $\pm$ 0.011
TransLATE	0.897 $\pm$ 0.006	0.987 $\pm$ 0.007	0.883 $\pm$ 0.014	0.971 $\pm$ 0.015	0.849 $\pm$ 0.026	0.934 $\pm$ 0.029	0.862 $\pm$ 0.015	0.948 $\pm$ 0.017	0.856 $\pm$ 0.026	0.942 $\pm$ 0.028	0.846 $\pm$ 0.021	0.930 $\pm$ 0.023
OSBP	0.907$\pm$0.003	0.993$\pm$0.001	0.848 $\pm$ 0.013	0.929 $\pm$ 0.010	0.792 $\pm$ 0.033	0.868 $\pm$ 0.034	0.788 $\pm$ 0.003	0.862 $\pm$ 0.001	0.813 $\pm$ 0.007	0.892 $\pm$ 0.005	0.777 $\pm$ 0.033	0.853 $\pm$ 0.035
DAMC	0.894 $\pm$ 0.002	0.980 $\pm$ 0.000	0.878 $\pm$ 0.011	0.962 $\pm$ 0.010	0.828 $\pm$ 0.006	0.901 $\pm$ 0.007	0.792 $\pm$ 0.025	0.858 $\pm$ 0.025	0.770 $\pm$ 0.028	0.838 $\pm$ 0.031	0.749 $\pm$ 0.017	0.814 $\pm$ 0.026
OSBP+CUA	0.907$\pm$0.003	0.993$\pm$0.001	0.855 $\pm$ 0.001	0.940 $\pm$ 0.000	0.853 $\pm$ 0.003	0.938 $\pm$ 0.004	0.852 $\pm$ 0.003	0.936 $\pm$ 0.004	0.855 $\pm$ 0.001	0.940 $\pm$ 0.001	0.846 $\pm$ 0.003	0.931 $\pm$ 0.003
DAMC+CUA	0.894 $\pm$ 0.002	0.980 $\pm$ 0.000	0.868 $\pm$ 0.015	0.953 $\pm$ 0.017	0.847 $\pm$ 0.018	0.931 $\pm$ 0.019	0.839 $\pm$ 0.010	0.923 $\pm$ 0.011	0.840 $\pm$ 0.031	0.924 $\pm$ 0.034	0.823 $\pm$ 0.016	0.905 $\pm$ 0.018
OuterAdapter $_{\mu}$	0.901 $\pm$ 0.007	0.991 $\pm$ 0.008	0.883 $\pm$ 0.007	0.971 $\pm$ 0.008	0.877$\pm$0.019	0.964$\pm$0.021	0.872$\pm$0.008	0.959$\pm$0.009	0.862 $\pm$ 0.021	0.948 $\pm$ 0.023	0.865 $\pm$ 0.007	0.951 $\pm$ 0.008
OuterAdapter	0.901 $\pm$ 0.007	0.991 $\pm$ 0.008	0.895$\pm$0.003	0.985$\pm$0.004	0.875 $\pm$ 0.009	0.962 $\pm$ 0.010	0.868 $\pm$ 0.012	0.954 $\pm$ 0.013	0.869$\pm$0.010	0.955$\pm$0.011	0.874$\pm$0.010	0.962$\pm$0.010

Table 3.13: Dynamic open-set transfer learning on Office-31 (Amazon $\rightarrow$ DSLR)

Method	$\mathcal{D}_{t_1}$		$\mathcal{D}_{t_2}$		$\mathcal{D}_{t_3}$		$\mathcal{D}_{t_4}$		$\mathcal{D}_{t_5}$		$\mathcal{D}_{t_6}$	
	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*
SourceOnly	0.451 $\pm$ 0.005	0.488 $\pm$ 0.005	0.215 $\pm$ 0.017	0.233 $\pm$ 0.019	0.167 $\pm$ 0.011	0.181 $\pm$ 0.011	0.146 $\pm$ 0.015	0.158 $\pm$ 0.017	0.131 $\pm$ 0.007	0.142 $\pm$ 0.008	0.123 $\pm$ 0.004	0.133 $\pm$ 0.004
DANN	0.544 $\pm$ 0.019	0.589 $\pm$ 0.021	0.363 $\pm$ 0.001	0.394 $\pm$ 0.001	0.261 $\pm$ 0.011	0.283 $\pm$ 0.012	0.244 $\pm$ 0.017	0.264 $\pm$ 0.019	0.188 $\pm$ 0.010	0.204 $\pm$ 0.011	0.196 $\pm$ 0.003	0.212 $\pm$ 0.003
MDAN	0.534 $\pm$ 0.010	0.578 $\pm$ 0.011	0.358 $\pm$ 0.009	0.388 $\pm$ 0.010	0.298 $\pm$ 0.004	0.323 $\pm$ 0.004	0.277 $\pm$ 0.014	0.300 $\pm$ 0.015	0.225 $\pm$ 0.009	0.244 $\pm$ 0.010	0.244 $\pm$ 0.010	0.265 $\pm$ 0.011
DARN	0.541 $\pm$ 0.004	0.586 $\pm$ 0.004	0.376 $\pm$ 0.022	0.408 $\pm$ 0.024	0.310 $\pm$ 0.022	0.336 $\pm$ 0.024	0.269 $\pm$ 0.023	0.291 $\pm$ 0.025	0.220 $\pm$ 0.028	0.238 $\pm$ 0.031	0.230 $\pm$ 0.017	0.250 $\pm$ 0.019
CUA	0.550 $\pm$ 0.012	0.596 $\pm$ 0.013	0.384 $\pm$ 0.013	0.417 $\pm$ 0.015	0.301 $\pm$ 0.010	0.326 $\pm$ 0.011	0.263 $\pm$ 0.012	0.286 $\pm$ 0.013	0.202 $\pm$ 0.033	0.219 $\pm$ 0.036	0.171 $\pm$ 0.020	0.186 $\pm$ 0.022
TransLATE	0.551 $\pm$ 0.024	0.597 $\pm$ 0.027	0.374 $\pm$ 0.020	0.405 $\pm$ 0.021	0.297 $\pm$ 0.025	0.322 $\pm$ 0.028	0.262 $\pm$ 0.023	0.283 $\pm$ 0.024	0.210 $\pm$ 0.020	0.228 $\pm$ 0.021	0.194 $\pm$ 0.028	0.210 $\pm$ 0.030
OSBP	0.608 $\pm$ 0.006	0.658 $\pm$ 0.007	0.392 $\pm$ 0.017	0.423 $\pm$ 0.018	0.273 $\pm$ 0.007	0.291 $\pm$ 0.010	0.237 $\pm$ 0.009	0.253 $\pm$ 0.011	0.198 $\pm$ 0.016	0.211 $\pm$ 0.017	0.202 $\pm$ 0.004	0.217 $\pm$ 0.002
DAMC	0.563 $\pm$ 0.011	0.605 $\pm$ 0.013	0.345 $\pm$ 0.007	0.371 $\pm$ 0.005	0.251 $\pm$ 0.020	0.272 $\pm$ 0.021	0.231 $\pm$ 0.009	0.249 $\pm$ 0.008	0.198 $\pm$ 0.008	0.213 $\pm$ 0.005	0.190 $\pm$ 0.013	0.205 $\pm$ 0.016
OSBP+CUA	0.608 $\pm$ 0.006	0.658 $\pm$ 0.007	0.412 $\pm$ 0.006	0.445 $\pm$ 0.006	0.312 $\pm$ 0.014	0.338 $\pm$ 0.015	0.267 $\pm$ 0.014	0.289 $\pm$ 0.015	0.201 $\pm$ 0.021	0.218 $\pm$ 0.024	0.200 $\pm$ 0.016	0.216 $\pm$ 0.017
DAMC+CUA	0.563 $\pm$ 0.011	0.605 $\pm$ 0.013	0.366 $\pm$ 0.021	0.396 $\pm$ 0.023	0.262 $\pm$ 0.019	0.284 $\pm$ 0.020	0.244 $\pm$ 0.023	0.265 $\pm$ 0.024	0.182 $\pm$ 0.023	0.197 $\pm$ 0.025	0.181 $\pm$ 0.017	0.196 $\pm$ 0.019
OuterAdapter $_{\mu}$	0.615$\pm$0.010	0.666$\pm$0.011	0.469 $\pm$ 0.011	0.508 $\pm$ 0.012	0.361 $\pm$ 0.012	0.391 $\pm$ 0.013	0.284 $\pm$ 0.010	0.308 $\pm$ 0.011	0.230 $\pm$ 0.006	0.249 $\pm$ 0.006	0.208 $\pm$ 0.010	0.225 $\pm$ 0.015
OuterAdapter	0.615$\pm$0.010	0.666$\pm$0.011	0.490$\pm$0.014	0.530$\pm$0.016	0.379$\pm$0.025	0.410$\pm$0.027	0.328$\pm$0.021	0.356$\pm$0.024	0.271$\pm$0.013	0.293$\pm$0.013	0.275$\pm$0.010	0.299$\pm$0.011

Table 3.14: Dynamic open-set transfer learning on Syn2Real-O (Synthetic $\rightarrow$ Real)

network for feature extraction. The overall model can be trained by backpropagation, and we update the model parameters using stochastic gradient descent with a mini-batch of size 16 where the number of selected “unknown” samples p is searched from $\{2,4,6,8,10\}$ on estimating $\mathcal{OS}$ -divergence. In addition, the hyper-parameters ρ , λ , and β are empirically set as 0.25, 0.1 and 1.0, respectively.

Main Results. Following [43, 268], we report the normalized accuracy for all classes (OS) and normalized accuracy for the known classes only (OS*). The results on Office-Home,

Office-31, and Syn2Real-O are shown in Table 3.12, Table 3.13 and Table 3.14, respectively. The classification performance on the dynamic target domain is reported (best results are indicated in bold) after running three times. It is observed that: (1) **OuterAdapter** algorithm consistently outperforms state-of-the-art baselines by a large margin; and (2) compared to **OuterAdapter** _{μ} , the proposed self-attention scheme on automatically learning the weights α_k significantly improves the performance.

Ablation Study. We investigate the impact of $\mathcal{O}_{PU}$ and $\mathcal{OS}$ -divergence as well as the historical target knowledge in the proposed **OuterAdapter** algorithm. To be specific, we focus on the following variants. (i) **OuterAdapter** without PU-Learning: the loss term $\mathcal{O}_{PU}(\theta)$ of the objective function Eq. (3.102) is ordinary cost-sensitive learning; (ii) **OuterAdapter** without historical data: it transfers the knowledge from labeled source domain $\mathcal{D}^s$ directly without using historical target knowledge; (iii)

OuterAdapter without $\mathcal{OS}$ -divergence: it uses only PU-Learning loss term without minimizing the domain discrepancy across domains; (iv) **OuterAdapter** with $\mathcal{H}\Delta\mathcal{H}$ -divergence (or $\mathcal{C}$ -divergence): the $\mathcal{OS}$ -divergence of the objective function in Eq. (3.102) is simply replaced with $\mathcal{H}\Delta\mathcal{H}$ -divergence [38] (or $\mathcal{C}$ -divergence [39]). Table 3.15 shows the results on newest target domain from Office-31 (DSLRL→Webcam) where “w” indicates “with” and “w/o” indicates “without”. It is observed that the derived PU-Learning loss term positively affects **OuterAdapter**. In addition, the historical target knowledge and the proposed $\mathcal{OS}$ -divergence could improve the performance of **OuterAdapter** on dynamic transfer learning with open-set targets.

Method	OS	OS*
OuterAdapter w/o PU-Learning	0.545	0.600
OuterAdapter w/o historical data	0.730	0.803
OuterAdapter w/o $\mathcal{OS}$ -divergence	0.782	0.860
OuterAdapter w $\mathcal{H}$ -divergence	0.829	0.912
OuterAdapter w $\mathcal{C}$ -divergence	0.839	0.922
OuterAdapter	0.848	0.933

Table 3.15: Ablation study

CHAPTER 4: NON-IID KNOWLEDGE TRANSFERABILITY

Previous chapter delves into understanding knowledge transferability when the samples in each domain are independent and identically distributed (IID). However, this IID assumption is often violated in real scenarios, e.g., node connections within social networks and user-item interactions within recommender systems. To this end, this chapter introduces our work on non-IID transfer learning where the samples in each domain are non-independent and identically distributed (non-IID). Specifically, we focus on graph transfer learning where the knowledge is transferred from a source graph to a target graph for node-level graph mining tasks. In addition to individual node attributes, the additional graph structure information may affect knowledge transfer across graphs. This motivates us to explore two graph transfer learning scenarios. In the context of graph transfer learning, Section 4.1 assumes that the source and target graphs follow the homophilic assumption [24], i.e., connected nodes are more likely to have the same class label. Furthermore, Section 4.2 relaxes this assumption because the source and target graphs may follow different graph structure assumptions [14].

Notation. The main notation used in this chapter is summarized in Table 4.1. Suppose that a graph is represented as $G = (V, E)$, where $V = \{v_1, \dots, v_n\}$ is the set of n nodes and $E \subseteq V \times V$ is the edge set in the graph. In this paper, we consider the attributed graph. That is, each node is associated with an attribute vector. The structure of the graph G can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{uv}$ is the similarity between node u and node v . In the context of graph transfer learning, we denote $G^s = (V^s, E^s, \mathbf{X}^s)$ and $G^t = (V^t, E^t, \mathbf{X}^t)$ to be the source and target graphs, respectively. The associated adjacent matrices of source and target graphs are represented as $\mathbf{A}^s$ and $\mathbf{A}^t$, respectively.

Notation	Definition
$G = (V, E, \mathbf{X})$	A graph with nodes V , edges E , and node attributes $\mathbf{X}$
$N(v)$	Local neighborhood around the node v
$\mathbb{P}, \hat{\mathbb{P}}$	Expected distribution and its corresponding empirical distribution
G^s, G^t	Source and target graphs
$\mathbf{A}^s, \mathbf{A}^t$	Source and target adjacent matrices
n^s, n^t	Number of source nodes and target nodes
$d_b(\cdot, \cdot)$	Base domain discrepancy
$\mathcal{L}(\cdot, \cdot)$	Loss function
$\mathbb{I}$	Identity matrix
$\mathbb{1}[\cdot]$	Indicator function

Table 4.1: Notation used in Chapter 4

4.1 GRAPH TRANSFER LEARNING WITH HOMOGENEOUS STRUCTURE

In this section, we study graph transfer learning [24] assuming that both source and target graphs have homophilic structures. This assumption holds that connected nodes are more likely to have the same class labels.

4.1.1 Problem Definition

Following [54, 143], we formally define the problem of graph transfer learning for node classification tasks as follows.

Problem 4.1. Given a source graph $G^s = (V^s, E^s, \mathbf{X}^s)$ with labeled nodes and a target graph $G^t = (V^t, E^t, \mathbf{X}^t)$ with only unlabeled nodes, *graph transfer learning* aims to learn a prediction function for the target graph by leveraging knowledge from the source graph, assuming that both graphs have homophilic structures.

The distribution shifts between source and target graphs can be induced by node attributes and graph structures. One special case of graph transfer learning is that when there is no edge in both source and target graphs, it will degenerate into a standard transfer learning [2, 38] with independently drawn samples. Compared to standard transfer learning, the additional distribution shifts induced by complex graph structure lead to a more challenging transfer learning problem setting [24].

4.1.2 Motivation

Weisfeiler-Lehman (WL) graph subtree kernel [101] aims to measure the similarity of two input graphs. It holds that the structural information around node v can be represented as a sequence of WL subtrees with different depths. The WL subtrees can then be compressed by a “relabeling” function. This “relabeling” function is a hashing table mapping the discrete node attributes within the subtree into a new representation.

Definition 4.1. (Weisfeiler-Lehman Subtree [101]) Let $f_m(v)$ denote the “relabeling” function to encode the subtree structure rooted at $v \in G$ with depth m . Given a graph $G = (V, E)$ associated with initial node attributes $f_0(v)$ for $v \in V$, the Weisfeiler-Lehman subtree of depth m rooted at $v \in V$ can be represented as $f_m(v) \triangleq f_m(f_{m-1}(v); \cup_{u \in N(v)} f_{m-1}(u))$ where $N(v)$ denotes the first-order neighbors of root node v within the graph.

Based on Definition 4.1, the graph structure around the node v can be represented as a sequence of WL subtrees $\{f_0(v), f_1(v), \dots, f_m(v), \dots\}$ with different depths m . Our prior

study [99] has shown that WL subtrees can capture the essential message-passing modules in widely-used message-passing graph neural networks (GNNs) [56, 57, 58, 96]. For these GNNs, the “relabeling” function is instantiated with deep neural networks to encode continuous node representations. To be specific, from the perspective of WL subtrees, GNNs mainly consist of the following crucial steps at each iteration for feature aggregation [99].

- Feature initialization (node attribute initialization): The node features are initialized by original attribute vectors.
- Neighborhood searching (multiset-attribute determination): It decides the local neighborhood in which node gathers the information from neighbors. A seed node followed by its neighbors generates a subtree pattern.
- Neighbors sorting (multiset-attribute sorting): The neighbors are sorted in the ascending or descending order of degree values. The subtrees with permutation order of neighbors are recognized as the same one.
- Feature aggregation (attribute compression): The node feature is updated by compressing the feature vectors of the aggregated neighbors including itself.

Next, we briefly go over some existing GNNs, which follow the aforementioned steps. Taking 1-hop neighborhood $N(v) = \{u | (v, u) \in E\}$ into consideration at each iteration, the following message-passing GNNs have the same feature initialization and neighborhood searching steps. When element-wise average or max operations are employed for feature aggregation, GNNs would be invariant to the order of neighbors. We summarize the feature aggregation functions (graph convolution) of GNNs as follows.

- Graph Convolutional Network (GCN) [96]:

$$h_v^l = \sigma \left(\sum_{u \in \{v\} \cup N(v)} \hat{a}_{vu} \mathbf{W}^l h_u^{l-1} \right) \quad (4.1)$$

where $\hat{\mathbf{A}} = (\hat{a}_{vu}) \in \mathbb{R}^{n \times n}$ (n is the number of nodes) is the re-normalization of the adjacency matrix $\mathbf{A}$ with added self-loops, and $\mathbf{W}^l$ is the trainable matrix at l^{th} layer. $\sigma(\cdot)$ is the non-linear activation function. It is essentially a weighted feature aggregation from the first-order node neighborhood.

- Graph Attention Network (GAT) [98]:

$$h_v^l = \sigma \left(\sum_{u \in \{v\} \cup N(v)} \alpha_{vu} \mathbf{W}^l h_u^{l-1} \right) \quad (4.2)$$

where α_{vu} is a self-attention score indicating the importance of node u to node v on feature aggregation. It is obvious that GCN can be considered as a special case of GAT when the attention score α_{vu} is defined as $\hat{a}_{vu}$.

- GraphSAGE [56]:

$$\begin{aligned} h_{N(v)}^l &= \text{AGGREGATE}_l(\{h_u^{l-1} | u \in N(v)\}) \\ h_v^l &= \sigma(W^l \cdot \text{CONCAT}(h_v^{l-1}, h_{N(v)}^l)) \end{aligned} \quad (4.3)$$

where mean-, max- and LSTM-aggregator are introduced for feature aggregation function AGGREGATE_k at the l^{th} layer. CONCAT denotes the vector concatenation of the current node representation h_v^{l-1} and the aggregated neighborhood vector $h_{N(v)}^l$. Though LSTM considers node neighbors as an ordered sequence, the LSTM (Long Short-Term Memory [269]) aggregator is adapted on unordered neighbors with random permutations.

It can be seen that the learned representation at the l^{th} layer of GNNs encodes the WL subtree around the seed node with depth l . Notably, for graph mining tasks involving a single graph, e.g., semi-supervised node classification [96], only the message-passing philosophy of WL subtrees is investigated to design GNNs [58, 98, 270, 271]. Analogous to WL subtrees, GNNs map the structurally equivalent nodes within a graph into the same low-dimensional representation in a latent embedding space. In the context of graph transfer learning, we highlight that the following WL subtree kernel sheds light on measuring the distribution shift between source and target graphs in the feature space learned by GNNs.

Definition 4.2. (Weisfeiler-Lehman Subtree Kernel [101]) Given any two graphs $G = (V, E)$ with n nodes and $G' = (V', E')$ with n' nodes, the Weisfeiler-Lehman subtree kernel on two graphs G and G' with M iterations is defined as:

$$\mathcal{K}(G, G') = \frac{1}{nn'} \sum_{m=0}^M \sum_{v \in G} \sum_{v' \in G'} \mathbb{1}[f_m(v) = f_m(v')] \quad (4.4)$$

where $\mathbb{1}[\cdot]$ is an indicator function that returns 1 when its argument is true, 0 otherwise. $f_m(v)$ represents the subtree pattern of depth m rooted at v .

This WL subtree kernel can be rewritten as $\mathcal{K}(G, G') = \sum_{m=0}^M s(\hat{\mathbb{P}}(G_m), \hat{\mathbb{P}}'(G'_m))$ where G_m (or G'_m) is the “reabeled” graph G (or G') at depth m , and $\hat{\mathbb{P}}$ (or $\hat{\mathbb{P}}'$) is the empirical node distribution of graph G (or G'), i.e., $\hat{\mathbb{P}}(\tau|G_m) = \frac{1}{n} \sum_{i=1}^n \delta(f_m(v_i) = \tau)$ for any subtree

pattern τ . Here $s(\cdot, \cdot)$ is an inner product metric to measure the distribution similarity of $\hat{\mathbb{P}}(G_m)$ and $\hat{\mathbb{P}}'(G'_m)$. Furthermore, we have the following observations. (1) The nonparametric WL subtree graph kernel [101] can be exploited to measure the distribution shift between source and target graphs by counting the occurrence of subtrees when the node attributes are discrete. However, it might suffer when using continuous node attributes to estimate the graph similarity (or distribution discrepancy in the context of graph transfer learning). (2) Prior studies [54, 143, 144] focus on characterizing the distribution discrepancy over only the m^{th} subtree representation. They can thereby partially reveal the distribution discrepancy between source and target graphs in practice.

4.1.3 Theoretical Analysis

Inspired by the connection of the WL subtree kernel and message-passing GNNs, we propose a parametric Graph Subtree Discrepancy (GSD) [24]. GSD measures the distribution discrepancy across graphs in the latent feature space learned by the message-passing GNNs.

Following WL subtree kernel [101], we assume that given a graph $G = (V, E)$, the WL subtrees (i.e., $\{f_m(v)|v \in V\}$) with a fixed depth m are conditionally independent with respect to WL subgraph G_{m-1} at depth $m-1$, i.e., $f_m(u) \perp f_m(v)|G_{m-1}$. In this case, given the WL subgraph G_{m-1} , the subtree representations $\{f_m(v)|v \in V\}$ can thus be considered as IID samples. This indicates that the subtree (at depth m) distribution shift of source and target graphs can be measured by any existing distribution discrepancy measures, e.g, discrepancy distance [111], Maximum Mean Discrepancy [115], f -divergence [31], etc. Therefore, the proposed Graph Subtree Discrepancy can be formally defined as follows.

Definition 4.3. (Graph Subtree Discrepancy) Given two graphs $G^s = (V^s, E^s)$ and $G^t = (V^t, E^t)$, the graph subtree discrepancy between them can be defined as:

$$d_{\text{GSD}}(G^s, G^t) = \lim_{M \rightarrow \infty} \frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t) \quad (4.5)$$

where $d_b(\cdot, \cdot)$ is the base domain discrepancy.

For example, we can use discrepancy distance [111] to instantiate the base domain discrepancy $d_b(\cdot, \cdot)$ in Definition 4.3.

$$d_b(G_m^s, G_m^t) = \sup_{h, h' \in \mathcal{H}} \left| \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f_m(v)), h'(f_m(v)))] - \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f_m(v)), h'(f_m(v)))] \right| \quad (4.6)$$

where $\mathcal{H}$ is the hypothesis space. Furthermore, we show that Graph Subtree Discrepancy (GSD) satisfies the following properties

Lemma 4.1. If the “relabeling” function f_m ($m = 0, 1, \dots$) is injective and the base domain discrepancy $d_b(\cdot, \cdot)$ is bounded, for any graphs G^s, G^t , then it holds

- (a) $d_{\text{GSD}}(G^s, G^t)$ exists when M goes to infinity.
- (b) $d_b(G_{m-1}^s, G_{m-1}^t) \leq d_b(G_m^s, G_m^t)$ for any depth m .
- (c) $d_{\text{GSD}}(G^s, G^t) = d_b(G^s, G^t)$, when G^s, G^t are null graphs containing only isolated nodes.
- (d) $d_{\text{GSD}}(G^s, G^t)$ is equivalent to WL subtree kernel, when $d_b(G_m^s, G_m^t) = \langle \phi(G_m^s), \phi(G_m^t) \rangle$ where $\phi(\cdot)$ counts the number of occurrences of subtree patterns.

Proof. We first show the second **property (b) of GSD**. The function $f_m(v) = f_m(f_{m-1}(v); \cup_{u \in N(v)} f_{m-1}(u))$ learns the WL subtree of depth m rooted at $v \in V$ by compressing the representation of v and its neighbors from previous depth $m - 1$. In this case, for any node $v^s \in V^s$ and $v^t \in V^t$, there are two scenarios for studying $f_m(v^s)$ and $f_m(v^t)$. (i) If $f_{m-1}(v^s) = f_{m-1}(v^t)$ at depth $m - 1$, then $f_m(v^s) = f_m(v^t)$ only when v^s and v^t has the same node degree and their neighbors have the same representation; $f_m(v^s) \neq f_m(v^t)$, otherwise. (ii) If $f_{m-1}(v^s) \neq f_{m-1}(v^t)$ at depth $m - 1$, then $f_m(v^s) \neq f_m(v^t)$. As a result, it can be seen that the distribution discrepancy of G_m^s and G_m^t would become larger than that of G_{m-1}^s and G_{m-1}^t , as the WL subtrees become more diverse with the increase of depth m . Therefore, we conclude that $d_b(G_m^s, G_m^t)$ is non-decreasing with respect to the depth m , i.e., $d_b(G_{m-1}^s, G_{m-1}^t) \leq d_b(G_m^s, G_m^t)$ for any integer m .

For **property (a) of GSD**, we show that $d_{\text{GSD}}(G^s, G^t)$ is bounded and non-decreasing w.r.t. the depth M . If the base domain discrepancy $d_b(\cdot, \cdot)$ is bounded, there exists a constant C such that $d_b(G_m^s, G_m^t) < C$ for any m . Then $d_{\text{GSD}}(G^s, G^t)$ is also bounded, as $d_{\text{GSD}}(G^s, G^t) < \lim_{M \rightarrow \infty} \frac{1}{M+1} \sum_{m=0}^M C = C$. In addition, using property (b) of GSD, we can show that $d_{\text{GSD}}(G^s, G^t)$ is non-decreasing as follows.

$$\begin{aligned}
d_b(G_m^s, G_m^t) &\leq d_b(G_M^s, G_M^t) \quad \forall m = 0, 1, \dots, M-1 \\
\frac{1}{M} \sum_{m=0}^{M-1} d_b(G_m^s, G_m^t) &\leq d_b(G_M^s, G_M^t) \\
\frac{1}{M(M+1)} \sum_{m=0}^{M-1} d_b(G_m^s, G_m^t) &\leq \frac{1}{M+1} d_b(G_M^s, G_M^t) \\
\frac{1}{M} \sum_{m=0}^{M-1} d_b(G_m^s, G_m^t) &\leq \frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t)
\end{aligned} \tag{4.7}$$

Therefore, based on the Monotone Convergence Theorem, the non-decreasing and bounded term $\frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t)$ will converge when M goes to infinity.

For **property (c) of GSD**, when one graph only contains the isolated nodes, it can be viewed as a set of independent nodes (variables). Thus, it would degenerate into a standard adaptation setting, where the distribution shift across domains can be measured by $d_b(G^s, G^t)$ on IID nodes. In other words, when there is no edge in the graph, each subtree at any depth m rooted at v is the node v itself. Therefore, it holds that $d_b(G_0^s, G_0^t) = \dots = d_b(G_m^s, G_m^t)$ for any m . Then we have $d_{\text{GSD}}(G^s, G^t) = d_b(G_0^s, G_0^t) = d_b(G^s, G^t)$.

For **property (d) of GSD**, it can be derived using the definition of WL subtree kernel.

QED.

Based on the proposed $d_{\text{GSD}}(G^s, G^t)$, we derive the generalization error bounds of graph transfer learning in the node classification task. For any hypothesis $h \in \mathcal{H}$, the node classification error on graph G of a message-passing GNN (with L convolutional layers) is defined as $\mathcal{E}(h \circ f) = \mathbb{E}_{(v,y) \in G} [\mathcal{L}(h(f(v)), y)]$, where $f(\cdot)$ extracts the node representation. Without loss of generality, we focus on the commonly used GNNs with the feature extraction $f(v) = f_L(v)$. The following theorem shows that in the cross-network node classification task, the expected classification error of the target graph can be upper bounded in terms of the classification error of the source graph and the distribution discrepancy across graphs.

Theorem 4.1. Suppose that the base domain discrepancy $d_b(\cdot, \cdot)$ of GSD is instantiated by the discrepancy distance, and the loss function $\mathcal{L}(\cdot, \cdot)$ obeys the triangle inequality. Given a message-passing GNN with the feature extractor f and the hypothesis $h \in \mathcal{H}$, the node classification error of the target graph G^t can be bounded as follows.

$$\mathcal{E}^t(h \circ f) \leq \mathcal{E}^s(h \circ f) + d_{\text{GSD}}(G^s, G^t) + \lambda^* + R^* \quad (4.8)$$

where $\lambda^* = \mathbb{E}_{v \in V^t} [\mathcal{L}(h_*^s(f(v)), h_*^t(f(v)))]$ measures the difference of optimal source and target hypotheses on the target graph, and $R^* = \mathbb{E}_{v \in V^s} [\mathcal{L}(y, h_*^s(f(v)))] + \mathbb{E}_{v \in V^t} [\mathcal{L}(h_*^t(f(v)), y)]$ is the Bayes error on the source and target graphs. y is the class label of v . In this case, $h_*^s \in \arg \min_{h \in \mathcal{H}} \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f(v)), y)]$ and $h_*^t \in \arg \min_{h \in \mathcal{H}} \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f(v)), y)]$ are optimal source and target hypotheses, respectively.

Proof. Using Lemma 4.1, it holds that for any $M \geq L$,

$$\begin{aligned} & \frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t) - \frac{M-L+1}{M+1} d_b(G_L^s, G_L^t) \\ &= \frac{1}{M+1} \sum_{m=0}^{L-1} d_b(G_m^s, G_m^t) + \frac{1}{M+1} \sum_{m=L}^M (d_b(G_m^s, G_m^t) - d_b(G_L^s, G_L^t)) \geq 0 \end{aligned} \quad (4.9)$$

Then we have

$$\begin{aligned} d_{\text{GSD}}(G^s, G^t) &= \lim_{M \rightarrow \infty} \frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t) \\ &\geq \lim_{M \rightarrow \infty} \frac{M-L+1}{M+1} d_b(G_L^s, G_L^t) = d_b(G_L^s, G_L^t) \end{aligned} \quad (4.10)$$

That means that the following holds.

$$\begin{aligned} d_{\text{GSD}}(G^s, G^t) &= \lim_{M \rightarrow \infty} \frac{1}{M+1} \sum_{m=0}^M \sup_{h, h' \in \mathcal{H}} \left| \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f_m(v)), h'(f_m(v)))] \right. \\ &\quad \left. - \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f_m(v)), h'(f_m(v)))] \right| \\ &\geq \sup_{h, h' \in \tilde{\mathcal{H}}} \left| \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f_L(v)), h'(f_L(v)))] - \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f_L(v)), h'(f_L(v)))] \right| \\ &\geq \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f_L(v)), h_*^s(f_L(v)))] - \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f_L(v)), h_*^s(f_L(v)))] \\ &= \mathbb{E}_{v \in V^t} [\mathcal{L}(h(f(v)), h_*^s(f(v)))] - \mathbb{E}_{v \in V^s} [\mathcal{L}(h(f(v)), h_*^s(f(v)))] \end{aligned} \quad (4.11)$$

where h_*^s denotes the optimal source hypothesis. If the classification error is defined using the last graph convolutional layer, i.e., $\mathcal{E}^t(h \circ f) = \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h(f_L(v)), y)]$, we have the following results.

$$\begin{aligned} \mathcal{E}^t(h \circ f) &= \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h(f(v)), y)] \\ &\leq \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h(f(v)), h_*^t(f(v)))] + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^t(f(v)), y)] \\ &\leq \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h(f(v)), h_*^s(f(v)))] + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^s(f(v)), h_*^t(f(v)))] \\ &\quad + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^t(f(v)), y)] \\ &\leq \mathbb{E}_{(v,y) \in G^s} [\mathcal{L}(h(f(v)), h_*^s(f(v)))] + d_{\text{GSD}}(G^s, G^t) \\ &\quad + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^s(f(v)), h_*^t(f(v)))] + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^t(f(v)), y)] \\ &\leq \mathcal{E}^s(h \circ f) + \mathbb{E}_{(v,y) \in G^s} [\mathcal{L}(y, h_*^s(f(v)))] + d_{\text{GSD}}(G^s, G^t) \\ &\quad + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^s(f(v)), h_*^t(f(v)))] + \mathbb{E}_{(v,y) \in G^t} [\mathcal{L}(h_*^t(f(v)), y)] \end{aligned} \quad (4.12)$$

which completes the proof.

QED.

Compared to previous work [54], our error bound of Theorem 4.1 is hypothesis-dependent. That is, the knowledge transferability across graphs can be enhanced, if the message-passing GNN learns a latent feature space such that the distribution shift across graphs is minimized. This is in sharp contrast to previous work which focuses on evaluating the transferability of a trained GNN model. Therefore, Theorem 4.1 provides insights on designing graph transfer learning algorithms based on existing message-passing GNNs in practice.

4.1.4 Proposed Algorithm

We focus on the cross-network node classification setting from a source graph with labeled nodes to a target graph with only unlabeled nodes. The goal is to identify the class label of every node in the target domain, by leveraging the relevant knowledge from the source domain. The objective function of the proposed graph transfer learning framework (**GRADE**) is given below.

$$\min_{\theta_f, \theta_h} \frac{1}{n^s} \sum_{v^s \in V^s} \mathcal{L}(h(f(v^s; \theta_f); \theta_h), y^s) + \lambda d_{\text{GSD}}(f(G^s; \theta_f), f(G^t; \theta_f)) \quad (4.13)$$

where $f(\cdot)$ is the message-passing graph neural network function parameterized by θ_f , and $h(\cdot)$ is the classifier function parameterized by θ_h . $d_{\text{GSD}}(G^s, G^t; \theta)$ is the discrepancy minimization between source and target graphs, and $\lambda \geq 0$ is a hyper-parameter to balance these terms.

Specifically, we adopt Graph Convolutional Network (GCN) [96] (see Eq. (4.1)) as the base model to extract the subtree representations of a graph. After M iterations, we obtain the sequence of subtree representations rooted at v : $f_0(v), f_1(v), \dots, f_M(v)$, where $f_0(v)$ is the raw node attributes of v . Following [96], the final representation $f_M(v)$ can be applied to identify the class label of node v . In addition, we consider finite iterations (e.g., M) of the message-passing graph neural network for estimating GSD. That is,

$$d_{\text{GSD}}(f(G^s; \theta_f), f(G^t; \theta_f)) = \frac{1}{M+1} \sum_{m=0}^M d_b(G_m^s, G_m^t) \quad (4.14)$$

4.1.5 Experiment

In the experiments, we evaluate the proposed **GRADE** algorithm for node-level graph mining tasks.

Data Sets. We use the following benchmark data sets:

- Airport network [272]: It contains three airport networks from Brazil (BR), USA (US), and Europe (EU). Each node corresponds to an airport and the edge indicates the existence of commercial flights between two airports. The class labels of nodes are assigned based on the level of activity measured by flights or people that pass the airports. Following [54], we use node degree one-hot encoding as the node feature.
- Citation network [143]: It has two networks ACMv9 (A) and DBLPv8 (D) from ArnetMine [273]. Each node is a paper and each edge indicates the citation between two papers. Each paper is associated with a 7537-dimensional feature vector extracted from paper content. Its class label indicates the research topics.
- Social network [274]: Blog1 (B1) and Blog2 (B2) are two disjoint social networks extracted from BlogCatalog [275]. Each node is a blogger and each edge indicates the friendship between two bloggers. The blogger is associated with an 8189-dimensional feature vector extracted from the blogger’s self-description, and its class label indicates the joining group.
- Agriculture data [223]: Plant Phenotyping predicts diverse traits (e.g., Nitrogen) of plants related to the plants’ growth using leaf hyperspectral reflectance. Here we use the agriculture data from two domains: Maize (M) and Maize.UNL (MU) (i.e., maize data are measured from different locations). In our case, the task is to predict the Nitrogen content of maize using the leaf hyperspectral reflectance. Specifically, each example can be represented as a 1901-dimensional feature vector, which characterizes the spectral wavelengths 500-2400 nm. Then we can adopt k -NN to build the graph for each domain. We use $k = 5$ in our experiments.

Baselines. The baseline approaches include (1) SourceOnly: GCN [96], SGC [276], GCNII [271]; (2) Feature-only adaptation: DAN [9], DANN [88], MDD [53]; (3) Cross-network adaptation: AdaGCN [144], UDAGCN [143], EGI [54].

Model Configuration. We use two hidden layers in the base GCN model when implementing the GRADE algorithm.

Main Results. Table 4.2 and Table 4.3 provide the cross-network node classification results of GRADE. Here we report the classification accuracy, i.e., mean and standard deviation over 5 runs. We observe that (1) the cross-network node classification algorithms outperform the vanilla graph neural networks and the feature-only adaptation methods, and (2) in most cases, the proposed GRADE algorithm improves the node classification performance (up to 13%) over baselines.

Methods	US $\rightarrow$ BR	US $\rightarrow$ EU	BR $\rightarrow$ US	BR $\rightarrow$ EU	EU $\rightarrow$ US	EU $\rightarrow$ BR	Avg.
GCN	0.366 $\pm$ 0.011	0.371 $\pm$ 0.004	0.491 $\pm$ 0.011	0.452 $\pm$ 0.012	0.439 $\pm$ 0.001	0.298 $\pm$ 0.022	0.403
SGC	0.527 $\pm$ 0.022	0.430 $\pm$ 0.009	0.432 $\pm$ 0.005	0.479 $\pm$ 0.000	0.447 $\pm$ 0.002	0.481 $\pm$ 0.011	0.466
GCNII	0.344 $\pm$ 0.086	0.393 $\pm$ 0.025	0.470 $\pm$ 0.056	0.494 $\pm$ 0.018	0.460 $\pm$ 0.012	0.542 $\pm$ 0.011	0.450
DAN	0.504 $\pm$ 0.020	0.393 $\pm$ 0.000	0.436 $\pm$ 0.006	0.393 $\pm$ 0.010	0.436 $\pm$ 0.003	0.542 $\pm$ 0.000	0.451
DANN	0.500 $\pm$ 0.005	0.386 $\pm$ 0.011	0.402 $\pm$ 0.048	0.350 $\pm$ 0.062	0.436 $\pm$ 0.000	0.538 $\pm$ 0.005	0.435
MDD	0.500 $\pm$ 0.005	0.378 $\pm$ 0.000	0.402 $\pm$ 0.048	0.350 $\pm$ 0.062	0.402 $\pm$ 0.048	0.477 $\pm$ 0.081	0.418
AdaGCN	0.466 $\pm$ 0.065	0.434 $\pm$ 0.004	0.501$\pm$0.003	0.486 $\pm$ 0.021	0.456 $\pm$ 0.034	0.561 $\pm$ 0.081	0.484
UDAGCN	0.607$\pm$0.059	0.388 $\pm$ 0.007	0.497 $\pm$ 0.005	0.510$\pm$0.019	0.434 $\pm$ 0.042	0.477 $\pm$ 0.024	0.486
EGI	0.523 $\pm$ 0.013	0.451 $\pm$ 0.011	0.417 $\pm$ 0.021	0.454 $\pm$ 0.046	0.452 $\pm$ 0.029	0.588$\pm$0.011	0.481
GRADE	0.550 $\pm$ 0.062	0.457$\pm$0.027	0.497 $\pm$ 0.010	0.506 $\pm$ 0.004	0.463$\pm$0.001	0.588$\pm$0.032	0.510

Table 4.2: Cross-network node classification on the Airport network

Methods	Citation		Social	
	A $\rightarrow$ D	D $\rightarrow$ A	B1 $\rightarrow$ B2	B2 $\rightarrow$ B1
GCN	0.435 $\pm$ 0.013	0.567 $\pm$ 0.046	0.408 $\pm$ 0.025	0.451 $\pm$ 0.044
SGC	0.430 $\pm$ 0.001	0.611 $\pm$ 0.000	0.331 $\pm$ 0.110	0.268 $\pm$ 0.059
GCNII	0.465 $\pm$ 0.002	0.559 $\pm$ 0.009	0.392 $\pm$ 0.022	0.473 $\pm$ 0.061
DAN	0.338 $\pm$ 0.005	0.421 $\pm$ 0.060	0.407 $\pm$ 0.015	0.422 $\pm$ 0.015
DANN	0.368 $\pm$ 0.021	0.381 $\pm$ 0.013	0.409 $\pm$ 0.019	0.419 $\pm$ 0.022
MDD	0.349 $\pm$ 0.029	0.391 $\pm$ 0.034	0.388 $\pm$ 0.012	0.421 $\pm$ 0.015
AdaGCN	0.451 $\pm$ 0.013	0.566 $\pm$ 0.042	0.498 $\pm$ 0.057	0.516 $\pm$ 0.025
UDAGCN	0.516$\pm$0.028	0.600 $\pm$ 0.014	0.471 $\pm$ 0.010	0.468 $\pm$ 0.009
EGI	0.489 $\pm$ 0.036	0.404 $\pm$ 0.006	0.494 $\pm$ 0.030	0.516 $\pm$ 0.010
GRADE	0.475 $\pm$ 0.011	0.635$\pm$0.009	0.567$\pm$0.042	0.541$\pm$0.008

Table 4.3: Cross-network node classification on the citation and social networks

Methods	M $\rightarrow$ MU		MU $\rightarrow$ M	
	MAE $\downarrow$	R^2 $\uparrow$	MAE $\downarrow$	R^2 $\uparrow$
GCN	0.489 $\pm$ 0.005	0.132 $\pm$ 0.020	0.679 $\pm$ 0.030	0.295 $\pm$ 0.051
GCNII	0.467 $\pm$ 0.022	0.192 $\pm$ 0.056	0.687 $\pm$ 0.042	0.254 $\pm$ 0.023
DANN	0.492 $\pm$ 0.002	0.104 $\pm$ 0.010	0.721 $\pm$ 0.030	0.192 $\pm$ 0.069
RSD	0.523 $\pm$ 0.011	0.019 $\pm$ 0.061	0.729 $\pm$ 0.009	0.170 $\pm$ 0.047
AdaGCN	0.454 $\pm$ 0.069	0.245 $\pm$ 0.082	0.649$\pm$0.023	0.351 $\pm$ 0.038
UDAGCN	0.449 $\pm$ 0.072	0.257 $\pm$ 0.086	0.684 $\pm$ 0.012	0.271 $\pm$ 0.003
GRADE	0.353$\pm$0.038	0.527$\pm$0.092	0.652 $\pm$ 0.007	0.352$\pm$0.031

Table 4.4: Cross-domain plant phenotyping on the agriculture data set

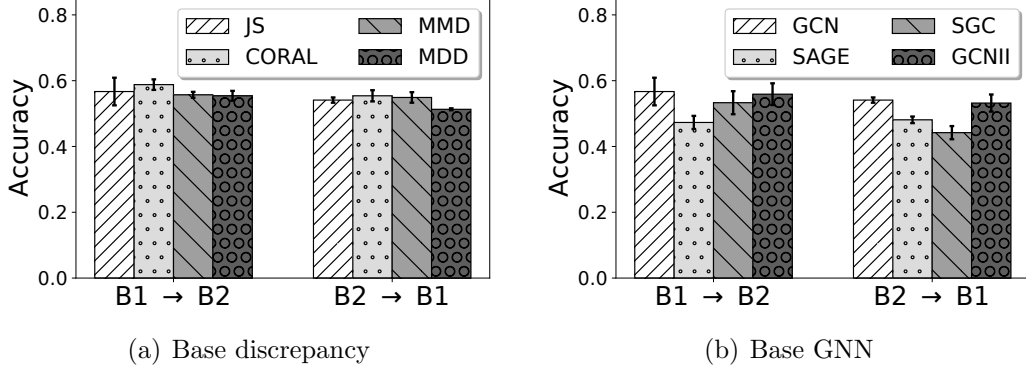


Figure 4.1: GRADE with different base discrepancies and base GNNs on social network

In addition, we investigate the effectiveness of GRADE on the regression task. In this case, we use mean square error (MSE) as the loss function of Eq. (4.13). Table 4.4 provides the results of GRADE on the agriculture data set. Here we use two regression evaluation metrics: mean absolute error (MAE) and R^2 . Since MDD [53] focuses only on the classification setting, we use another state-of-the-art adaptation regression baseline RSD [89] in our experiments. It is observed that our proposed GRADE outperforms the state-of-the-art baselines for both MAE (lower is better) and R^2 (higher is better).

Ablation Study. Figure 4.1 shows the performance of GRADE with different base discrepancies and base graph neural network architectures on social networks. It shows that our GRADE framework is flexible to incorporate existing domain discrepancy measures (i.e., JS-divergence [88], CORAL [215], MMD [115] and MDD [53]) and message-passing graph neural networks (i.e., GCN [96], SAGE [56], SGC [276] and GCNII [271]).

4.2 GRAPH TRANSFER LEARNING WITH HETEROGENEOUS STRUCTURE

In this section, we study a more realistic graph transfer learning problem under the assumption that the source and target graphs may follow different structure assumptions.

4.2.1 Problem Definition

Similar to Problem 4.1, we define the problem of graph transfer learning [14] with a relaxed assumption as follows.

Problem 4.2. Given a source graph $G^s = (V^s, E^s, \mathbf{X}^s)$ and a target graph $G^t = (V^t, E^t, \mathbf{X}^t)$, *graph transfer learning* aims to improve the prediction performance of a graph learning

algorithm on the target graph by leveraging latent knowledge from a source graph, under the assumption that the source and target graphs may follow the different structure assumptions.

The challenge of graph transfer learning induced by the graph structure information can be explained in Figure 4.2. It can be seen that Figure 4.2(a) provides source and target samples associated with input features and output class labels (indicated by node colors). Suppose that source and target distributions have been aligned (e.g., optimized by domain discrepancy minimization algorithms [53, 88]), Figure 4.2(a) implies that source and target domains have similar data distributions in the feature space, thereby enabling feasible knowledge transfer across domains. Compared to Figure 4.2(a), Figure 4.2(b)(c) introduce additional graph structure information. In Figure 4.2(b), source and target graphs follow similar structures where nodes within the same class tend to be connected (i.e., the homophily assumption [57]). In contrast, the structures of source and target graphs fundamentally differ in Figure 4.2(c). That is, the target graph follows the homophily assumption, while the source graph holds the heterophily assumption that connected nodes may have different class labels and dissimilar node attributes [59].

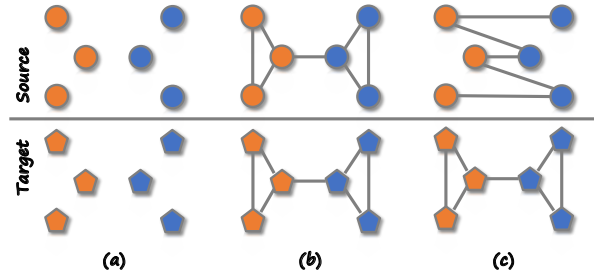


Figure 4.2: Illustration of graph transfer learning. (a) Samples are independently drawn. (b) Samples are connected in each domain, and both domains follow the homophily assumption. (c) Source and target domains follow different graph structure assumptions.

4.2.2 Structure-Aware Neural Network

The graph generation process from a probability distribution space $\mathcal{P}$ can be explained as follows. A graph $G = (V, E)$ is generated by first drawing a graph domain distribution $\mathbb{P}_G$ from $\mathcal{P}$, and then drawing a specific graph G from $\mathbb{P}_G$. To be more specific, one realization from $\mathcal{P}$ is a graph domain distribution $\mathbb{P}_G$ (also denoted as $\mathbb{P}_G = (\mathbb{P}_V, \mathbb{P}_E)$ for node and edge sampling distribution) characterizing a two-stage graph sampling as follows. A set of nodes V is sampled from the graph distribution $\mathbb{P}_G$ (more specifically, sampled from $\mathbb{P}_V$), and then the edge weight is sampled from $\mathbb{P}_G$ (more specifically, sampled from $\mathbb{P}_E$) over any pair of nodes in V . In addition to the graph structure (triggered by the node connections), the output value of a node can also be sampled accordingly in the context of node regression

tasks. That is, a label-informed graph domain distribution $\mathbb{P}_{\mathcal{G}, \mathcal{Y}}$ is sampled from $\mathcal{P}$. Next, a graph $G = (V, E)$ is sampled from $\mathbb{P}_{\mathcal{G}} = (\mathbb{P}_{\mathcal{V}}, \mathbb{P}_{\mathcal{E}})$, and then the output labels of nodes are sampled from $\mathbb{P}_{\mathcal{Y}|\mathcal{G}}$. The goal of node regression task is to learn a prediction function $f(\cdot)$ that predicts $\hat{y}_v = f(v, G)$ for each node $v \in V$.

Under covariate shift assumption [2], the source and target graphs share the same conditional distribution $\mathbb{P}_{\mathcal{Y}|\mathcal{G}}(y|(v, G))$ but different marginal distributions $\mathbb{P}_{\mathcal{G}}(v, G)$. The aforementioned graph sampling process motivates us to learn the input-output relationship between an input sample (G, v) and its associated output y_v for graph transfer learning. To this end, for each input graph G (e.g., source G^s or target graph G^t), we define a structure-aware neural network $f(\cdot, \cdot)$ as follows.

$$f_i^{(l)}(v, G) = \frac{1}{\sqrt{M}} \sum_{j=1}^M \mathbf{W}_{ij}^{(l)} \cdot \mu_j^{(l)}(v|G) \cdot \nu_j^{(l)}(G) \quad (4.15)$$

where $\mu^{(l)}(v|G) = h_v^{(l)}$ and $\nu^{(l)}(G) = \text{READOUT}(\tilde{h}_v^{(l)} | v \in V)$

where M is the layer width. $\mu^{(l)}(v|G)$ denotes the node (sample-level) representation of v given the graph G at the l^{th} layer, and $\nu^{(l)}(G)$ denotes the graph (domain-level) representation of the graph G . Here **READOUT** function compresses a set of node representations into a single vector representation [57]. This definition indicates that the embedding representation of the input (v, G) (e.g., a pair of a node v and its associated graph G) is given by two crucial components: an individual node embedding representation $\mu^{(l)}(v|G)$ and a global graph embedding representation $\nu^{(l)}(G)$. $h_v^{(l)}$ and $\tilde{h}_v^{(l)}$ represent graph neural networks to learn node and graph representations separately.

The intuition behind this definition is explained below. The graph G is a realization of a graph domain distribution $\mathbb{P}_{\mathcal{G}}$. Then, the embedding representation of the graph G can be explained as an empirical estimate of the distribution $\mathbb{P}_{\mathcal{G}}$, i.e., $\nu(G) \approx \nu(\mathbb{P}_{\mathcal{G}})$. Moreover, when the size of the graph G goes to infinity (i.e., associated with an infinite number of nodes), G approximates the true distribution $\mathbb{P}_{\mathcal{G}}$ and the embedding representation of the graph G will be able to recover the true embedding of probability distribution $\mathbb{P}_{\mathcal{G}}$ [115, 277]. Besides, the embedding representation of a node v can also be explained as an approximation of $\mathbb{P}_{\mathcal{G}}$ at the location v , i.e., $\mu(v|G) \approx \mu(v|\mathbb{P}_{\mathcal{G}})$.

Remark. The structure-aware neural network Eq. (4.15) is defined over message-passing graph neural networks commonly used for various single-domain graph learning tasks [57, 58]. Different from previous studies [56, 98, 271, 276], we focus on graph transfer learning scenarios across graphs. This motivates us to consider the task relatedness between source and target graphs, which can be explicitly measured by the domain distribution similar-

ity [18, 88]. To this end, we design the structure-aware neural network Eq. (4.15) for learning domain-aware node presentation, i.e., the integration of the node (sample-level) representation and the entire graph (domain-level) representation. Specifically, we can show the connection between graph domain distribution and graph representation learning in the reproducing kernel Hilbert space for cross-graph learning tasks (see Corollary 4.2).

4.2.3 Graph-Structured Gaussian Process

Inspired by [59, 278, 279], we derive a universal message-passing graph neural network for both homophily and heterophily graphs. More specifically, the message-passing graph convolutional layer for node v within an input graph G can be defined as follows.

$$\begin{aligned} \mathbf{h}_v^{(l)} &= \sum_{i=0}^k \alpha_i \left(\frac{1}{\sqrt{M}} \sum_{u \in N_i(v)} \mathbf{W}_{\text{SANN}}^{(l)} \mathbf{x}_u^{(l)} + \mathbf{b}_{\text{SANN}}^{(l)} \right) \\ \mathbf{x}_u^{(l)} &= \phi(\mathbf{h}_u^{(l-1)}) \end{aligned} \quad (4.16)$$

where $\mathbf{x}_u^{(0)} = \mathbf{x}_u \in \mathbb{R}^D$, $N_0(v) = \{v\}$ denotes the seed node, $\mathbf{W}_{\text{SANN}}^{(l)}, \mathbf{b}_{\text{SANN}}^{(l)}$ are the weight and bias parameters, and α_i explicitly indicates the importance of the i -order neighborhood in finding relevant neighbors around node v . Similarly, we define the domain-level graph neural layer $\tilde{\mathbf{h}}_v^{(l)}$ parameterized by different $\tilde{\mathbf{W}}_{\text{SANN}}^{(l)}, \tilde{\mathbf{b}}_{\text{SANN}}^{(l)}$ and shared α_i .

$$\begin{aligned} \tilde{\mathbf{h}}_v^{(l)} &= \sum_{i=0}^k \alpha_i \left(\frac{1}{\sqrt{M}} \sum_{u \in N_i(v)} \tilde{\mathbf{W}}_{\text{SANN}}^{(l)} \tilde{\mathbf{x}}_u^{(l)} + \tilde{\mathbf{b}}_{\text{SANN}}^{(l)} \right) \\ \tilde{\mathbf{x}}_u^{(l)} &= \phi(\tilde{\mathbf{h}}_u^{(l-1)}) \end{aligned} \quad (4.17)$$

Generally, it is flexible to apply different model architectures to learn the node (sample-level $\mu^{(l)}(v|G)$) and graph (domain-level $\nu^{(l)}(G)$) representations. For simplicity, we adopt the same model architecture but different model parameters to learn node-level and graph-level representations. It is noteworthy that the neighborhood importance α_i is shared because both node and graph representation would share the same assumption (homophily or heterophily) for a given graph. Then, the structure-aware neural network Eq. (4.15) can be

given by

$$f_i^{(l)}(v, G) = \frac{1}{\sqrt{M}} \sum_{j=1}^M \mathbf{W}_{ij}^{(l)} \cdot \mu_j^{(l)}(v|G) \cdot \nu_j^{(l)}(G) \quad (4.18)$$

where $\mu^{(l)}(v|G) = \mathbf{h}_v^{(l)}$ and $\nu^{(l)}(G) = \frac{1}{|V|} \sum_{v \in V} \tilde{\mathbf{h}}_v^{(l)}$

where the **READOUT** function of Eq. (4.15) is instantiated with mean pooling [280]. Different from previous works [59, 278, 279], we would take the neighborhood importance scores α_i as hyper-parameters for building transferable graph Gaussian process model as follows.

Theorem 4.2. Assuming that all the parameters of structure-aware graph neural network $f(v, G)$ are independent and randomly drawn from Gaussian distributions, i.e., $\mathbf{W}^{(l)} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbb{I})$, $\mathbf{b}_{\text{SANN}}^{(l)} \sim \mathcal{N}(\mathbf{0}, \varsigma_b^2 \mathbb{I})$, $\mathbf{W}_{\text{SANN}}^{(l)} \sim \mathcal{N}(\mathbf{0}, \varsigma_w^2 \mathbb{I})$, $\tilde{\mathbf{b}}_{\text{SANN}}^{(l)} \sim \mathcal{N}(\mathbf{0}, \tilde{\varsigma}_b^2 \mathbb{I})$, $\tilde{\mathbf{W}}_{\text{SANN}}^{(l)} \sim \mathcal{N}(\mathbf{0}, \tilde{\varsigma}_w^2 \mathbb{I})$, when the layer width M goes to infinity, the output function $f_i^{(l)}$ in Eq. (4.18) follows a Gaussian process with $f_i^{(l)} \sim \mathcal{GP}(0, \mathcal{K}^{(l)})$, where the covariance function $\mathcal{K}^{(l)}$ is given by

$$\mathcal{K}^{(l)}((v, G), (v', G')) = \sigma_w^2 \cdot \mathcal{K}_\mu^{(l)}(v, v'|G, G') \cdot \mathcal{K}_\nu^{(l)}(G, G') \quad (4.19)$$

where

$$\begin{aligned} \mathcal{K}_\mu^{(l)}(v, v'|G, G') &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_b^2 + \varsigma_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \mathbf{C}_{uu'}^{(l-1)}(\varsigma_w^2, \varsigma_b^2) \right) \\ \mathcal{K}_\nu^{(l)}(G, G') &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\tilde{\varsigma}_b^2 + \frac{\tilde{\varsigma}_w^2}{|V| \cdot |V'|} \mathbf{1}^T \mathbf{P}^{(i)} \mathbf{C}^{(l-1)}(\tilde{\varsigma}_w^2, \tilde{\varsigma}_b^2) (\mathbf{P}'^{(j)})^T \mathbf{1} \right) \\ \mathbf{C}^{(l-1)}(a, b) &= \mathbb{E}_{z_i^{(l-1)} \sim \mathcal{GP}(0, K_{ab}^{(l-1)})} \left[\phi(z_i^{(l-1)}) \phi(z_i^{(l-1)})^T \right] \end{aligned} \quad (4.20)$$

Here $\mathbf{P}^{(i)}$ ($\mathbf{P}'^{(j)}$) denotes the adjacency matrix given by the i -order neighborhood from graph G (the j -order neighborhood from graph G'). $\mathbf{C}_{uu'}^{(0)}(a, b) = \langle \mathbf{x}_u, \mathbf{x}_{u'} \rangle$ for any $u \in V, u' \in V'$, and $K_{ab}^{(l-1)} = \sum_{i,j=0}^k \alpha_i \alpha'_j (b^2 + a^2 \mathbf{P}^{(i)} \mathbf{C}^{(l-2)}(a, b) (\mathbf{P}'^{(j)})^T)$.

Proof. The l -th layer of the function $f(v, G)$ is defined as

$$f_i^{(l)}(v, G) = \frac{1}{\sqrt{M}} \sum_{j=1}^M \mathbf{W}_{ij}^{(l)} \cdot \mu_j^{(l)}(v|G) \cdot \nu_j^{(l)}(G) \quad (4.21)$$

where $\mu^{(l)}(v|G) = \mathbf{h}_v^{(l)}$ and $\nu^{(l)}(G) = \frac{1}{|V|} \sum_{v \in V} \tilde{\mathbf{h}}_v^{(l)}$

Thus, we obtain $\mathbb{E}[f_i^{(l)}(v, G)] = 0$ and

$$\begin{aligned}\mathcal{K}^{(l)}((v, G), (v', G')) &\equiv \mathbb{E}[f_i^{(l)}(v, G)f_i^{(l)}(v', G')] \\ &= \sigma_w^2 \cdot \mathbb{E} \left[\mu_i^{(l)}(v|G) \cdot \mu_i^{(l)}(v'|G') \right] \cdot \mathbb{E} \left[\nu_i^{(l)}(G) \cdot \nu_i^{(l)}(G') \right]\end{aligned}\quad (4.22)$$

Moreover,

$$\begin{aligned}\mathcal{K}_\mu^{(l)}(v, v'|G, G') &\equiv \mathbb{E} \left[\mu_i^{(l)}(v|G) \cdot \mu_i^{(l)}(v'|G') \right] = \mathbb{E} \left[[\mathbf{h}_v^{(l)}]_i \cdot [\mathbf{h}_{v'}^{(l)}]_i \right] \\ &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_b^2 + \varsigma_w^2 \cdot \mathbb{E} \left[\sum_{u \in N_i(v)} [\mathbf{x}_u^{(l)}]_i \sum_{u' \in N_j(v')} [\mathbf{x}_{u'}^{(l)}]_i \right] \right) \\ &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_b^2 + \varsigma_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \mathbb{E} \left[[\mathbf{x}_u^{(l)}]_i [\mathbf{x}_{u'}^{(l)}]_i \right] \right) \\ &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_b^2 + \varsigma_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \mathbf{C}_{uu'}^{(l-1)}(\varsigma_w^2, \varsigma_b^2) \right)\end{aligned}\quad (4.23)$$

where $[\mathbf{a}]_i$ denotes the i -th entry of vector $\mathbf{a}$. Besides,

$$\begin{aligned}\mathcal{K}_\nu^{(l)}(G, G') &\equiv \mathbb{E} \left[\nu_i^{(l)}(G) \cdot \nu_i^{(l)}(G') \right] \\ &= \mathbb{E} \left[\frac{1}{|V|} \sum_{v \in V} [\tilde{\mathbf{h}}_v^{(l)}]_i \cdot \frac{1}{|V'|} \sum_{v' \in V'} [\tilde{\mathbf{h}}_{v'}^{(l)}]_i \right] \\ &= \frac{1}{|V| \cdot |V'|} \sum_{v \in V} \sum_{v' \in V'} \mathbb{E} \left[[\tilde{\mathbf{h}}_v^{(l)}]_i \cdot [\tilde{\mathbf{h}}_{v'}^{(l)}]_i \right] \\ &= \frac{1}{|V| \cdot |V'|} \sum_{v \in V} \sum_{v' \in V'} \left(\sum_{i,j=0}^k \alpha_i \alpha'_j \left(\tilde{\varsigma}_b^2 + \tilde{\varsigma}_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \mathbf{C}_{uu'}^{(l-1)}(\tilde{\varsigma}_w^2, \tilde{\varsigma}_b^2) \right) \right) \\ &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\tilde{\varsigma}_b^2 + \frac{\tilde{\varsigma}_w^2}{|V| \cdot |V'|} \mathbf{1}^T \mathbf{P}^{(i)} \mathbf{C}^{(l-1)}(\tilde{\varsigma}_w^2, \tilde{\varsigma}_b^2) (\mathbf{P}'^{(j)})^T \mathbf{1} \right)\end{aligned}\quad (4.24)$$

which completes the proof. QED.

In the following, we show that the sample-level covariance/kernel $\mathcal{K}_\mu^{(l)}$ of Theorem 4.2 can be explained as a message-passing operation in the kernel space. Following [281, 282], when using ReLU as activation function $\phi(\cdot)$ (i.e., $\phi(x) = \max\{0, x\}$), $\mathbf{C}^{(l-1)}$ is the arc-cosine

kernel [283] as follows.

$$\mathbf{C}_{vv'}^{(l-1)}(\varsigma_w^2, \varsigma_b^2) = \frac{\kappa_1(\zeta)}{2} \sqrt{\mathcal{K}_\mu^{(l-1)}(v, v|G) \cdot \mathcal{K}_\mu^{(l-1)}(v', v'|G')} \quad (4.25)$$

where $\zeta = \frac{\mathcal{K}_\mu^{(l-1)}(v, v'|G, G')}{\sqrt{\mathcal{K}_\mu^{(l-1)}(v, v|G) \cdot \mathcal{K}_\mu^{(l-1)}(v', v'|G')}}}$ and $\kappa_1(\zeta) = \frac{1}{\pi} \left(\zeta \cdot (\pi - \arccos(\zeta)) + \sqrt{1 - \zeta^2} \right)$. Then, the following corollary derives the feature map of $\mathcal{K}_\mu^{(l)}(v, v'|G, G')$ in the kernel space.

Corollary 4.1. Let $\varphi_1 : \mathcal{H} \rightarrow \mathcal{H}_1$ denote the kernel mapping from a pre-activation RKHS $\mathcal{H}$ to post-activation RKHS $\mathcal{H}_1$, i.e., $\langle \varphi_1(s), \varphi_1(s') \rangle = \frac{1}{2} \|s\| \cdot \|s'\| \cdot \kappa_1 \left(\frac{\langle s, s' \rangle}{\|s\| \cdot \|s'\|} \right)$. Given the sample-level kernel $\mathcal{K}_\mu^{(l)}(v, v'|G, G')$ in Theorem 4.2, if the graph convolutional layer of Eq. (4.16) has no bias term ($\varsigma_b = 0$), the feature map of this kernel is given by

$$\Psi_v^{(l)} = \varsigma_w \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1(\Psi_u^{(l-1)}) \quad \text{and} \quad \Psi_{v'}^{(l)} = \varsigma_w \sum_{i=0}^k \alpha'_i \sum_{u' \in N_i(v')} \varphi_1(\Psi_{u'}^{(l-1)}) \quad (4.26)$$

with $\mathcal{K}_\mu^{(l)}(v, v'|G, G') = \langle \Psi_v^{(l)}, \Psi_{v'}^{(l)} \rangle$.

Proof. Based on Theorem 4.2, we have

$$\mathbf{C}_{vv'}^{(l-1)} = \frac{\kappa_1(\zeta)}{2} \sqrt{\mathcal{K}_\mu^{(l-1)}(v, v|G) \cdot \mathcal{K}_\mu^{(l-1)}(v', v'|G')} = \left\langle \varphi_1(\Psi_v^{(l-1)}), \varphi_1(\Psi_{v'}^{(l-1)}) \right\rangle \quad (4.27)$$

Then, with $\varsigma_b = 0$, we have

$$\begin{aligned} \mathcal{K}_\mu^{(l)}(v, v'|G, G') &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \mathbf{C}_{uu'}^{(l-1)}(\varsigma_w^2, \varsigma_b^2) \right) \\ &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\varsigma_w^2 \cdot \sum_{u \in N_i(v)} \sum_{u' \in N_j(v')} \left\langle \varphi_1(\Psi_u^{(l-1)}), \varphi_1(\Psi_{u'}^{(l-1)}) \right\rangle \right) \\ &= \left\langle \varsigma_w \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1(\Psi_u^{(l-1)}), \varsigma_w \sum_{j=0}^k \alpha'_j \sum_{u' \in N_j(v')} \varphi_1(\Psi_{u'}^{(l-1)}) \right\rangle \end{aligned} \quad (4.28)$$

Therefore, it holds that

$$\Psi_v^{(l)} = \varsigma_w \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1(\Psi_u^{(l-1)}) \quad \text{and} \quad \Psi_{v'}^{(l)} = \varsigma_w \sum_{j=0}^k \alpha'_j \sum_{u' \in N_j(v')} \varphi_1(\Psi_{u'}^{(l-1)}) \quad (4.29)$$

This result provides the feature map for nodes from graph G and G' separately. But we show

that it is equivalent to the following unified feature mapping on a large graph integrating G and G' . More specifically, graphs $G = (V, E)$ and $G' = (V', E')$ can be considered as a single large graph $G'' = (V \cup V', E \cup E')$ with two disjoint components. Then, for any node $v'' \in V''$ in graph G'' , we have

$$\Psi_{v''}^{(l)} = \varsigma_w \sum_{i=0}^k (\alpha_i \cdot \mathbb{1}[v'' \in V] + \alpha'_i \cdot \mathbb{1}[v'' \in V']) \sum_{u'' \in N_j(v'')} \varphi_1 \left(\Psi_{u''}^{(l-1)} \right) \quad (4.30)$$

where v'' can be a node from either G or G' , and $\mathbb{1}[\cdot]$ is an indicator function that returns 1 if the argument is true, 0 otherwise. QED.

Corollary 4.1 shows that the feature map $\Psi_v^{(l)}$ of sample-level kernel $\mathcal{K}_\mu^{(l)}$ is a kernelized version of the graph convolutional layer in Eq. (4.16), which recursively aggregates message from the nodes' neighborhood in the kernel space. The neighborhood can be automatically selected by optimizing the hyper-parameters α_i and α'_i .

Next, we show the implication of domain-level kernel $\mathcal{K}_\nu^{(l)}(G, G')$, by showing the connection between the empirical domain representation $\nu^{(l)}(G)$ and the embedding of probability distribution $\mathbb{P}_G$ [115, 277].

Definition 4.4 (Mean Embedding of Probability Distributions [115]). Given an RKHS $\mathcal{H}$ induced by a kernel $\mathcal{K}(\cdot, \cdot)$, the mean embedding of a probability distribution $\mathbb{P}$ is defined as $\tau_{\mathbb{P}} = \int \mathcal{K}(\cdot, x) d\mathbb{P}(x)$, i.e., $\mathbb{E}_{x \sim \mathbb{P}}[f(x)] = \langle f, \tau_{\mathbb{P}} \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. Furthermore, the empirical mean embedding of $\mathbb{P}$ is given by $\hat{\tau}_{\mathbb{P}} = \frac{1}{n} \sum_{i=1}^n \mathcal{K}(\cdot, x_i)$, where n samples are independent and identically drawn from $\mathbb{P}$.

Analogously, the mean embedding of a graph domain distribution $\mathbb{P}_G$ can also be defined as $\tau_{\mathbb{P}_G} = \int \mathcal{K}_G(\cdot, v | \mathbb{P}_G) d\mathbb{P}_G(v)$ given an RKHS $\mathcal{H}_G$ induced by a kernel $\mathcal{K}_G(\cdot, \cdot)$. In the context of graph learning, $v | \mathbb{P}_G$ indicates the individual node attributes of v and graph structure information around node v induced by $\mathbb{P}_G$. It subsumes the conventional mean embedding [115] of a probability distribution $\tau_{\mathbb{P}}$ with independent and identically distributed (IID) samples $x \sim \mathbb{P}$, if there are no edges in the graph (i.e., no graph structure is involved around node v). It is observed that the additional graph structure challenges empirical estimation of graph domain distribution $\mathbb{P}_G$, thereby resulting in a nontrivial domain similarity estimator between source and target graphs in graph transfer learning. The following corollary states that the graph representation $\nu^{(l)}(G)$ can be considered as the empirical mean embedding of $\tau_{\mathbb{P}_G}$ in the RKHS induced by $\mathcal{K}_\nu^{(l)}(\cdot, \cdot)$.

Corollary 4.2. With the same conditions in Theorem 4.2, for each l , in the limit on the layer width, the graph representation $\nu^{(l)}(G)$ recovers the empirical mean embedding $\hat{\tau}_{\mathbb{P}_G}^{(l)}$ of

the domain distribution $\mathbb{P}_G$ in the reproducing kernel Hilbert space induced by $\mathcal{K}_\nu^{(l)}$, and $\hat{\tau}_{\mathbb{P}_G}^{(l)}$ is given by

$$\hat{\tau}_{\mathbb{P}_G} = \frac{1}{|V|} \sum_{v \in V} \tilde{\Psi}_v^{(l)} = \frac{1}{|V|} \sum_{v \in V} \mathcal{K}_\nu^{(l)}(\cdot, v|G) \quad (4.31)$$

where $\tilde{\Psi}_v^{(l)}$ is the feature map of $\mathcal{K}_\nu^{(l)}$ with $\tilde{\Psi}_v^{(l)} = \tilde{\zeta}_w \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1 \left(\tilde{\Psi}_u^{(l-1)} \right)$.

Proof. We derive the feature map of domain-level kernel $\mathcal{K}_\nu^{(l)}$ as follows. Considering $\tilde{\zeta}_b = 0$, we have

$$\begin{aligned} \mathcal{K}_\nu^{(l)}(G, G') &= \sum_{i,j=0}^k \alpha_i \alpha'_j \left(\frac{\tilde{\zeta}_w^2}{|V| \cdot |V'|} \mathbf{1}^T \mathbf{P}^{(i)} \mathbf{C}^{(l-1)}(\tilde{\zeta}_w^2, \tilde{\zeta}_b^2) (\mathbf{P}'^{(j)})^T \mathbf{1} \right) \\ &= \left\langle \frac{\tilde{\zeta}_w}{|V|} \sum_{v \in V} \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1 \left(\tilde{\Psi}_u^{(l-1)} \right), \frac{\tilde{\zeta}_w}{|V'|} \sum_{v' \in V'} \sum_{j=0}^k \alpha'_j \sum_{u' \in N_j(v')} \varphi_1 \left(\tilde{\Psi}_{u'}^{(l-1)} \right) \right\rangle \end{aligned} \quad (4.32)$$

That is, given the node representation $\tilde{\Psi}_v^{(l)} = \tilde{\zeta}_w \sum_{i=0}^k \alpha_i \sum_{u \in N_i(v)} \varphi_1 \left(\tilde{\Psi}_u^{(l-1)} \right)$ in the kernel space, the mean pooling in the **READOUT** function (see Eq.(4.15)) results in the mean embedding of empirical node distribution over $\{\tilde{\Psi}_v^{(l)}\}_{v \in V}$, i.e.,

$$\hat{\tau}_{\mathbb{P}_G} = \frac{1}{|V|} \sum_{v \in V} \tilde{\Psi}_v^{(l)} \quad (4.33)$$

Moreover, when the sample size of graph G goes to infinity, the expected mean embedding of node distribution over $\{\tilde{\Psi}_v^{(l)}\}_{v \in V}$ is given by

$$\tau_{\mathbb{P}_G} = \mathbb{E}_v \left[\tilde{\Psi}_v^{(l)} \right] = \mathbb{E}_v \left[\mathcal{K}_\nu^{(l)}(\cdot, v|G) \right] = \mathbb{E}_v \left[\mathcal{K}_\nu^{(l)}(\cdot, v|\mathbb{P}_G) \right] \quad (4.34)$$

which corresponds to expression of graph domain distribution $\mathbb{P}_G$ in the reproducing kernel Hilbert space induced by $\mathcal{K}_\nu^{(l)}$ (given by the kernel mapping $\tilde{\Psi}_v^{(l)}$). **QED.**

Corollary 4.2 shows that $\mathcal{K}_\nu^{(l)}(G, G')$ in Theorem 4.2 can be explained as the similarity of the empirical mean embeddings of domain distributions $\mathbb{P}_G$ and $\mathbb{P}_{G'}$ in the kernel space e.g., $\mathcal{K}_\nu^{(l)}(G, G') = \langle \hat{\tau}_{\mathbb{P}_G}^{(l)}, \hat{\tau}_{\mathbb{P}_{G'}}^{(l)} \rangle_{\mathcal{H}_{\mathcal{K}_\nu^{(l)}}}$. A similar metric over distributions is the maximum mean discrepancy (MMD) [115] defined over the distance of empirical mean embeddings, i.e., $\text{MMD}_\nu^{(l)} = \|\hat{\tau}_{\mathbb{P}_G}^{(l)} - \hat{\tau}_{\mathbb{P}_{G'}}^{(l)}\|_{\mathcal{H}_{\mathcal{K}_\nu^{(l)}}}$, where $\mathcal{H}_{\mathcal{K}_\nu^{(l)}}$ denotes the RKHS induced by $\mathcal{K}_\nu^{(l)}$. MMD has been widely applied to transfer learning [9, 116] for measuring the distribution shift across image domains, under the covariate shift assumption [2] (i.e., source and target domains share conditional distribution $p(y|x)$ but different marginal distributions $p(x)$). In contrast, we focus on measuring the domain distribution similarity for graph transfer learning [14].

4.2.4 Instantiated Algorithms

Given a source graph $G^s = (V^s, E^s)$ with fully labeled nodes (e.g., $y_v \in \mathbb{R}$ is associated with each node $v \in V^s$) and a target graph $G^t = (V^t, E^t)$ with a limited number of labeled nodes (e.g., $|V_l^t| \ll |V^s|$ where $V_l^t \subset V^t$ is the set of labeled target nodes), we propose the adaptive graph Gaussian process algorithm (termed as **GraphGP**) as follows [14].

For notation simplicity, we let $f_v = f^{(L)}(v, G)$ (given by L graph convolutional layers in Eq. (4.18)) be the function value at node v . Let $\mathbf{f}^s = [f_1^s, f_2^s, \dots, f_{n^s}^s]^T$ be a vector of latent function values over labeled source nodes, and $\mathbf{y}^s = [y_1^s, y_2^s, \dots, y_{n^s}^s]^T$ be the associated ground-truth output values. Similarly, we can define $\mathbf{f}^t$ and $\mathbf{y}^t$ over target nodes. Then, the GP prior over the function values can be defined as $f \sim \mathcal{GP}(0, \mathcal{K}^{(L)})$ (defined in Theorem 4.2), and its instantiation at labeled training nodes is given by $p(\mathbf{f}|V^s \cup V_l^t) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{(s+t)(s+t)})$ where $\mathbf{K}_{(s+t)(s+t)}$ is a block matrix, i.e., $\mathbf{K}_{(s+t)(s+t)} = \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{st} \\ \mathbf{K}_{ts} & \mathbf{K}_{tt} \end{bmatrix}$ and its entry is $[\mathbf{K}_{ab}]_{v^a v^b} = \mathcal{K}^{(L)}((v^a, G^a), (v^b, G^b))$ for $a, b \in \{s, t\}$. For the likelihood, we consider the noisy scenarios where $p(\mathbf{y}^a|\mathbf{f}^a) = \mathcal{N}(\mathbf{f}^a, \varrho_a^2 \mathbb{I})$ for $a \in \{s, t\}$ (ϱ_a measures the noisy magnitude). Following standard Gaussian process [214], the posterior distribution of **GraphGP** over testing nodes $v_*^t \in V^t$ has a closed-form expression, i.e., $p(\mathbf{f}_*^t|v_*^t, \mathbf{f}, V^s \cup V_l^t) = \mathcal{N}(\gamma, \Gamma)$ where

$$\begin{aligned} \gamma(v_*^t, G^t) &= \mathbf{k}_* \left(\mathbf{K}_{(s+t)(s+t)} + \begin{bmatrix} \varrho_s^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} \right)^{-1} \mathbf{y} \\ \Gamma((v_*^t, G^t), (v_*^t, G^t)) &= \mathbf{k}_{**} - \mathbf{k}_* \left(\mathbf{K}_{(s+t)(s+t)} + \begin{bmatrix} \varrho_s^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} \right)^{-1} \mathbf{k}_*^T \end{aligned} \quad (4.35)$$

Here, $\mathbf{y} = \begin{bmatrix} \mathbf{y}^s \\ \mathbf{y}_l^t \end{bmatrix}$ denotes the ground-truth output values of labeled nodes $V^s \cup V_l^t$ from source and target graphs. Each entry of $\mathbf{k}_*$ represents the covariance between testing target node and training node, and $\mathbf{k}_{**}$ denotes the covariance matrix over testing target nodes.

The posterior distribution of **GraphGP** has the following hyper-parameters: $\sigma_w, \varsigma_w, \varsigma_b, \alpha_i^s, \alpha_j^t$, and noise variances ϱ_s, ϱ_t . The hyper-parameters of the Gaussian process can be optimized by maximizing the marginal likelihood $p(\mathbf{y})$ over all the training samples [214]. However, in the context of graph transfer learning, the number of labeled target nodes is much smaller than the number of labeled source nodes. By directly maximizing the marginal likelihood over all labeled source and target nodes, **GraphGP** might be biased towards the source domain. Therefore, we propose to optimize the marginal distribution $p(\mathbf{y}_l^t)$ over labeled target nodes by considering all the nodes $V^s \cup V_l^t$ as the inducing points [284]. The

objective function is given by maximizing the following log marginal likelihood

$$\log p(\mathbf{y}_l^t) = \log \left[\mathcal{N} \left(\mathbf{y}_l^t | \mathbf{0}, \mathbf{K}_{t(s+t)} \mathbf{K}_{(s+t)(s+t)}^{-1} \mathbf{K}_{t(s+t)}^T + \varrho_t^2 \mathbb{I} \right) \right] \quad (4.36)$$

4.2.5 Theoretical Analysis

We define the generalization error of the target graph for transferable node regression. Given the ground-truth labeling function f^* in the target domain and the estimator $f^{(L)}$ learned from observed source and target graphs, the generalization error is given by

$$\mathcal{E}^t = \int \int \mathcal{L}(\gamma(V^t, G^t), f^*(V^t, G^t)) p(V^t, G^t) p(f^*) d(V^t, G^t) df^* \quad (4.37)$$

where $\gamma(\cdot)$ is the posterior mean of $f^{(L)}(\cdot)$ in Eq. (4.35), $\mathcal{L}$ is the mean squared loss function, and $p(V^t, G^t)$ denotes the sampling probability of a target input (V^t, G^t) . It is shown [214, 285] that if the GP prior is correctly specified, i.e., the predictor $f^{(L)}$ has the same prior as f^* , the generalization error is given by the predictive variance of the GP. That is, $\epsilon_t = \int \Gamma(V^t, G^t) p(V^t, G^t) d(V^t, G^t)$ where $\Gamma(V^t, G^t)$ is the posterior variance at (V^t, G^t) in Eq. (4.35). Furthermore, the following theorem shows that the generalization error can be upper bounded in terms of the variance that takes all inputs (V^s, G^s) of the source graph as the target samples, i.e., both (V^s, G^s) and (V^t, G^t) are assumed to follow the same target distribution for single-domain graph learning.

Theorem 4.3. Let $\mathbf{K}_{ss}^\mu$ be the sample-level covariance matrix over source nodes, i.e., $[\mathbf{K}_{ss}^\mu]_{(v_1^s, v_2^s)} = \mathcal{K}_\mu^{(L)}(v_1^s, v_2^s | G^s)$, $\nu_{ss} = \mathcal{K}_\nu^{(L)}(G^s, G^s)$, $\nu_{tt} = \mathcal{K}_\nu^{(L)}(G^t, G^t)$ be the intra-graph kernels and $\nu_{st} = \mathcal{K}_\nu^{(L)}(G^s, G^t)$ be the inter-graph kernel. Suppose $\bar{\varrho}_s^2 \triangleq \left(\frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} - 1 \right) \sigma_w^2 \nu_{tt} \bar{\lambda}_{ss} + \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2$ where $\bar{\lambda}_{ss}$ is the maximum eigenvalue of $\mathbf{K}_{ss}^\mu$, for any (V^t, G^t) the generalization error is bounded by

$$\mathcal{E}^t \leq \int \Gamma_t(V^t, G^t; \bar{\varrho}_s^2, \varrho_t^2) p(V^t, G^t) d(V^t, G^t) \quad (4.38)$$

where $\Gamma_t(V^t, G^t; \bar{\varrho}_s^2, \varrho_t^2)$ is the variance assuming that all source samples are observed in the target domain w.r.t. to noises $\bar{\varrho}_s^2, \varrho_t^2$.

Proof. The covariance matrix $\mathbf{K}$ over training samples can be rewritten as

$$\mathbf{K} = \begin{bmatrix} \sigma_w^2 \nu_{ss} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{st} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{st} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} = \begin{bmatrix} \frac{\nu_{st}}{\nu_{tt}} \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} \end{bmatrix} \begin{bmatrix} \frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{tt} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} \begin{bmatrix} \frac{\nu_{st}}{\nu_{tt}} \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} \end{bmatrix} \quad (4.39)$$

Then

$$\Gamma(v^t, G^t) = \mathbf{k}_{**} - \mathbf{k}_* (\mathbf{K}_{ss} + \varrho_s^2 \mathbb{I})^{-1} \mathbf{k}_*^T = \mathbf{k}_{**} - \mathbf{k}_*^t (\Sigma)^{-1} (\mathbf{k}_*^t)^T \quad (4.40)$$

Following Proposition 7 of [285], we have

$$\Delta(v^t, G^t) = \mathbf{k}_*^t (\Sigma_t^{-1} - \Sigma^{-1}) (\mathbf{k}_*^t)^T \quad (4.41)$$

where

$$\begin{aligned} \Sigma &= \begin{bmatrix} \frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{tt} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} + \begin{bmatrix} \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} \\ \Sigma_t &= \begin{bmatrix} \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{tt} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} + \begin{bmatrix} a^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} \end{aligned} \quad (4.42)$$

Thus, it holds

$$\begin{aligned} \Delta(v^t, G^t) &\leq 0 \\ \Sigma_t^{-1} - \Sigma^{-1} &\preceq 0 \\ \Sigma_t &\succeq \Sigma \\ \begin{bmatrix} \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{tt} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} + \begin{bmatrix} a^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} &\succeq \begin{bmatrix} \frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{st}^\mu \\ \sigma_w^2 \nu_{tt} \mathbf{K}_{ts}^\mu & \sigma_w^2 \nu_{tt} \mathbf{K}_{tt}^\mu \end{bmatrix} + \begin{bmatrix} \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \varrho_t^2 \mathbb{I} \end{bmatrix} \\ \begin{bmatrix} \left(1 - \frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}}\right) \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} &\succeq \begin{bmatrix} \left(\frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 - a^2\right) \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \left(1 - \frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}}\right) \sigma_w^2 \nu_{tt} \mathbf{K}_{ss}^\mu &\succeq \left(\frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 - a^2\right) \mathbb{I} \\ \mathbf{K}_{ss}^\mu &\preceq \frac{1}{\left(\frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} - 1\right) \sigma_w^2 \nu_{tt}} \left(a^2 - \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2\right) \mathbb{I} \\ \bar{\lambda}_{ss} &\leq \frac{1}{\left(\frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} - 1\right) \sigma_w^2 \nu_{tt}} \left(a^2 - \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2\right) \\ a^2 &\geq \left(\frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} - 1\right) \sigma_w^2 \nu_{tt} \bar{\lambda}_{ss} + \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 \end{aligned} \quad (4.43)$$

Thus, the minimum of the upper bound is given by

$$\bar{\varrho}_s^2 \triangleq \left(\frac{\nu_{ss} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} - 1\right) \sigma_w^2 \nu_{tt} \bar{\lambda}_{ss} + \frac{\nu_{tt} \cdot \nu_{tt}}{\nu_{st} \cdot \nu_{st}} \varrho_s^2 \quad (4.44)$$

which completes the proof.

QED.

This theorem confirms that compared to the single-domain Gaussian process (assuming all inputs are drawn from the same distribution), **GraphGP** enables the reduced generalization error. In addition, the following corollary considers a special scenario where no labeled nodes are available in the target graph. In this case, the generalization error is determined by the normalized graph domain similarity $\frac{\mathcal{K}_\nu^{(L)}(G^s, G^t)}{\sqrt{\mathcal{K}_\nu^{(L)}(G^s, G^s) \cdot \mathcal{K}_\nu^{(L)}(G^t, G^t)}}$.

Corollary 4.3. When there are no labeled nodes in the target graph, i.e., $V_l^t = \emptyset$, we have

$$\lim_{n^s \rightarrow \infty} \mathcal{E}^t = \left(1 - \frac{\nu_{st}^2}{\nu_{tt}\nu_{ss}}\right) \int \mathcal{K}^{(L)}((V^t, G^t), (V^t, G^t)) p(V^t, G^t) d(V^t, G^t) \quad (4.45)$$

where $\nu_{ss} = \mathcal{K}_\nu^{(L)}(G^s, G^s)$, $\nu_{tt} = \mathcal{K}_\nu^{(L)}(G^t, G^t)$ and $\nu_{st} = \mathcal{K}_\nu^{(L)}(G^s, G^t)$.

Proof. It is given $\mathcal{K}^{(l)}((v, G), (v', G')) = \sigma_w^2 \mathcal{K}_\mu^{(l)}(v, v' | G, G') \mathcal{K}_\nu^{(l)}(G, G')$. For all (v^t, G^t) , it holds

$$\begin{aligned} \Gamma(v^t, G^t) &= \mathbf{k}_{**} - \mathbf{k}_* (\mathbf{K}_{ss} + \varrho_s^2 \mathbb{I})^{-1} \mathbf{k}_*^T \\ &= \sigma_w^2 \nu_{tt} \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) - (\sigma_w^2 \nu_{st} \mathbf{k}_{*\mu}) (\mathbf{K}_{ss} + \varrho_s^2 \mathbb{I})^{-1} (\sigma_w^2 \nu_{st} \mathbf{k}_{*\mu})^T \\ &= \sigma_w^2 \left(\nu_{tt} - \frac{\nu_{st}^2}{\nu_{ss}} \right) \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) \\ &\quad + \frac{\nu_{st}^2}{\nu_{ss}^2} \left[\sigma_w^2 \nu_{ss} \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) - (\sigma_w^2 \nu_{ss} \mathbf{k}_{*\mu}) (\mathbf{K}_{ss} + \varrho_s^2 \mathbb{I})^{-1} (\sigma_w^2 \nu_{ss} \mathbf{k}_{*\mu})^T \right] \\ &= \sigma_w^2 \left(\nu_{tt} - \frac{\nu_{st}^2}{\nu_{ss}} \right) \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) + \frac{\nu_{st}^2}{\nu_{ss}^2} \Gamma_s(v^t, G^t) \end{aligned} \quad (4.46)$$

where $\Gamma_s(v^t, G^t)$ is the posterior variance by considering (v^t, G^t) as a virtual source sample. Then when the number of source samples goes to infinity, we have

$$\lim_{n^s \rightarrow \infty} \Gamma_s(v^t, G^t) = 0 \quad (4.47)$$

Thus,

$$\begin{aligned} \Gamma(v^t, G^t) &= \sigma_w^2 \left(\nu_{tt} - \frac{\nu_{st}^2}{\nu_{ss}} \right) \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) \\ &= \left(1 - \frac{\nu_{st}^2}{\nu_{tt}\nu_{ss}} \right) \sigma_w^2 \nu_{tt} \mathcal{K}_\mu^{(l)}(v^t, v^t | G^t) \\ &= \left(1 - \frac{\nu_{st}^2}{\nu_{tt}\nu_{ss}} \right) \mathcal{K}^{(L)}((v^t, G^t), (v^t, G^t)) \end{aligned} \quad (4.48)$$

where $\nu_{ss} = \mathcal{K}_\nu^{(L)}(G_s, G_s)$, $\nu_{tt} = \mathcal{K}_\nu^{(L)}(G^t, G^t)$ denote the intra-graph kernels and $\nu_{st} =$

$\mathcal{K}_\nu^{(L)}(G_s, G^t)$ is the inter-graph kernel.

$$\begin{aligned} \lim_{n^s \rightarrow \infty} \mathcal{E}^t &= \int \Gamma(v^t, G^t) p(v^t, G^t) d(v^t, G^t) \\ &= \left(1 - \frac{\nu_{st}^2}{\nu_{tt}\nu_{ss}}\right) \int \mathcal{K}^{(L)}((v^t, G^t), (v^t, G^t)) p(v^t, G^t) d(v^t, G^t) \end{aligned} \quad (4.49)$$

which completes the proof. QED.

4.2.6 Experiment

In the experiments, we evaluate the proposed **GraphGP** algorithm for node regression tasks. Specifically, we compare **GraphGP** with state-of-the-art Graph Gaussian processes for graph transfer learning, assuming that the source and target graphs may have different structure assumptions.

Data Sets: The following graph learning benchmarks are used.

- Twitch [286]: It has 6 different domains ("DE", "EN", "ES", "FR", "PT", and "RU"). In each domain, the graph represents the friendships among Twitch users, where each node corresponds to a Twitch user and each edge represents the mutual friendship of two users. Each node is also associated with an attribute vector encoding the game, location, and streaming habit information. Different from previous work [286] which considers the binary classification of whether a streamer uses explicit language, in this paper, we focus on the node regression task. Thus, we use the number of views in the original data as the output value for each node/user.
- Agriculture [48, 287]: It has 3 different domains ("Maize" (MA), "Sorghum" (SG), and "Soybean" (SY)). The goal of agriculture analysis is to predict diverse traits (e.g., specific leaf area) of plants related to the plants' growth using leaf hyperspectral reflectance. The knowledge transferability across different species allows us to effectively infer the biochemical traits in real scenarios. Each data point represents a plant associated with a 1901-dimensional feature vector (e.g., spectral wavelengths 500-2400 nm). Following [24], we use k -NN to build the graph based on the feature similarity.
- Airports [272]: It has 3 different domains ("USA" (US), "Brazil" (BR), and "Europe" (EU)). Those domains involve the airport networks where each node corresponds to an airport and each edge represents the existence of commercial flights. Each node/airport is assigned a label based on their level of activity, measured in flights or people. Following [54], we use the degree-guided feature vector for each node.

Model	Airport graphs			Agriculture graphs		
	BR $\rightarrow$ EU	EU $\rightarrow$ BR	BR $\rightarrow$ US	MA $\rightarrow$ SG	SG $\rightarrow$ MA	MA $\rightarrow$ SY
RBFGP	0.4849 $\pm$ 0.0260	0.4479 $\pm$ 0.0211	0.3682 $\pm$ 0.0301	0.4859 $\pm$ 0.0716	0.3359 $\pm$ 0.0297	0.7314 $\pm$ 0.0172
DINO	0.5241 $\pm$ 0.0147	0.4855 $\pm$ 0.0337	0.3877 $\pm$ 0.0263	0.6227 $\pm$ 0.0223	0.4591 $\pm$ 0.0445	0.7620 $\pm$ 0.0125
GGP	0.3400 $\pm$ 0.0144	0.3990 $\pm$ 0.0401	0.4720 $\pm$ 0.0218	0.4515 $\pm$ 0.0099	0.2403 $\pm$ 0.0172	0.7420 $\pm$ 0.0425
SAGEGP	0.4581 $\pm$ 0.0308	0.3822 $\pm$ 0.0508	0.4928 $\pm$ 0.0330	0.5348 $\pm$ 0.0278	0.3633 $\pm$ 0.0101	0.7632 $\pm$ 0.0349
GINGP	0.5216 $\pm$ 0.0227	0.4471 $\pm$ 0.0219	0.4901 $\pm$ 0.0255	0.5380 $\pm$ 0.0288	0.3559 $\pm$ 0.0196	0.7746 $\pm$ 0.0518
GraphGP	0.5567$\pm$0.0246	0.4983$\pm$0.0370	0.5293$\pm$0.0335	0.6586$\pm$0.0244	0.5125$\pm$0.0171	0.7921$\pm$0.0168

Table 4.5: Graph transfer learning on Airport and Agriculture

- Wikipedia [286]: It has 3 different domains (“chameleon” (CH), “crocodile” (CR), and “squirrel” (SQ)). As introduced in [286], those domain data are given by Wikipedia page-page networks on different topics: chameleons, crocodiles, and squirrels. Each node corresponds to an article and each edge represents mutual links of two articles. The node feature is given by the presence of particular nouns in the articles and the average monthly traffic. It aims to predict the log average monthly traffic.
- WebKB [288]: It has 3 different domains (“Cornell” (CO), “Texas” (TX), and “Wisconsin” (WS)). It is collected from computer science departments of various universities by Carnegie Mellon University. Each node corresponds to a web page, and each edge represents the hyperlink of two web pages. The node features are given by the bag-of-words representation of web pages. The web pages are manually classified into five categories, student, project, course, staff, and faculty. In contrast to previous works [288], we would like to consider the regression task where each node is associated with an output value from $[0, 4]$, e.g., 4.0 for nodes from class 4.

The train/validation/test split in our experiments is given below. For transferable node regression, it is given a source graph $G^s = (V^s, E^s)$ with fully labeled nodes and a target graph $G^t = (V^t, E^t)$ with a limited number of labeled nodes (e.g., $|V_l^t| \ll |V^s|$ where $V_l^t \subset V^t$ is the set of labeled target nodes). Therefore, for Airport, Wikipedia, and WebKB data sets, we randomly select 10% of target nodes for the training set, 10% for the validation set, and 80% for the testing set. For Agriculture and Twitch data sets, we randomly select 1% of target nodes for the training set, 1% for the validation set, and 98% for the testing set.

Baselines. We consider the following Gaussian process baselines. (1) RBFGP [214] and DINO [18] are feature-Only Gaussian processes without using graph structures. (2) GGP [289], SAGEGP [282], and GINGP [282] are graph Gaussian processes by considering source and target graphs as a large disjoint graph. (3) LINKXGP, MixHopGP, and H2GCNGP are graph Gaussian processes derived from LINKX [290], MixHop [291], H2GCN [59] respec-

Model	PT $\rightarrow$ RU	EN $\rightarrow$ PT	RU $\rightarrow$ ES	RU $\rightarrow$ PT	PT $\rightarrow$ EN	ES $\rightarrow$ RU
RBFGP	0.4324 $\pm$ 0.0087	0.5430 $\pm$ 0.0117	0.4241 $\pm$ 0.0024	0.4962 $\pm$ 0.0055	0.5507 $\pm$ 0.0024	0.4738 $\pm$ 0.0059
DINO	0.6498 $\pm$ 0.0059	0.7565 $\pm$ 0.0140	0.6717 $\pm$ 0.0114	0.7641 $\pm$ 0.0082	0.5920 $\pm$ 0.0195	0.5636 $\pm$ 0.0072
GGP	0.3167 $\pm$ 0.0079	0.5689 $\pm$ 0.0028	0.3699 $\pm$ 0.0087	0.5173 $\pm$ 0.0018	0.4237 $\pm$ 0.0037	0.4237 $\pm$ 0.0035
SAGEGP	0.5655 $\pm$ 0.0196	0.5191 $\pm$ 0.0029	0.5958 $\pm$ 0.0076	0.6921 $\pm$ 0.0174	0.5720 $\pm$ 0.0187	0.5960 $\pm$ 0.0074
GINGP	0.5782 $\pm$ 0.0218	0.7525 $\pm$ 0.0294	0.6803 $\pm$ 0.0198	0.6910 $\pm$ 0.0540	0.6349 $\pm$ 0.0090	0.5997 $\pm$ 0.0185
GraphGP	0.7069$\pm$0.0055	0.8013$\pm$0.0109	0.7520$\pm$0.0144	0.7909$\pm$0.0382	0.6745$\pm$0.0127	0.6789$\pm$0.0172

Table 4.6: Graph transfer learning on Twitch

Model	Wikipedia graphs		WebKB graphs	
	SQ $\rightarrow$ CH	CH $\rightarrow$ SQ	CO $\rightarrow$ TX	WS $\rightarrow$ TX
RBFGP	-0.0383 $\pm$ 0.0526	-0.0119 $\pm$ 0.0288	0.3089 $\pm$ 0.0533	0.2756 $\pm$ 0.0204
DINO	0.2938 $\pm$ 0.0253	0.1157 $\pm$ 0.0145	0.3536 $\pm$ 0.0442	0.2537 $\pm$ 0.0232
LINKXGP	0.4100 $\pm$ 0.0205	0.0838 $\pm$ 0.0307	0.3836 $\pm$ 0.0530	0.2898 $\pm$ 0.0176
MixHopGP	0.4050 $\pm$ 0.0456	0.3509$\pm$0.0102	0.3264 $\pm$ 0.0569	0.3062 $\pm$ 0.0337
H2GCNGP	0.4165 $\pm$ 0.0559	0.2652 $\pm$ 0.0106	0.3816 $\pm$ 0.0430	0.3036 $\pm$ 0.0442
GraphGP	0.4938$\pm$0.0352	0.3214 $\pm$ 0.0080	0.4146$\pm$0.0402	0.3301$\pm$0.0585

Table 4.7: Graph transfer learning on Wikipedia and WebKB

tively. It is notable that recent work [282] shows the equivalence between graph neural networks (e.g., GraphSAGE [56], GIN [57]) and graph Gaussian processes. Similarly, we can derive the corresponding graph Gaussian processes for LINKX [290], MixHop [291] and H2GCN [59], which are termed as LINKXGP, MixHopGP, and H2GCNGP, respectively.

Model Configuration. We use GPyTorch [292] to build the graph Gaussian process models and optimize the hyperparameters with gradient descent optimizer. For the proposed GraphGP algorithm, we adopt $k = 2$ and $L = 2$ for all the experiments. The hyperparameters are optimized using Adam [293] with a learning rate of 0.01 and a total number of training epochs of 500.

Main Results. Table 4.5 and Table 4.6 provide the transferable node regression results on Airport, Agriculture, and Twitch graphs. Following [282], we report the coefficient of determination R^2 (mean and standard deviation with 5 runs) on the testing target nodes for performance comparison. The experimental results verify the effectiveness of GraphGP over Gaussian process baselines. It is notable that the graphs in Table 4.5 and Table 4.6 follow the homophily assumption. In contrast, Wikipedia and WebKB graphs in Table 4.7 hold the heterophily assumption [59]. It can be seen that under heterophily assumption, GraphGP also archives much better performance in most cases. In addition, we evaluate the knowledge transferability across graphs with different assumptions in Figure 4.3. It is observed that

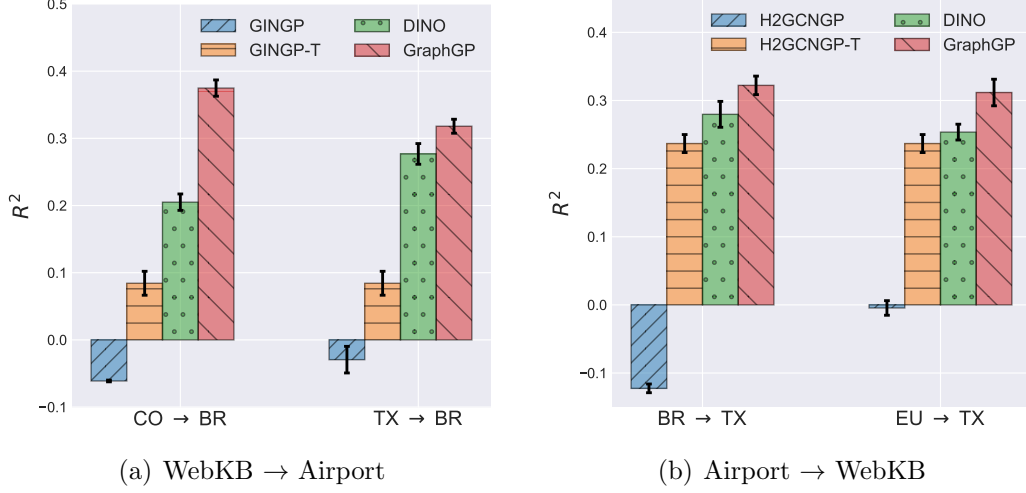


Figure 4.3: Knowledge transfer across graphs with different assumptions

compared to GINGP (or H2GCNGP) over both labeled source and target nodes, GINGP-T (or H2GCNGP-T) learns the Gaussian process over only target nodes, thereby enabling much better transfer performance from WebKB to Airport in Figure 4.3(a) (or from Airport to WebKB in Figure 4.3(b)). Moreover, GraphGP can outperform those target-only graph Gaussian processes in this scenario.

Ablation Study. We investigate the correlation between transfer performance and normalized graph domain similarity $\frac{\kappa_v^{(L)}(G^s, G^t)}{\sqrt{\kappa_v^{(L)}(G^s, G^s) \cdot \kappa_v^{(L)}(G^t, G^t)}}$ in Corollary 4.3. Figure 4.4 visualizes the estimated normalized graph domain similarity of GraphGP and R^2 over the target testing nodes for RU → PT on Twitch, where the graph domain similarity changes when injecting noise to the target graph. To be specific, we gradually add some noise to the target graph such that the graph domain similarity between the source and target graphs can change accordingly. Given the feature vector (node attribute) x_v of each node v , we simply multiply the feature vector by a constant δ_0 , i.e., $\hat{x}_v = \delta_0 \cdot x_v$, where δ_0 denotes the perturbation magnitude. In the experiments, we choose δ_0 from 0.1 to 1 with the interval of 0.1. Figure 4.4 shows that the transfer performance is positively correlated with the normalized graph domain similarity. This is consistent with our observation in Corollary 4.3.

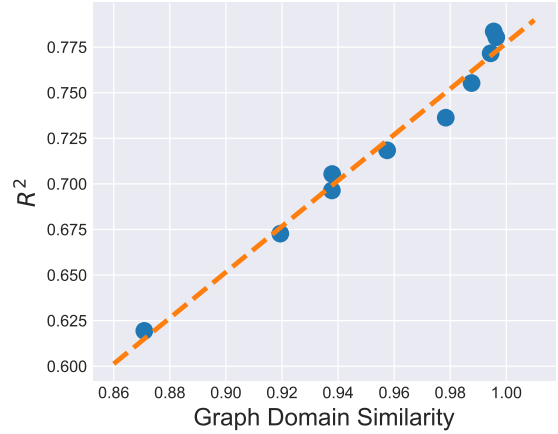


Figure 4.4: Ablation study

CHAPTER 5: ADVERSARIAL ROBUSTNESS OF KNOWLEDGE TRANSFER

In the previous chapters, we study the knowledge transferability in various transfer learning scenarios, by building the connections between the transfer accuracy and the distribution shift across domains. Beyond transfer accuracy, this thesis aims to understand deep transfer learning from the perspective of knowledge trustworthiness. To this end, this chapter studies adversarial robustness in the context of deep transfer learning. Generally speaking, adversarial robustness refers to a machine learning model’s capability to handle carefully crafted adversarial samples. Hence we investigate adversarial attacks and defenses within transfer learning frameworks. Section 5.1 demonstrates that the generated poisoned attacks on the source domain can indirectly manipulate the prediction performance of transfer learning approaches on the target domain [44, 294]. Furthermore, Section 5.2 delves into the adversarial robustness of knowledge transfer through bias-variance decomposition [104].

Notation. Table 5.1 summarizes the main notation used in this chapter. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input feature space and output label space. We denote $\mathcal{D}^s, \mathcal{D}^t$ to be the source and target domains associated with data distributions $\mathbb{P}^s, \mathbb{P}^t$ over $\mathcal{X} \times \mathcal{Y}$, respectively. The source and target marginal distributions over $\mathcal{X}$ are denoted as $\mathbb{P}_X^s$ and $\mathbb{P}_X^t$, respectively. The distributional discrepancy between the source and target domains can be measured by the domain discrepancy $d(\cdot, \cdot)$. The loss function is denoted by $\mathcal{L}(\cdot, \cdot)$. For any prediction function $h(\cdot)$, the expected prediction error is defined as $\mathcal{E}(h) = \mathbb{E}_{(x,y) \sim \mathbb{P}}[\mathcal{L}(h(x), y)]$ for any $h \in \mathcal{H}$, and its empirical estimate is given by $\hat{\mathcal{E}}(h) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(h(x_i), y_i)$ given n samples.

Notation	Definition
$\mathcal{X}, \mathcal{Y}$	Input space, output space
$\mathcal{D}^s, \mathcal{D}^t$	Source and target domains
$\mathbb{P}^s, \mathbb{P}^t$	Source and target joint distributions over $\mathcal{X} \times \mathcal{Y}$
$\mathbb{P}_X^s, \mathbb{P}_X^t$	Source and target marginal distributions over $\mathcal{X}$
$\{x_i^s, y_i^s\}_{i=1}^{n^s}$	n^s labeled source samples
$\{x_j^t\}_{j=1}^{n^t}$	n^t unlabeled target samples
$\mathcal{E}^s, \mathcal{E}^t$	Expected prediction errors of source and target domains
$\hat{x}$	Adversarially perturbed x
$\Omega(\cdot)$	Perturbation constraint
y_*, y_m, y	optimal prediction, main prediction, and ground-truth of sample x
$\mathcal{L}(\cdot, \cdot)$	Loss function
$d(\cdot, \cdot)$	Domain discrepancy

Table 5.1: Notation used in Chapter 5

5.1 ADVERSARIAL VULNERABILITY OF KNOWLEDGE TRANSFER

In this section, we propose a generic indirect invisible poisoning framework to demonstrate the vulnerability of deep transfer learning against adversarial perturbation in the training phase [44].

5.1.1 Problem Definition

Formally, we defined the problem of data poisoning attacks on deep transfer learning [44] as follows.

Definition 5.1. Given a source domain $\mathcal{D}^s$ with labeled samples $\{(x_i^s, y_i^s)\}_{i=1}^{n^s}$ and a target domain $\mathcal{D}^t$ with unlabeled samples $\{x_j^t\}_{j=1}^{n^t}$, **indirect and invisible data poisoning attack** aims to degrade the overall classification performance of a transfer learning algorithm on the target domain, and meanwhile, it satisfies the following three conditions: (i) **imperceptible**: poisoned inputs are perceptibly indistinguishable from real inputs; (ii) **indirect**: only source samples are manipulated; (iii) **invisible**: both source classification error and marginal domain discrepancy will not increase.

We focus on the learning scenarios where the target domain has no labeled data and both domains share the same group of class labels (a.k.a. unsupervised domain adaptation [2]). Specifically, we focus on transfer learning algorithms based on invariant representation learning, with the following unified objective function.

$$\min_{\theta, \phi} \frac{1}{n^s} \sum_{i=1}^{n^s} \mathcal{L}(h(f(x_i^s; \theta); \phi), y_i^s) + d(\mathbb{P}_X^s, \mathbb{P}_X^t; \theta) \quad (5.1)$$

where $f(\cdot; \theta)$ is a feature extractor parameterized by θ and $h_\phi(\cdot)$ is a classifier parameterized by ϕ , and $d(\cdot, \cdot; \theta)$ denotes a generic hypothesis-dependent domain discrepancy in the feature space. Many popular algorithms could be fitted into the objective function in Eq. (5.1), e.g., CORAL [215], DAN [9], DANN [88], MDD [53], f -DAL [31], etc.

Without label information in the target domain, the input-output relationship can only be transferred from the source to the target domains via transfer learning. This enables the adversary to maliciously manipulate the source domain to control the prediction behaviors of transfer learning algorithms in the target domain. This is even more alarming given recent findings showing that modern neural networks are vulnerable to adversarial samples [64, 65], and the public data sets often contain some mislabeled samples [295].

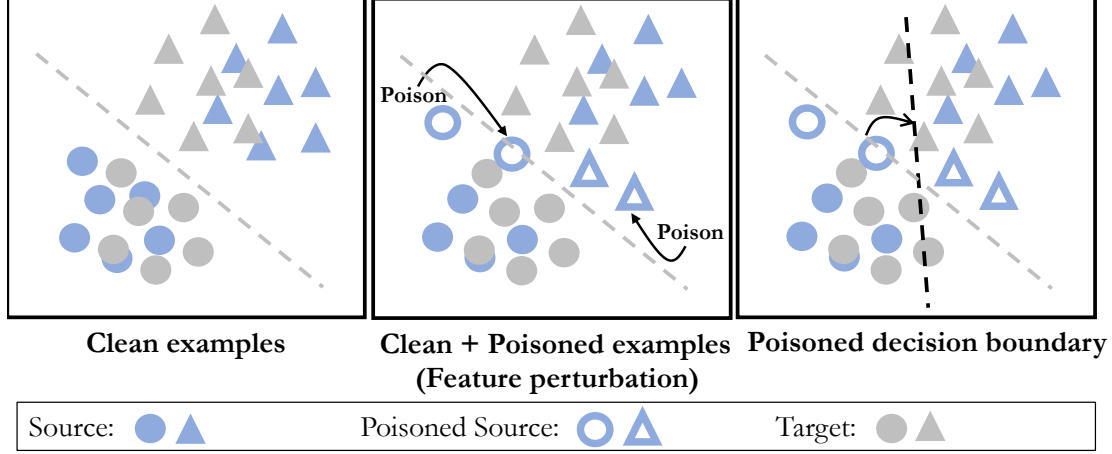


Figure 5.1: A motivating example of data poisoning attacks on transfer learning scenarios where the decision boundary of a learning algorithm becomes much worse after perturbing raw source samples

5.1.2 Motivation

Figure 5.1 provides a motivating scenario to show how transfer learning techniques can be attacked by carefully adding adversarial noise to source data. It focuses on manipulating the source samples only (*indirect attacks*) while preserving the discrimination of source samples and marginal domain discrepancy across domains (*invisible attacks*).

It is observed [103] that minimizing the marginal data distribution and source classification error in transfer learning algorithms could lead to negative transfer with undesirable prediction performance on the target domain. Furthermore, the following theorem shows that for any target domain, there exists a source domain satisfying that it is class-separable and has identical marginal data distribution with the target domain over the input space $\mathcal{X}$, such that the target error of the optimal hypothesis $h \in \mathcal{H}$ is large.

Theorem 5.1. Given any class-separable target domain $\mathbb{P}^t$, there exist at least one source domain $\mathbb{P}^s$ and $h \in \mathcal{H}$ satisfying $\mathcal{E}^s(h) = 0$ and $d(\mathbb{P}_X^s, \mathbb{P}_X^t) = 0$, such that the expected error of the target domain $\mathcal{E}^t(h) = 1$.

Proof. We first consider a binary classification scenario with $\mathcal{Y} = \{-1, 1\}$. Given any target domain $\mathbb{P}^t$, we can construct a source domain $\mathbb{P}^s$ as follows. For any data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we let the data distribution of source domain be either $\mathbb{P}^s(x, y = c) = \mathbb{P}^t(x, y = c)$ for all $c \in \{-1, 1\}$ or $\mathbb{P}^s(x, y = c) = \mathbb{P}^t(x, y = -c)$ for all $c \in \{-1, 1\}$. Then, it is easy to show $\mathbb{P}_X^s = \mathbb{P}_X^t$, so we have $d(\mathbb{P}_X^s, \mathbb{P}_X^t) = 0$ for any domain discrepancy measure. Based on the constructed source domain, we can then define the hypothesis $h \in \mathcal{H}$ as follows. Let $h^* \in \mathcal{H}$ be the optimal predictor of the target domain. If one data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ from the

target domain satisfies $h^*(x) = y$, we hold $\mathbb{P}^s(x, y = c) = \mathbb{P}^t(x, y = -c)$, $\mathbb{P}^s(x, y = c) = \mathbb{P}^t(x, y = c)$ otherwise, for all $c \in \{-1, 1\}$. Then there exists $h = -h^* \in \mathcal{H}$ such that $\mathcal{E}^s(h) = 0$. In this case, the target classification error $\mathcal{E}^t(h) = 1 - \mathcal{E}^t(h^*)$. Furthermore, we can see that $\mathcal{E}^t(h) = 1$ when $\mathcal{E}^t(h^*) = 0$.

Similarly, we can construct the source domain for multi-class learning scenarios, i.e., $\mathcal{Y} = \{0, 1, \dots, C - 1\}$ in total. For any data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we let the data distribution of source domain be either $\mathbb{P}^s(x, y = i) = \mathbb{P}^t(x, y = j)$ where i and j represents the i^{th} and j^{th} classes respectively. Suppose the optimal target hypothesis h^* can partition the space $\mathcal{X} \times \mathcal{Y}$ into C sub-regions where C is the number of classes. In this case, we can set the class label of samples in each sub-region as one of classes such that $\mathbb{P}^s(x, y = i) = \mathbb{P}^t(x, y = j)$ and $\mathbb{P}^s(x, y = j) = \mathbb{P}^t(x, y = i)$ for any $i \neq j$. QED.

5.1.3 A Generic Framework

The overall goal of data poisoning attacks is to inject the adversarial noise into the source data in the training phase such that the overall prediction performance of a base transfer learning algorithm on the target domain could largely deteriorate. Conventional poisoning attacks [296, 297] on a single domain can be applied to degrade the transfer performance by enforcing the source samples to be non-separable in the feature space. But in this case, the training loss (e.g., classification error in the source domain) would significantly increase, and thus such attacks can be easily noticed in the training phase.

To solve this problem, we focus on the indirect and invisible poisoning attacks such that both source classification error and marginal domain discrepancy across domains would not increase significantly during model training. Here it is assumed that the adversary has the full knowledge about the source/target training samples and the learning algorithm (e.g., model architecture, hyper-parameters, etc.). Besides, in order to enforce the adversarial attacks to be perceptibly unnoticeable, it requires producing the poisoned sample $\hat{x}$ with respect to the input x under a reasonable perturbation constraint $\Omega(x)$, i.e., $\hat{x} \in \Omega(x)$. We consider the commonly used constraint $\Omega(x) := \{\hat{x} \mid \|\hat{x} - x\|_\infty \leq \epsilon\}$ for a perturbation magnitude ϵ in image classification tasks [64].

For notation brevity, we let $\mathbf{X}^s = \{x_i^s\}_{i=1}^{n^s}$ be the raw source samples associated with class labels $\mathbf{Y}^s = \{y_i^s\}_{i=1}^{n^s}$, and $\hat{\mathbf{X}}^s = \{\hat{x}_i^s\}_{i=1}^{n^s}$ be the poisoned source samples (associated with unchanged class labels $\mathbf{Y}^s = \{y_i^s\}_{i=1}^{n^s}$). The overall objective function of **I2Attack** could be

mathematically formulated as the following bi-level optimization problem [44].

$$\begin{aligned}
& \max_{\hat{\mathbf{X}}^s} \mathcal{O}(\hat{\mathbf{X}}^s, \mathbf{X}^s, \mathbf{Y}^s; \theta^*, \phi^*) \\
& \text{s.t., } \theta^*, \phi^* = \arg \min_{\theta, \phi} \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta); \phi \right), \mathbf{Y}^s \right) + d \left(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta) \right) \\
& \text{s.t., } d(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1 \\
& \text{s.t., } \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta^*); \phi^* \right), \mathbf{Y}^s \right) \leq \delta_2 \\
& \text{s.t., } \hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s)
\end{aligned} \tag{5.2}$$

where $\mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta); \phi), \mathbf{Y}^s) = \frac{1}{n^s} \sum_{i=1}^{n^s} \mathcal{L}(h(f(\hat{x}_i^s; \theta); \phi), y_i^s)$. $d(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta))$ is the domain discrepancy across domains in the feature space learned by $f(\cdot; \theta)$. $\mathcal{O}(\cdot)$ is the attacking function for learning the poisoned source samples. Here $\delta_1 \geq 0$ constrains the marginal domain discrepancy between the poisoned source domain and the clean source domain, and $\delta_2 \geq 0$ constrains the classification error on the poisoned source domain. In this case, the adversary poisons the source data under the following conditions: (i) the model parameters θ and ϕ are optimal with respect to the poisoned source domain and raw target domain; (ii) the last three constraints guarantee that the poisoning attacks are perceptibly unnoticeable and algorithmically invisible.

If $d(\cdot, \cdot)$ obeys triangle inequality, we have $d(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta)) \leq d(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^s; \theta)) + d(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta)) \leq \delta_1 + d(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta))$. The minimization of $d(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*))$ on the source domain encourages the poisoning attacks to preserve the marginal domain discrepancy between source and target domains. Thus, the constraints $d(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1$ and $\mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta^*); \phi^*), \mathbf{Y}^s) \leq \delta_2$ ensure that the source classification error and domain discrepancy will not be significantly affected, thus leading to the algorithmically invisible adversarial attacks.

Besides, the attacking function $\mathcal{O}(\cdot)$ of **I2Attack** in Eq. (5.2) can be defined as follows.

$$\mathcal{O}(\hat{\mathbf{X}}^s, \mathbf{X}^s, \mathbf{Y}^s; \theta^*, \phi^*) = d \left(\hat{\mathbf{X}}^s \circ \mathbf{Y}^s, \mathbf{X}^s \circ \mathbf{Y}^s; \theta^*, \phi^* \right) \tag{5.3}$$

where $\circ$ is the combination of input feature and output class label over $\mathcal{X} \times \mathcal{Y}$. This aims to maximize the label-informed domain discrepancy [39] over $\mathcal{X} \times \mathcal{Y}$ between the raw source domain and the poisoned source domain.

We show that the proposed framework **I2Attack** has the following advantages in analyzing the adversarial vulnerability of deep transfer learning techniques. (1) **Flexible**: It is flexible to be instantiated for attacking any discrepancy-based transfer learning algorithm, especially when there is no label information in the target domain. (2) **Unnoticeable**:

The properties of indirect and invisible attacks would make it difficult to be noticed in the training phase, thus posing a significant threat to public source domain data in real scenarios. (3) **Interpretable**: The label-informed data distribution over $\mathcal{X} \times \mathcal{Y}$ between source and target domains is implicitly maximized in **I2Attack**, such that the existing algorithms [9, 53, 88, 215] with matched marginal distributions over $\mathcal{X}$ across domains leads to negative transfer [26] on the target domain without affecting the training loss.

5.1.4 Instantiated Algorithms

We instantiate our framework **I2Attack** for attacking existing deep transfer learning algorithms, including DAN [9], DANN [88], and MDD [53].

I2Attack-DAN. Deep Adaptation Network (DAN) [9] learns the domain-invariant feature representation in a reproducing kernel Hilbert space where the mean embeddings of different domain distributions are explicitly matched as follows.

$$\min_{\theta, \phi} \mathcal{L}(h(f(\mathbf{X}^s; \theta); \phi), \mathbf{Y}^s) + d_k(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta)) \quad (5.4)$$

where $d_k(\cdot, \cdot)$ represents the (multi-kernel) maximum mean discrepancy (MMD) [115, 298] between source and target domains in the feature space learned by $f(\cdot; \theta)$.

Following Eq. (5.2), we propose to learn the poisoned source samples with the following bi-level optimization problem.

$$\begin{aligned} \max_{\hat{\mathbf{X}}^s} \quad & d_k(f(\hat{\mathbf{X}}^s; \theta^*) \circ \mathbf{Y}^s, f(\mathbf{X}^s; \theta^*) \circ \mathbf{Y}^s) \\ \text{s.t.,} \quad & \theta^*, \phi^* = \arg \min_{\theta, \phi} \mathcal{L}\left(h\left(f(\hat{\mathbf{X}}^s; \theta); \phi\right), \mathbf{Y}^s\right) + d_k\left(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta)\right) \\ \text{s.t.,} \quad & d_k(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1 \\ \text{s.t.,} \quad & \mathcal{L}\left(h\left(f(\hat{\mathbf{X}}^s; \theta^*); \phi^*\right), \mathbf{Y}^s\right) \leq \delta_2 \\ \text{s.t.,} \quad & \hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s) \end{aligned} \quad (5.5)$$

Below we tackle this bi-level optimization problem using the model-agnostic meta-learning (MAML) [258]. MAML aims to find appropriate hyper-parameter configurations (e.g., model initialization, learning rate schedules, etc.) of neural networks. In this case, we can consider the poisoned source samples $\hat{\mathbf{X}}^s$ as the hyper-parameters of a transfer learning algorithm and then optimize Eq. (5.5) using the meta-gradient of attacking function with respect to $\hat{\mathbf{X}}^s$ as follows.

$$\hat{\mathbf{X}}^s \leftarrow \text{Proj}_{\Omega(\mathbf{X}^s)} \left(\hat{\mathbf{X}}^s - \alpha \nabla_{\hat{\mathbf{X}}^s}^{\text{meta}} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^*, \phi^*) \right) \quad (5.6)$$

where $\text{Proj}_{\Omega(\mathbf{X}^s)}(\cdot)$ projects the updated poisoned input onto $\Omega(\mathbf{X}^s)$ in every iteration, and

$$\begin{aligned} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^*, \phi^*) &= d_k \left(f(\hat{\mathbf{X}}^s; \theta^*) \circ \mathbf{Y}^s, f(\mathbf{X}^s; \theta^*) \circ \mathbf{Y}^s \right) \\ &\quad - \mu \left(d_k(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) + \mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta^*); \phi^*), \mathbf{Y}^s) \right) \end{aligned} \quad (5.7)$$

is the meta attacking function. Here, θ^*, ϕ^* can be learned with vanilla gradient descent as follows.

$$\begin{aligned} \theta^{l+1} &= \theta^l - \beta \nabla_{\theta^l} \left(\mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta^l); \phi^l), \mathbf{Y}^s) + d_k(f(\hat{\mathbf{X}}^s; \theta^l), f(\mathbf{X}^s; \theta^l)) \right) \\ \phi^{l+1} &= \phi^l - \beta \nabla_{\phi^l} \mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta^l); \phi^l), \mathbf{Y}^s) \end{aligned} \quad (5.8)$$

where l is the iteration index. With L updates of gradient descent on θ and ϕ , following the first-order approximation of meta-gradient (FO-MAML) proposed in [258], we have

$$\begin{aligned} &\nabla_{\hat{\mathbf{X}}^s}^{\text{meta}} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^*, \phi^*) \\ &\approx \nabla_{\hat{\mathbf{X}}^s}^{\text{meta}} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^L, \phi^L) \\ &= \nabla_f \mathcal{J}(\hat{\mathbf{X}}^s; \theta^L, \phi^L) [\nabla_{\hat{\mathbf{X}}^s} f(\hat{\mathbf{X}}^s; \theta^L) + \nabla_{\theta^L} f(\hat{\mathbf{X}}^s; \theta^L) \nabla_{\hat{\mathbf{X}}^s} \theta^L] \\ &\quad + \nabla_h \mathcal{J}(\hat{\mathbf{X}}^s; \theta^L, \phi^L) [\nabla_{\hat{\mathbf{X}}^s} h(f(\hat{\mathbf{X}}^s; \theta^L); \phi^L) + \nabla_{\phi^L} h(f(\hat{\mathbf{X}}^s; \theta^L); \phi^L) \nabla_{\hat{\mathbf{X}}^s} \phi^L] \\ &\approx \nabla_f \mathcal{J}(\hat{\mathbf{X}}^s; \theta^L, \phi^L) \nabla_{\hat{\mathbf{X}}^s} f(\hat{\mathbf{X}}^s; \theta^L) + \nabla_h \mathcal{J}(\hat{\mathbf{X}}^s; \theta^L, \phi^L) \nabla_{\hat{\mathbf{X}}^s} h(f(\hat{\mathbf{X}}^s; \theta^L); \phi^L) \end{aligned} \quad (5.9)$$

Then the approximated meta-gradient $\nabla_{\hat{\mathbf{X}}^s}^{\text{meta}} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^*, \phi^*)$ is used to update the poisoned input $\hat{\mathbf{X}}^s$ in Eq. (5.6). The overall training procedures are illustrated in Algorithm 5.1. The algorithm iteratively updates the poisoned source samples and stops when the user-defined stopping criteria are satisfied.

I2Attack-DANN. Inspired by generative adversarial network [125], Domain-Adversarial Neural Network (DANN) [88] learns the domain-invariant latent feature space in an adversarial manner where the $\mathcal{H}$ -divergence across domains and the source classification error could be minimized in the feature space. The overall objective function of DANN is given below.

$$\min_{\theta, \phi} \mathcal{L}(h(f(\mathbf{X}^s; \theta); \phi), \mathbf{Y}^s) + d_{\mathcal{A}}(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta)) \quad (5.10)$$

where $d_{\mathcal{A}}(\cdot, \cdot)$ is the Proxy $\mathcal{A}$ -distance [29] between source and target domains in the feature space learned by $f(\cdot; \theta)$. Mathematically,

$$\begin{aligned} d_{\mathcal{A}}(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta)) &= \min_{\theta} \max_D \mathbb{E}_{x^s \in \mathbf{X}^s} [D(f(x^s; \theta))] \\ &\quad + \mathbb{E}_{x^t \in \mathbf{X}^t} [1 - D(f(x^t; \theta))] \end{aligned} \quad (5.11)$$

Algorithm 5.1: I2Attack

Input: Labeled source samples $(\mathbf{X}^s, \mathbf{Y}^s)$ and unlabeled target samples $\mathbf{X}^t$, a base transfer learning algorithm with discrepancy measure $d(\cdot, \cdot)$, and perturbation magnitude ϵ ;

Output: Poisoned source samples $\hat{\mathbf{X}}^s$.

Initialize $\hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s)$ and base model parameters θ, ϕ ;

while *Stopping criterion is not satisfied* **do**

foreach $l = 1, \dots, L$ **do**

 | Update base model parameters θ, ϕ using Eq. (5.8);

end

 Estimate meta-gradient $\nabla_{\hat{\mathbf{X}}^s}^{\text{meta}} \mathcal{J}(\hat{\mathbf{X}}^s; \theta^*, \phi^*)$ using Eq.(5.9);

 Update poisoned source samples $\hat{\mathbf{X}}^s$ using Eq. (5.6);

end

return Poisoned source samples $\hat{\mathbf{X}}^s$.

where $D(\cdot)$ is a domain discriminator for identifying which domain a sample comes from. Following Eq. (5.2), we propose to generate the poisoning attacks with the following quad-level optimization problem:

$$\begin{aligned} \max_{\hat{\mathbf{X}}^s} \quad & d_{\mathcal{A}}(f(\hat{\mathbf{X}}^s; \theta^*) \circ \mathbf{Y}^s, f(\mathbf{X}^s; \theta^*) \circ \mathbf{Y}^s) \\ \text{s.t.,} \quad & \theta^*, \phi^*, D^* = \arg \min_{\theta, \phi} \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta); \phi \right), \mathbf{Y}^s \right) + d_{\mathcal{A}} \left(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta) \right) \\ \text{s.t.,} \quad & d_{\mathcal{A}}(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1 \\ \text{s.t.,} \quad & \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta^*); \phi^* \right), \mathbf{Y}^s \right) \leq \delta_2 \\ \text{s.t.,} \quad & \hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s) \end{aligned} \tag{5.12}$$

Solving such an optimization problem is challenging due to its high-order combinatorial nature. Instead, we would like to derive an efficient approximation by substituting the domain discrepancy $d_{\mathcal{A}}(\cdot, \cdot)$ of the attacking function with MMD $d_k(\cdot, \cdot)$ as follows.

$$\begin{aligned} \max_{\hat{\mathbf{X}}^s} \quad & d_k(f(\hat{\mathbf{X}}^s; \theta^*) \circ \mathbf{Y}^s, f(\mathbf{X}^s; \theta^*) \circ \mathbf{Y}^s) \\ \text{s.t.,} \quad & \theta^*, \phi^*, D^* = \arg \min_{\theta, \phi} \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta); \phi \right), \mathbf{Y}^s \right) + d_{\mathcal{A}} \left(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta) \right) \\ \text{s.t.,} \quad & d_{\mathcal{A}}(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1 \\ \text{s.t.,} \quad & \mathcal{L} \left(h \left(f(\hat{\mathbf{X}}^s; \theta^*); \phi^* \right), \mathbf{Y}^s \right) \leq \delta_2 \\ \text{s.t.,} \quad & \hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s) \end{aligned} \tag{5.13}$$

In particular, the inner parameters θ^*, ϕ^*, D^* of DANN can be trained using standard back-propagation with gradient reversal layer [88]. On top of this observation, the overall optimization problem of Eq. (5.13) can then be efficiently solved via meta-learning [258] when adopting vanilla gradient descent to update the inner parameters.

I2Attack-MDD. Margin Disparity Discrepancy (MDD) [53] minimizes the empirical source classification error and the disparity discrepancy across domains in the feature space using the following objective function.

$$\min_{\theta, \phi} \mathcal{L}(h(f(\mathbf{X}^s; \theta); \phi), \mathbf{Y}^s) + d_{\text{MDD}}(f(\mathbf{X}^s; \theta), f(\mathbf{X}^t; \theta)) \quad (5.14)$$

where $d_{\text{MDD}}(\cdot, \cdot)$ denotes the margin-aware disparity discrepancy between source and target domains and can be empirically minimized using a minimax adversarial game in the feature space learned by $f(\cdot; \theta)$.

Similar to **I2Attack-DANN**, we design the attacking function with non-adversarial domain discrepancy $d_k(\cdot, \cdot)$ for efficiently generating the poisoning attacks on the MDD algorithm as follows.

$$\begin{aligned} \max_{\hat{\mathbf{X}}^s} & d_k(f(\hat{\mathbf{X}}^s; \theta^*) \circ \mathbf{Y}^s, f(\mathbf{X}^s; \theta^*) \circ \mathbf{Y}^s) \\ \text{s.t.,} & \quad \theta^*, \phi^*, D^* = \arg \min_{\theta, \phi} \mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta); \phi), \mathbf{Y}^s) + d_{\text{MDD}}(f(\hat{\mathbf{X}}^s; \theta), f(\mathbf{X}^t; \theta)) \\ \text{s.t.,} & \quad d_{\mathcal{A}}(f(\hat{\mathbf{X}}^s; \theta^*), f(\mathbf{X}^s; \theta^*)) \leq \delta_1 \\ \text{s.t.,} & \quad \mathcal{L}(h(f(\hat{\mathbf{X}}^s; \theta^*); \phi^*), \mathbf{Y}^s) \leq \delta_2 \\ \text{s.t.,} & \quad \hat{\mathbf{X}}^s \in \Omega(\mathbf{X}^s) \end{aligned} \quad (5.15)$$

It can then be optimized with meta-gradient based updating methods.

5.1.5 Experiment

In the experiments, we focus on answering whether the proposed **I2Attack** framework can maliciously manipulate the prediction performance of existing transfer learning algorithms.

Data Sets. We use the following image classification benchmarks.

- **Digits:** We use three digital image data sets: MNIST (M) [299], USPS (U) [300] and SVHN (S) [301] with 70,000, 99,289 and 9,297 images of 10 categories, respectively.
- **Office-31** [259]: It has 4,652 images of 31 categories from three domains: Amazon (A), Webcam (W), and DSLR (D).

- Office-Home [265]: It has 15,500 images of 65 categories from four domains: Artistic images (Ar), Clip Art (Cl), Product images (Pr), and Real-World images (Rw).
- Image-CLEF [123]: It has 2,400 images of 12 categories from four domains: Caltech-256 (C), ImageNet ILSVRC 2012 (I), Pascal VOC2012 (P), and Bing (B).
- VisDA2017 [302]: It has over 200K images of 12 categories from two domains: Synthetic (Syn) and Real.

Baselines. We compare our proposed **I2Attack** framework with the following baselines: naïve RAttack that adds random Gaussian noise to source samples, SAttack [296] that generates the poisoned source samples by maximizing the classification error; BFGSM [167] that pre-trains a source model and then generates the adversarial source samples via Fast Sign Gradient Method (FGSM) [64]. The generated poisoned source samples are then used to evaluate the adversarial vulnerability of transfer learning algorithms.

Model Configuration. We adopt stochastic gradient descent with a mini-batch of 72 for inner updates with $L = 1$, $\beta = 0.001$, $\alpha = 0.01$, and $\epsilon = 0.1$. The overall iterations are 25 in our experiments. Besides, for DAN [9], DANN [88] and MDD [53], we adopted the ResNet-50 [69] pre-trained on ImageNet for feature extraction with an added 256-dimension bottleneck layer between the *res5c* and *fc* layers. It is then optimized using stochastic gradient descent with a mini-batch of size 32. Following [88], the learning rate η_p is adjusted as: $\eta_p = \frac{\eta_0}{(1+\omega p)^\tau}$, where p is an epoch-dependent scalar linearly varying from 0 to 1, and $\eta_0 = 0.01$, $\omega = 10$, $\tau = 0.75$.

Main Results. Table 5.2 provides the results on evaluating the adversarial vulnerability of transfer learning techniques under **I2Attack**. We report the source classification accuracy (S Acc), domain discrepancy (Disc), and target classification accuracy (T Acc) to demonstrate the impact of adversarial attacks on transfer learning methods. In this case, maximum mean discrepancy, Proxy $\mathcal{A}$ -distance, and margin-aware disparity discrepancy are adopted to measure the distribution discrepancy on DAN [9], DANN [88] and MDD [53], respectively. It is observed that: (1) the target performance is significantly degraded (e.g., up to 90% degradation on Office-31) with the

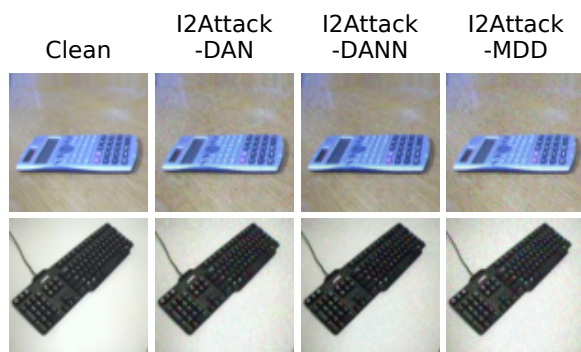


Figure 5.2: Visualization of clean and poisoned source samples on W→D of Office-31

		Digits		Office-31			Office-Home		Image-CLEF			VisDA2017
		M→U	S→M	W→A	W→D	D→A	Ar→Cl	Pr→Rw	B→I	C→P	P→B	Syn→Real
DAN (base model)	S Acc	0.997	0.916	1.000	1.000	1.000	1.000	0.999	1.000	1.000	1.000	0.987
	Disc	0.078	0.085	2.459	2.315	2.156	1.835	1.931	2.137	2.589	1.742	0.478
	T Acc	0.861	0.724	0.654	0.994	0.656	0.498	0.750	0.848	0.750	0.588	0.584
I2Attack-DAN	S Acc	1.000 [−]	1.000 [↑]	0.996 [−]	0.998 [−]	0.994 [−]	0.998 [−]	0.999 [−]	1.000 [−]	1.000 [−]	1.000 [−]	0.986 [−]
	Disc	0.079 [−]	0.079 [↑]	2.304 [↑]	1.975 [↑]	2.152 [−]	1.579 [↑]	1.684 [↑]	1.919 [↑]	1.939 [↑]	1.555 [↑]	0.437 [↑]
	T Acc	0.664 [↓]	0.495 [↓]	0.065 [↓]	0.062 [↓]	0.046 [↓]	0.293 [↓]	0.660 [↓]	0.113 [↓]	0.203 [↓]	0.252 [↓]	0.469 [↓]
DANN (base model)	S Acc	0.997	0.911	1.000	1.000	1.000	1.000	0.999	1.000	1.000	1.000	0.983
	Disc	0.567	0.520	0.646	0.642	0.609	0.506	0.500	0.602	0.758	0.733	0.742
	T Acc	0.896	0.795	0.679	0.998	0.668	0.513	0.756	0.888	0.782	0.597	0.637
I2Attack-DANN	S Acc	1.000 [−]	0.948 [↑]	0.996 [−]	1.000 [−]	0.998 [−]	0.994	0.999 [−]	1.000 [−]	1.000 [−]	0.998 [−]	0.990 [−]
	Disc	0.569 [−]	0.516 [−]	0.588 [↑]	0.643 [−]	0.550 [↑]	0.501 [−]	0.500 [−]	0.572 [↑]	0.695 [↑]	0.593 [↑]	0.688 [↑]
	T Acc	0.801 [↓]	0.510 [↓]	0.078 [↓]	0.046 [↓]	0.105 [↓]	0.378 [↓]	0.673 [↓]	0.083 [↓]	0.233 [↓]	0.142 [↓]	0.201 [↓]
MDD (base model)	S Acc	0.997	0.901	1.000	1.000	1.000	1.000	0.999	1.000	1.000	1.000	0.969
	Disc	1.373	1.496	1.374	1.493	1.028	1.735	1.697	1.501	1.214	1.341	1.123
	T Acc	0.908	0.753	0.693	0.998	0.679	0.505	0.781	0.903	0.788	0.617	0.657
I2Attack-MDD	S Acc	1.000 [−]	0.944	0.996 [−]	0.991 [−]	0.996 [−]	0.993 [−]	0.991 [−]	1.000 [−]	1.000 [−]	1.000 [−]	0.996 [↑]
	Disc	1.317 [↑]	1.453 [↑]	1.056 [↑]	1.473 [↑]	0.938 [↑]	1.603 [↑]	1.645 [↑]	1.449 [−]	1.010 [↑]	1.339 [−]	0.999 [↑]
	T Acc	0.789 [↓]	0.585 [↓]	0.050 [↓]	0.024 [↓]	0.137 [↓]	0.382 [↓]	0.679 [↓]	0.163 [↓]	0.170 [↓]	0.217 [↓]	0.301 [↓]

Table 5.2: Data poisoning attacks on deep transfer learning (‘[−]’: almost unchanged; ‘[↑]’: improved; ‘[↓]’: degraded)

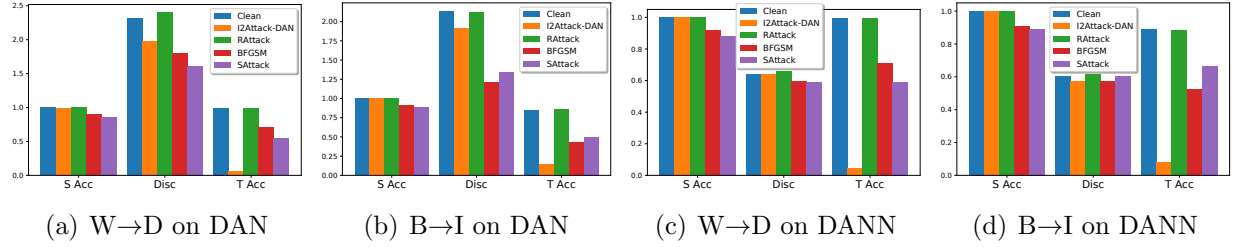


Figure 5.3: Comparison of I2Attack and baselines on W→D of Office-31 and B→I of Image-CLEF

learned poisoned source samples; (2) the source classification accuracy and domain discrepancy could be almost unchanged and even become better (i.e., higher source accuracy or lower marginal domain discrepancy) in some cases.

Figure 5.3 demonstrates the effectiveness of I2Attack on attacking DAN and DANN, compared to baseline methods on Office-31 and Image-CLEF. More specifically, we observe that (1) naïve RAttack with random Gaussian noise would not largely degrade the transfer accuracy; (2) the poisoned source samples generated from SAttack and BFGSM deteriorate both source and target classification performance, and thus they are easily detected in the model training phase; (3) compared to baselines, the proposed I2Attack framework achieves lower prediction accuracy on the target domain by maximizing the discrepancy of joint data

	DAN			DANN		
	S Acc	Disc	T Acc	S Acc	Disc	T Acc
Clean	1.000	2.315	0.994	1.000	0.642	0.998
I2Attack-DAN	0.998	1.975	0.062	0.996	0.622	0.020
I2Attack-DANN	0.999	2.031	0.068	1.000	0.643	0.046
I2Attack-MDD	0.991	2.156	0.092	0.994	0.649	0.032

Table 5.3: Transferability of **I2Attack** on Office-31 (W→D)

	Clean			I2Attack		
	S Acc	Disc	T Acc	S Acc	Disc	T Acc
B → I	1.000	2.137	0.848	1.000	1.919	0.113
B → C	1.000	2.215	0.907	1.000	1.921	0.120
B → P	1.000	1.927	0.717	1.000	1.755	0.098

Table 5.4: Universality of **I2Attack** on Image-CLEF

distribution over $\mathcal{X} \times \mathcal{Y}$ across domains.

Ablation Study. In the proposed **I2Attack** framework, we assume that the adversary has full knowledge regarding the base models and target data. This enables the white-box poisoning attacks for knowledge transfer. The key idea is to estimate the gradient of the base model with respect to the source data, and then inject the noise for manipulating the source distribution. However, in black-box poisoning attack scenarios, the base model and the target domain are not accessible to the adversary. This challenges the application of gradient-based white-box attacks. Inspired by [303], we can generalize the **I2Attack** framework to black-box attack scenarios by using the substitute base transfer learning model and target domain. To this end, we conduct additional experiments to validate the effectiveness of **I2Attack** for black-box attacks. Table 5.3 shows the results on DAN and DANN using the generated source samples from different **I2Attack** algorithms. It shows that the generated source samples by one attacking algorithm (e.g., **I2Attack-MDD**) can be used to successfully attack any other algorithm (e.g., DAN and DANN). This indicates that **I2Attack** allows attacking the black-box transfer learning model without knowledge of model configuration. Table 5.4 shows the attack results of poisoned source samples on different target domains of Image-CLEF where all poisoned samples are generated from DAN on B→I and then applied to attack other target domains, i.e., C or P. It can be seen that the generated poisoned samples on B can be directly used to attack multiple downstream target domains. This enables the black-box attacks without access to the target domain.

5.2 ADVERSARIALLY ROBUST KNOWLEDGE TRANSFER

In this section, we study the adversarial robustness in the context of federated learning. Federated learning empowers a group of clients to collaborate in training machine learning models under the coordination of a central server [105]. In this case, each client is associated with some training samples drawn from a specific domain. As demonstrated in Section 5.1, adversarial attacks in one domain could result in degraded performance in another related domain. This thus motivates us to rethink the adversarial robustness of the machine learning model learned by different clients [104].

5.2.1 Problem Definition

The goal of standard federated learning [105] is to learn the prediction function by aggregating the knowledge of user data from a set of local clients without leaking user privacy. As illustrated in Figure 5.4, a typical framework [105, 106] of federated learning is summarized as follows.

- (a) *Client Update*: Each client updates local parameters θ_k by minimizing the empirical prediction loss w.r.t. its own private training samples;
- (b) *Forward Communication*: Each client uploads its parameters or parameter updates to the central server;
- (c) *Server Update*: It synchronously aggregates the received parameters;
- (d) *Backward Communication*: The aggregated global parameters are sent back to local clients.

It is shown [105] that the neural network in federated learning actually approximates the updating behavior of a centralized model. For example, the update behavior of FedAvg [105] is closely equivalent to the stochastic gradient descent (SGD) on the centralized data when each client only takes one step of gradient descent. That explains why it might potentially inherit the adversarial vulnerability of centralized neural network models [64, 65]. Therefore, we explore the adversarial robustness of neural networks in the federated learning setting. Formally, we define the problem of adversarially robust federated learning [104] as follows.

Problem 5.1. Given a set of clients/domains $\{\mathcal{D}_k\}_{k=1}^K$ each with labeled training samples $\{x_{ik}, y_{ik}\}_{i=1}^{n_k}$, a learning algorithm $f(\cdot)$ and loss function $\mathcal{L}(\cdot, \cdot)$, *adversarially robust federated learning* aims to output a trained model on the central server that is robust against adversarial perturbations on the test set.

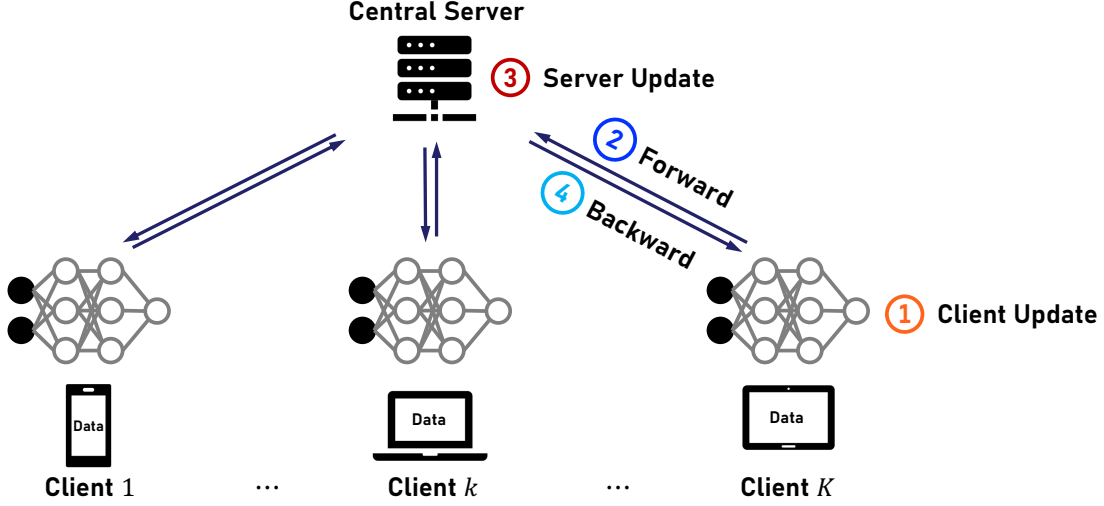


Figure 5.4: Illustration of a standard federated learning framework [105] with four major steps: client updates, server updates, and forward and backward communications between clients and the central server.

It has been revealed [304, 305] that the adversarial robustness of federated learning can be achieved by directly applying adversarial training techniques [306] to clients' local training. Nevertheless, it is unclear how the local adversarial training on clients affects the adversarial robustness of the aggregated global model on the central server. To answer this question, we study the intrinsic relationship between the set of local clients' models and the aggregated server's model from the perspective of bias-variance decomposition [307]. We observe that the generalization error of the server's model is induced by both bias and variance of local clients' models. Therefore, instead of separately focusing on the adversarial robustness of individual clients [304], we propose to analyze the adversarially robust federated learning in a unified framework. The crucial idea is to generate global adversarial samples shared with clients, by leveraging a tiny auxiliary training set $D_{\text{share}} = \{(x_j, y_j)\}_{j=1}^{n_{\text{share}}}$ with n_{share} ($n_{\text{share}} \ll \sum_{k=1}^K n_k$) samples on the central server. This will not break the privacy constraints, as the local data of every client is not shared.

We would like to point out that Problem 5.1 has the following properties. (1) *Asymmetrical communication*: The asymmetrical communication between each client and server cloud is allowed: the server provides both global model parameters and limited shared data to the clients; while each client uploads its local model parameters back to the server. This implies that compared to standard federated learning [105], the communication cost might increase due to those shared data, but it enables the improvement of adversarial robustness in federated learning. (2) *Data distribution*: All training samples on the clients and the central server are assumed to follow identical data distribution. However, the experiments show

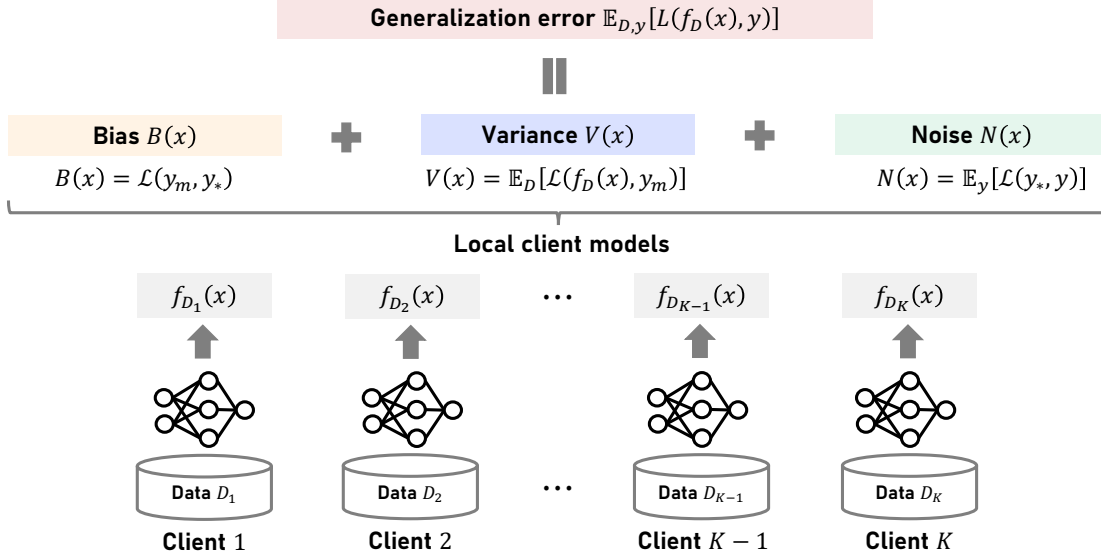


Figure 5.5: Illustration of the proposed Fed_BVA framework in generating poisoned data

that the proposed algorithm also achieves promising performance under the non-IID setting, which is commonly seen among personalized clients in real-world applications. (3) *Shared learning algorithm*: All the clients use the identical model $f(\cdot)$, including architectures, and hyper-parameters (e.g., learning rate, local epochs, local batch size, etc.).

5.2.2 Proposed Framework

We present the adversarially robust federated learning framework [104]. The proposed framework follows the same paradigm of traditional federated learning (see Figure 5.4) but with substantial modifications on the server update and client update.

Server Update with Data Poisoning. The server update has two crucial components. One is model aggregation, which synchronously compresses and aggregates the received local model parameters. The other component is to produce adversarially perturbed samples that are induced by a poisoning attack algorithm for the usage of adversarial training.

The overall goal of federated learning is to learn a prediction function using knowledge from all clients. When the central server receives the parameter updates from local clients, it aggregates the locally updated parameters to obtain a shared global model. Notably, most existing federated learning approaches [105, 178, 179, 308] focus on developing advanced aggregation schemes. One of the popular aggregation methods is FedAvg [105], which aims to average the element-wise parameters of local models, i.e., $\theta_G = \text{Aggregate}(\theta_1, \dots, \theta_K) = \sum_{k=1}^K \frac{n_k}{n} \theta_k$ where θ_k is the model parameters in the k^{th} client and $n = \sum_{k=1}^K n_k$. In this

paper, we focus on improving the adversarial robustness of federated learning by encouraging the local model parameters to be updated with adversarial samples. Then the adversarial robustness of local models can be naturally propagated into the server’s global model after model aggregation. Therefore, the proposed framework is flexible to be incorporated with existing model aggregation methods, such as FedAvg [105] and AFL [178].

It is shown [64] that the adversarial robustness of deep neural networks can be improved by updating the model parameters over adversarial samples. That is because the generalization error of neural networks on adversarial samples can be minimized during model training. However, it will encounter a few issues if we apply adversarial training on federated learning directly. Specifically, one intuitive solution for generating adversarial examples is to separately maximize the generalization error in each local client. Its drawbacks are twofold. First, it will significantly increase the computational burden and memory usage on local clients. Second, the locally generated adversarial samples make the augmented data distributions of local clients more biased [309], which challenges the standard server-level aggregation mechanisms [178].

Instead, we delve into the adversarial robustness of federated learning from the perspective of bias-variance decomposition. Below we provide basic concepts of bias-variance decomposition. Following [307, 310], we start with defining the optimal prediction, main prediction as well as the bias, variance, and noise for any real-valued loss function $\mathcal{L}(\cdot, \cdot)$.

Definition 5.2. Given a loss function $\mathcal{L}(\cdot, \cdot)$ and a learning algorithm $f(\cdot)$, optimal prediction y_* and main prediction y_m for input x are defined as:

$$y_*(x) = \arg \min_{\bar{y}} \mathbb{E}_y[L(\bar{y}, y)] \quad y_m(x) = \arg \min_{\bar{y}'} \mathbb{E}_D[L(f_D(x), \bar{y}')] \quad (5.16)$$

where the label⁷ y and data set D are viewed as the random variables to denote the class label and training set, and f_D denotes the model trained on D .

Definition 5.3. Given a loss function $\mathcal{L}(\cdot, \cdot)$ and a learning algorithm $f(\cdot)$, the bias, variance, and noise can be defined as follows.

$$B(x) = \mathcal{L}(y_m, y_*), \quad V(x) = \mathbb{E}_D[\mathcal{L}(f_D(x), y_m)], \quad N(x) = \mathbb{E}_y[\mathcal{L}(y_*, y)] \quad (5.17)$$

Furthermore, there exists $\lambda, \lambda_0 \in \mathbb{R}$ such that the expected prediction loss $\mathbb{E}_{D,y}[\mathcal{L}(f_D(x), y)]$

⁷Generally, the ground-truth output y is a non-deterministic function [307, 310] of x when the irreducible noise is considered. This means that if x is drawn repeatedly, different values of y could be observed.

for an example x can be decomposed into bias, variance and noise as follows:

$$\mathbb{E}_{D,y}[\mathcal{L}(f_D(x), y)] = B(x) + \lambda V(x) + \lambda_0 N(x) \quad (5.18)$$

In short, bias is the loss incurred by the main prediction w.r.t. the optimal prediction, and variance is the average loss incurred by all individual predictions w.r.t. the main prediction. Noise is conventionally assumed to be irreducible and independent of $f(\cdot)$. Our definitions of optimal prediction, main prediction, bias, variance, and noise slightly differ from previous ones [307, 310]. For example, the conventional optimal prediction was defined as $y_*(x) = \arg \min_{\bar{y}} \mathbb{E}_y[\mathcal{L}(y, \bar{y})]$, and it is equivalent to our definition when the loss function is symmetric over its arguments, i.e., $\mathcal{L}(y_1, y_2) = \mathcal{L}(y_2, y_1)$.

The following theorem shows that in the classification setting, the generalization error of a learning algorithm can be decomposed into bias, variance, and irreducible noise. Note that this decomposition holds for any real-valued loss function in the binary setting [310] with a bias & variance trade-off coefficient that has a closed-form expression.

Theorem 5.2. In binary case, the decomposition in Eq. (5.18) is valid for any real-valued loss function that satisfy $\forall_y \mathcal{L}(y, y) = 0$ and $\forall_{y_1 \neq y_2} \mathcal{L}(y_1, y_2) \neq 0$ with $\lambda = 1$ if $y_m = y_*$ or $\lambda = -\frac{\mathcal{L}(y_m, y_*)}{\mathcal{L}(y_*, y_m)}$ otherwise.

The proof of Theorem 5.2 is similar to [310]. Note that noise is irreducible and we usually drop this term in real applications [307, 310, 311, 312]. Commonly used loss functions for this decomposition are square loss, zero-one loss, and cross-entropy loss with one-hot labels. For the multi-class setting, we treat the trade-off coefficient λ as a hyper-parameter to tune because no closed-form expression of λ is available in a general multi-class learning scenario.

Following [307, 310], we assume a noise-free scenario where the class label y is a deterministic function of x (i.e., if x is sampled repeatedly, the same values of its class y will be observed). The bias and variance can be empirically estimated over the clients' models. This motivates us to generate adversarial samples by attacking the bias and variance induced by clients' models as:

$$\max_{\hat{x} \in \Omega(x)} B(\hat{x}; \theta_1, \dots, \theta_K) + \lambda V(\hat{x}; \theta_1, \dots, \theta_K) \quad \forall (x, y) \in D_{\text{share}} \quad (5.19)$$

where $B(\hat{x}; \theta_1, \dots, \theta_K)$ and $V(\hat{x}; \theta_1, \dots, \theta_K)$ could be empirically estimated from a finite number of clients' parameters $\theta_1, \dots, \theta_K$ trained on local training sets $\{D_1, \dots, D_K\}$. Here λ is a hyper-parameter to measure the trade-off of bias and variance, and $\Omega(x)$ is the perturbation constraint. $\hat{x}$ is the perturbed sample w.r.t. the clean sample x associated with class

label y . Specifically, the perturbation constraint $\hat{x} \in \Omega(x)$ forces the adversarial sample $\hat{x}$ to be visually indistinguishable w.r.t. x . In our paper, we use l_∞ -bounded adversaries [64], i.e., $\Omega(x) := \{\hat{x} \mid \|\hat{x} - x\|_\infty \leq \epsilon\}$ for a perturbation magnitude ϵ .

Here D_{share} (on the server) is the candidate subset of all available training samples that would lead to their perturbed counterparts. This is a more feasible setting as compared to generating adversarial samples on clients' devices because the server usually has powerful computational capacity in real scenarios that allows the usage of flexible poisoning attack algorithms. In this case, both poisoned samples and server model parameters would be sent back to each client (*Backward Communication*), while only clients' local parameters would be uploaded to the server (*Forward Communication*), i.e., *asymmetrical communication*.

Robust Client Update. The robust training of one client's prediction model (i.e., θ_k) can be formulated as the following minimization problem.

$$\min_{\theta_k} \left(\sum_{i=1}^{n_k} \mathcal{L}(f_{D_k}(x_{ik}; \theta_k), y_{ik}) + \sum_{j=1}^{\text{share}} \mathcal{L}(f_{D_k}(\hat{x}_j; \theta_k), y_j) \right) \quad (5.20)$$

where $\hat{x}_j \in \Omega(x_j)$ is the perturbed sample asymmetrically transmitted from the server.

Intuitively, the bias measures the systematic loss of a learning algorithm, and the variance measures the prediction consistency of the learner over different training sets. Therefore, the proposed framework has the following advantages. (i) It encourages the clients to consistently produce the optimal prediction for perturbed samples, thereby leading to a better generalization performance. (ii) Local adversarial training on perturbed samples allows for learning a robust local model, and thus a robust global model could be aggregated from clients.

5.2.3 Instantiated Algorithm

we present the instantiated algorithm **Fed_BVA** with bias-variance attacks. We start by studying the maximization problem in Eq. (5.19) using bias-variance based adversarial attacks. It finds the adversarial sample $\hat{x}$ (from the original sample x) that produces large bias and variance values w.r.t. clients' local models. To this end, we propose two gradient-based algorithms to generate adversarial samples.

Bias-variance based Fast Gradient Sign Method (BV-FGSM): Following FGSM [64], we design the bias-variance attacks by linearizing the maximization problem in Eq. (5.19) with one-step attack.

$$\hat{x}_{\text{BV-FGSM}} = x + \epsilon \cdot \text{sign}(\nabla_x (B(x; \theta_1, \dots, \theta_K) + \lambda V(x; \theta_1, \dots, \theta_K))) \quad (5.21)$$

where $\text{sign}(\cdot)$ denotes the sign function.

Bias-variance based Projected Gradient Descent (BV-PGD): PGD [306] can be considered as a multi-step variant of FGSM and might generate powerful adversarial examples. This motivated us to derive a bias-variance based PGD attack:

$$\hat{x}_{\text{BV-PGD}}^{l+1} = \text{Proj}_{\Omega(x)}(\hat{x}^l + \epsilon \text{sign}(\nabla_{\hat{x}^l}(B(\hat{x}^l; \theta_1, \dots, \theta_K) + \lambda V(\hat{x}^l; \theta_1, \dots, \theta_K)))) \quad (5.22)$$

where $\hat{x}^l$ is the adversarial sample at the l^{th} step with the initialization $\hat{x}^0 = x$ and $\text{Proj}_{\Omega(x)}(\cdot)$ projects each step onto $\Omega(x)$.

The proposed framework could be generalized to any gradient-based adversarial attack algorithms where the gradients of bias $B(\cdot)$ and variance $V(\cdot)$ w.r.t. x are tractable when estimated from finite training sets. Compared with the existing attack methods [64, 164], our loss function the adversary aims to optimize is a linear combination of bias and variance, whereas existing work mainly focused on attacking the overall classification error involving bias only.

The following theorem states that bias $B(\cdot)$ and variance $V(\cdot)$ as well as their gradients over input x could be estimated using the clients' models.

Theorem 5.3. Assume that $\mathcal{L}(\cdot, \cdot)$ is the cross-entropy loss function, then, the empirical estimated main prediction y_m for a sample (x, y) has the following closed-form expression: $y_m(x) = \frac{1}{K} \sum_{k=1}^K f_{D_k}(x; \theta_k)$. Furthermore, the empirical bias and variance over an input x are estimated as $B_{CE}(x) = \frac{1}{K} \sum_{k=1}^K \mathcal{L}(f_{D_k}(x; \theta_k), y)$ and $V_{CE}(x) = H(y_m)$ and their corresponding gradients over x are:

$$\begin{aligned} \nabla_x B_{CE}(x) &= \frac{1}{K} \sum_{k=1}^K \nabla_x L(f_{D_k}(x; \theta_k), y) \\ \nabla_x V_{CE}(x) &= -\frac{1}{K} \sum_{k=1}^K \sum_{j=1}^C (\log y_m^{(j)} + 1) \cdot \nabla_x f_{D_k}^{(j)}(x; \theta_k) \end{aligned} \quad (5.23)$$

where $H(y_m) = -\sum_{j=1}^C y_m^{(j)} \log y_m^{(j)}$ is the entropy of the main prediction y_m and C is the number of classes.

Proof. We first calculate the main prediction. Let

Algorithm 5.2: Fed_BVA

Input: K (number of clients, with local data sets $\{D_k\}_{k=1}^K$); f (learning model); $\mathcal{L}(\cdot, \cdot)$ (loss function); E (number of local epochs); F (fraction of clients selected on each round); B (batch size of local client); η (learning rate); D_{share} (shared data set on server); ϵ (perturbation magnitude);

Output: Adversarially robust model parameters θ_G .

Initialization: Initialize θ_G^0 and $\hat{D}_{\text{share}} = D_{\text{share}}$;

foreach *each round* $r = 1, 2, \dots$ **do**

$m = \max(F \cdot K, 1)$;

$S_r \leftarrow$ randomly sampled m clients;

foreach *each client* $k \in S_r$ **in parallel do**

Initialize k^{th} client's model with θ_G^{r-1} ;

$B \leftarrow$ split $D_k \cup \hat{D}_s$ into batches of size B ;

foreach *each local epoch* $i = 1, 2, \dots, E$ **do**

foreach *local batch* $(x, y) \in B$ **do**

$w_k^r \leftarrow w_k^r - \eta \nabla_w \mathcal{L}(f_{D_k}(x; w_k^r), y)$;

end

end

Calculate $f_{D_k}(x; w_k^r)$, $\nabla_x f_{D_k}(x; w_k^r)$ for $\forall x \in D_{\text{share}}$;

end

foreach $(x, y) \in D_{\text{share}}$ **do**

Estimate the gradients $\nabla_x B(x)$ and $\nabla_x V(x)$;

Update $\hat{x} \in \hat{D}_{\text{share}}$ using Eq. (5.21) or (5.22);

end

$\theta_G^r \leftarrow \text{Aggregate}(\theta_k^r | k \in S_r)$

end

Return θ_G .

$$\begin{aligned}
\mathcal{M} &= \frac{1}{K} \sum_{k=1}^K \mathcal{L}(f_{D_k}(x; \theta_k), \bar{y}') \\
&= -\frac{1}{K} \sum_{k=1}^K (f_{D_k}(x; \theta_k) \cdot \log \bar{y}' + (1 - f_{D_k}(x; \theta_k)) \cdot \log(1 - \bar{y}'))
\end{aligned} \tag{5.24}$$

Take the derivative of $\mathcal{M}$ with respect to $\bar{y}'$, we have

$$\frac{\partial \mathcal{M}}{\partial \bar{y}'} = -\frac{1}{K} \sum_{k=1}^K \left(\frac{f_{D_k}(x; \theta_k)}{\bar{y}'} - \frac{1 - f_{D_k}(x; \theta_k)}{1 - \bar{y}'} \right) = -\frac{1}{K} \sum_{k=1}^K \frac{f_{D_k}(x; \theta_k) - \bar{y}'}{\bar{y}'(1 - \bar{y}')} \tag{5.25}$$

By letting $\partial \mathcal{M} / \partial \bar{y}' = 0$, we have $y_m(x; \theta_1, \dots, \theta_K) = \frac{1}{K} \sum_{k=1}^K f_{D_k}(x; \theta_k)$. Then for bias and

variance, from their definitions, we have,

$$\begin{aligned}
B(x) &= \mathcal{L}\left(\frac{1}{K} \sum_{k=1}^K f_{D_k}(x; \theta_k), y\right) \\
&= -\frac{1}{K} \sum_{k=1}^K f_{D_k}(x; \theta_k) \cdot \log y + \left(1 - \frac{1}{K} \sum_{k=1}^K f_{D_k}(x; \theta_k)\right) \cdot \log(1 - y) \\
&= -\frac{1}{K} \sum_{k=1}^K \left(f_{D_k}(x; \theta_k) \cdot \log y + (1 - f_{D_k}(x; \theta_k)) \cdot \log(1 - y)\right) \\
&= \frac{1}{K} \sum_{k=1}^K \mathcal{L}(f_{D_k}(x; \theta_k), y)
\end{aligned} \tag{5.26}$$

and

$$\begin{aligned}
V(x) &= \frac{1}{K} \sum_{k=1}^K \mathcal{L}(f_{D_k}(x; \theta_k), y_m) \\
&= -\frac{1}{K} \sum_{k=1}^K \left(f_{D_k}(x; \theta_k) \log y_m + (1 - f_{D_k}(x; \theta_k)) \log(1 - \log y_m)\right) \\
&= -y_m \log y_m - (1 - y_m) \log(1 - \log y_m) \\
&= H(y_m)
\end{aligned} \tag{5.27}$$

which completes the proof.

QED.

We present a novel robust federated learning algorithm with our proposed bias-variance attacks, named **Fed_BVA**. Following the framework defined in Eq. (5.19) and Eq. (5.20), key components of the proposed **Fed_BVA** algorithm include (1) bias-variance attacks for generating adversarial samples on the server, and (2) adversarial training using poisoned server samples together with clean local samples on each client. Therefore, we optimize these two objectives iteratively by producing the adversarial samples and updating the local model parameters.

The proposed algorithm is summarized in Algorithm 5.2. Given the server's D_{share} and clients' training data $\{D_k\}_{k=1}^K$ as input, the output is a robust global model on the server. In this case, the clean server data D_{share} will be shared with all the clients. First, it initializes the server's model parameter θ_G and perturbed data $\hat{D}_{\text{share}}$. It then assigns θ_G and $\hat{D}_{\text{share}}$ to the randomly selected clients. Next, each client optimizes its own local model with the received global parameters θ_G as well as its own clean data D_k , and uploads the updated parameters as well as the gradients of the local model on each shared server example back to the server. At last, the server generates the perturbed data $\hat{D}_{\text{share}}$ using the proposed

bias-variance attacks with aggregations (model parameters average, bias gradients average, and variance gradients average) in a similar manner as FedAvg [105]. These aggregations can be privacy secured if additive homomorphic encryption [313] is applied.

5.2.4 Experiment

In the experiments, we validate the adversarial robustness of the proposed **Fed_BVA** algorithm in several federated learning benchmarks.

Data Sets. We evaluate our proposed **Fed_BVA** algorithm on four data sets: MNIST [299], Fashion-MNIST [314], CIFAR-10 [315], and CIFAR-100 [315]. Following [105], we consider two methods to partition the data over clients: IID and non-IID. For the IID setting, the data is shuffled and uniformly partitioned into each client. For the non-IID setting, data is divided into $2F \cdot K$ shards based on sorted labels and then assigned to each client with 2 shards. Thereby, each client will have data with at most two classes. For MNIST and Fashion-MNIST, we deployed 100 clients (each has 1% of the overall data) and 10% of these clients would be selected for model updating (with 50 local training epochs) and aggregation on the server. In the client’s training stage, there will be $n_{\text{share}} = 64$ shared samples being transmitted from the server using asymmetrical communication. Compared to the total number of samples in the data set (i.e., 60k examples), the number of the shared samples, n_{share} , would be only 0.1% of the data set’s size.

For the CIFAR-10 and CIFAR-100 data sets (we use its 20 class version with coarse labels), we deployed 20 clients (each has 5% of the overall data) since the samples have more variations in terms of their categories and cluttered backgrounds. 20% of these clients would be selected for model updating (with 5 local epochs since we utilized deeper models with longer training time per epoch) at every round. Similarly, we only transmitted 30 or 60 samples from the server to the clients in CIFAR-10 or CIFAR-100. Due to the complexity of CIFAR’s data distributions, we enforce n_{share} to be spread out in every category (i.e., both CIFAR-10 and CIFAR-100 with coarse labels will have 3 shared samples among these clients per category). An alternative method is to select n_{share} randomly from the data sets. In CIFAR-10, we choose n_{share} to be only 0.05% of the data set’s size.

Baselines. The baseline models include: (1). **Centralized**: the training with one centralized model, which is identical to the federated learning case that only has one client ($K = 1$) with a fraction ($F = 1$). (2). **FedAvg**: the classical federated averaging model [105]. (3). **FedAvg_AT**: The simplified version of our proposed method where the local clients perform adversarial training with the asymmetrical transmitted perturbed data generated on top of FedAvg’s aggregation. (4) - (6). **Fed_Bias**, **Fed_Variance**, **Fed_BVA**: Our proposed meth-

Method	IID			non-IID		
	Clean	FGSM	PGD-20	Clean	FGSM	PGD-20
Centralized	0.991 ± 0.000	0.689 ± 0.000	0.182 ± 0.000	n/a	n/a	n/a
FedAvg	0.989 ± 0.001	0.669 ± 0.009	0.267 ± 0.014	0.980 ± 0.002	0.491 ± 0.067	0.158 ± 0.074
FedAvg_AT	0.988 ± 0.000	0.802 ± 0.001	0.512 ± 0.042	0.974 ± 0.005	0.649 ± 0.066	0.363 ± 0.066
Fed_Bias	0.986 ± 0.000	0.812 ± 0.009	0.583 ± 0.036	0.971 ± 0.004	0.679 ± 0.040	0.394 ± 0.103
Fed_Variance	0.985 ± 0.001	0.803 ± 0.007	0.572 ± 0.019	0.973 ± 0.005	0.684 ± 0.004	0.395 ± 0.049
Fed_BVA	0.986 ± 0.001	0.818 ± 0.003	0.613 ± 0.020	0.969 ± 0.002	0.705 ± 0.009	0.469 ± 0.031
EAT	0.981 ± 0.000	0.902 ± 0.001	0.811 ± 0.004	0.972 ± 0.002	0.789 ± 0.016	0.415 ± 0.035
EAT+Fed_BVA	0.980 ± 0.001	0.901 ± 0.006	0.821 ± 0.013	0.965 ± 0.005	0.811 ± 0.020	0.670 ± 0.014

Table 5.5: Accuracy of MNIST under white-box attacks in IID and non-IID settings

ods where the asymmetrical transmitted perturbed data is generated using the gradients of bias-only attack, variance-only attack, and bias-variance attack, respectively. (7). **EAT**: Ensemble adversarial training [316], where each client performs local adversarial training, and their model updates are aggregated on the server using **FedAvg**. (8). **EAT+Fed_BVA**: A combination of baselines (6) and (7). Note that the baselines (7) and (8) have high computational requirements on client devices, and are usually not preferred in real scenarios. For fair comparisons, all baselines are modified to the asymmetrical communication setting (**FedAvg** and **EAT** have clean D_{share} received), and all their initializations are set to be the same.

Model Configuration. For the proposed **Fed_BVA** algorithm, we use a 4-layer CNN model for MNIST and Fashion-MNIST, and VGG9 architecture for CIFAR-10 and CIFAR-100. The training is performed using the SGD optimizer with a fixed learning rate of 0.01 and momentum of value 0.9. The trade-off coefficient between bias and variance is set to $\lambda = 0.01$ for all the experiments. Following [316], the maximum perturbations allowed are $\epsilon = 0.3$ on MNIST and Fashion-MNIST, and $\epsilon = \frac{16}{255}$ on CIFAR-10 and CIFAR-100 for both threat and defense models.

Main Results. For the comprehensive experiments in Table 5.5 and Table 5.6, it is easy to verify that our proposed model outperforms all other baselines regardless of the source of the perturbed examples (i.e., locally generated like **EAT+Fed_BVA** or asymmetrically transmitted from the server like **Fed_BVA**). In this case, BV-FGSM (see Eq. (5.21)) is used to generate the adversarial examples $\hat{D}_s$ during model training for robust federated learning. Compared to standard robust federated learning **FedAvg_AT**, the performance of **Fed_BVA** against adversarial attacks still increases 4% – 13% and 2% – 9% on IID and non-IID settings respectively, although **Fed_BVA** is theoretically suitable for the cases that clients have

Method	IID			non-IID		
	Clean	FGSM	PGD-20	Clean	FGSM	PGD-20
Centralized	0.882 ± 0.000	0.229 ± 0.000	0.009 ± 0.000	n/a	n/a	n/a
FedAvg	0.877 ± 0.001	0.300 ± 0.021	0.036 ± 0.016	0.804 ± 0.013	0.193 ± 0.036	0.017 ± 0.003
FedAvg_AT	0.866 ± 0.001	0.490 ± 0.021	0.139 ± 0.011	0.730 ± 0.023	0.445 ± 0.065	0.087 ± 0.042
Fed_Bias	0.862 ± 0.001	0.505 ± 0.015	0.159 ± 0.003	0.709 ± 0.025	0.460 ± 0.038	0.115 ± 0.054
Fed_Variance	0.862 ± 0.002	0.496 ± 0.012	0.157 ± 0.017	0.719 ± 0.036	0.499 ± 0.081	0.120 ± 0.038
Fed_BVA	0.862 ± 0.003	0.528 ± 0.016	0.180 ± 0.027	0.710 ± 0.045	0.495 ± 0.030	0.093 ± 0.028
EAT	0.860 ± 0.005	0.773 ± 0.029	0.103 ± 0.013	0.791 ± 0.012	0.597 ± 0.033	0.027 ± 0.023
EAT+Fed_BVA	0.838 ± 0.009	0.715 ± 0.011	0.226 ± 0.006	0.735 ± 0.020	0.632 ± 0.015	0.106 ± 0.039

Table 5.6: Accuracy of Fashion-MNIST under white-box attacks in IID and non-IID settings

Method	CIFAR-10			CIFAR-100		
	Clean	FGSM	PGD-20	Clean	FGSM	PGD-20
Centralized	0.903 ± 0.003	0.288 ± 0.001	0.074 ± 0.005	0.741 ± 0.003	0.166 ± 0.012	0.032 ± 0.003
FedAvg	0.890 ± 0.002	0.225 ± 0.022	0.062 ± 0.008	0.730 ± 0.003	0.161 ± 0.009	0.035 ± 0.006
FedAvg_AT	0.890 ± 0.003	0.280 ± 0.021	0.099 ± 0.014	0.707 ± 0.003	0.162 ± 0.006	0.048 ± 0.003
Fed_Bias	0.890 ± 0.004	0.280 ± 0.018	0.103 ± 0.012	0.702 ± 0.002	0.163 ± 0.005	0.061 ± 0.003
Fed_Variance	0.889 ± 0.001	0.267 ± 0.014	0.092 ± 0.009	0.710 ± 0.007	0.161 ± 0.005	0.045 ± 0.016
Fed_BVA	0.889 ± 0.003	0.286 ± 0.013	0.104 ± 0.012	0.709 ± 0.003	0.163 ± 0.007	0.062 ± 0.005
EAT	0.833 ± 0.003	0.596 ± 0.003	0.561 ± 0.002	0.661 ± 0.001	0.267 ± 0.002	0.188 ± 0.001
EAT+Fed_BVA	0.833 ± 0.003	0.598 ± 0.002	0.564 ± 0.003	0.657 ± 0.002	0.272 ± 0.003	0.211 ± 0.002

Table 5.7: Accuracy of CIFAR-10 and CIFAR-100 under white-box attacks

IID samples. In Table 5.7, we observe a similar trend where **Fed_BVA** outperforms **FedAvg_AT** on CIFAR-10 and CIFAR-100 (with 0.2% – 10% increases) when defending different types of adversarial examples. Compared to the strong local adversarial training baseline **EAT**, we also observe a maximum 13% accuracy increase when applying its bias-variance oriented baseline **EAT+Fed_BVA**. Overall, the takeaway is that without local adversarial training, using a bias-variance based robust learning framework could outperform other baselines for defending FGSM and PGD attacks. When local adversarial training is allowed (e.g., the client device has powerful computation ability), using bias-variance robust learning with local adversarial training will mostly have the best robustness.

In addition to white-box attacks, we demonstrate the adversarial robustness of the proposed **Fed_BVA** against black-box attacks and Byzantine attacks. As discussed in [303], when the adversary has no prior knowledge regarding the model architecture, it can use a substitute model to generate the adversarial samples and then apply the generated adversarial samples to fool the black-box models. In the experiments, we use a similar idea to evaluate

Method	FGSM				PGD-20			
	R	V	X	M	R	V	X	M
FedAvg	0.707	0.688	0.689	0.793	0.611	0.623	0.597	0.787
FedAvg_AT	0.742	0.710	0.720	0.808	0.695	0.670	0.661	0.808
Fed_Bias	0.740	0.703	0.715	0.799	0.690	0.667	0.654	0.799
Fed_Variance	0.738	0.704	0.719	0.810	0.677	0.656	0.648	0.809
Fed_BVA	0.744	0.706	0.722	0.809	0.693	0.669	0.664	0.809
EAT	0.821	0.806	0.815	0.823	0.819	0.808	0.813	0.822
EAT+Fed_BVA	0.828	0.808	0.817	0.828	0.825	0.809	0.812	0.829

Table 5.8: Adversarial robustness on CIFAR-10 against black-box attacks (R: ResNet18; V: VGG11; X: Xception; M: MobileNetV2)

Method	FGSM				PGD-20			
	R	V	X	M	R	V	X	M
FedAvg	0.561	0.532	0.564	0.613	0.539	0.512	0.537	0.620
FedAvg_AT	0.593	0.545	0.585	0.624	0.580	0.552	0.576	0.636
Fed_Bias	0.593	0.550	0.588	0.621	0.582	0.547	0.576	0.631
Fed_Variance	0.585	0.546	0.583	0.621	0.573	0.545	0.570	0.629
Fed_BVA	0.588	0.547	0.585	0.620	0.576	0.545	0.577	0.631
EAT	0.623	0.592	0.620	0.629	0.622	0.598	0.617	0.634
EAT+Fed_BVA	0.624	0.593	0.619	0.629	0.621	0.599	0.620	0.6333

Table 5.9: Adversarial robustness on CIFAR-100 against black-box attacks (R: ResNet18; V: VGG11; X: Xception; M: MobileNetV2)

the adversarial robustness of the proposed **Fed_BVA** algorithm against black-box attacks. To be specific, we use ResNet18 [69], VGG11 [317], Xception [318], and MobileNetV2 [319] as the substitute threat models for generating the single-step and multi-step adversarial samples. Then we evaluate the robustness of federated learning algorithms based on VGG9 against those generated adversarial samples. The results are reported in Table 5.8 and Table 5.9. It is observed that without adversarial training, **FedAvg** will suffer a maximum of 28% accuracy lost. For comparison, the robust federated learning model with global transmitted perturb samples (i.e., **FedAvg_AT**, **Fed_Bias**, **Fed_Variance**, **Fed_BVA**) will have increase robustness with a maximum of 23% accuracy drop on best baselines. For the computation-demanding local robust training methods (i.e., **EAT** and **EAT+Fed_BVA**), the maximum accuracy drop is only 3%, respectively.

Furthermore, the proposed **Fed_BVA** algorithm can be incorporated with Byzantine-robust aggregation variants, to improve both the adversarial robustness against the corrupted test data set and the Byzantine robustness against the corrupted local model updates. Here we

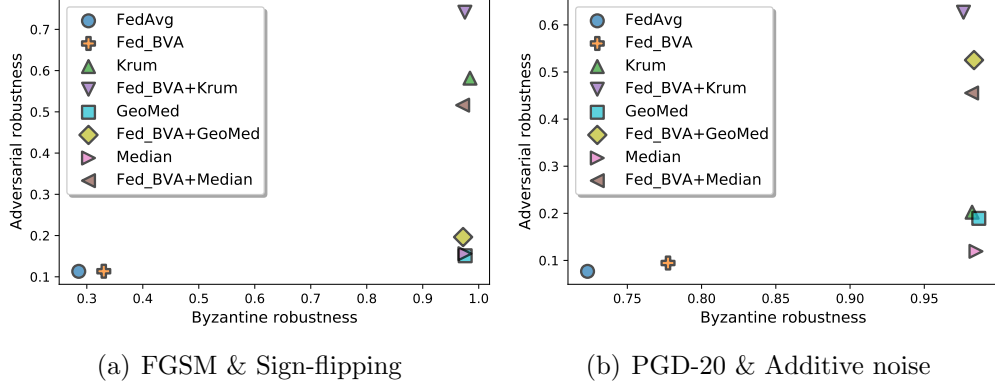


Figure 5.6: Adversarial and Byzantine robustness on MNIST with (a) FGSM attack on the test set and sign-flipping attack on local model updates, and (b) PGD-20 attack on the test set and additive noise attack on local model updates

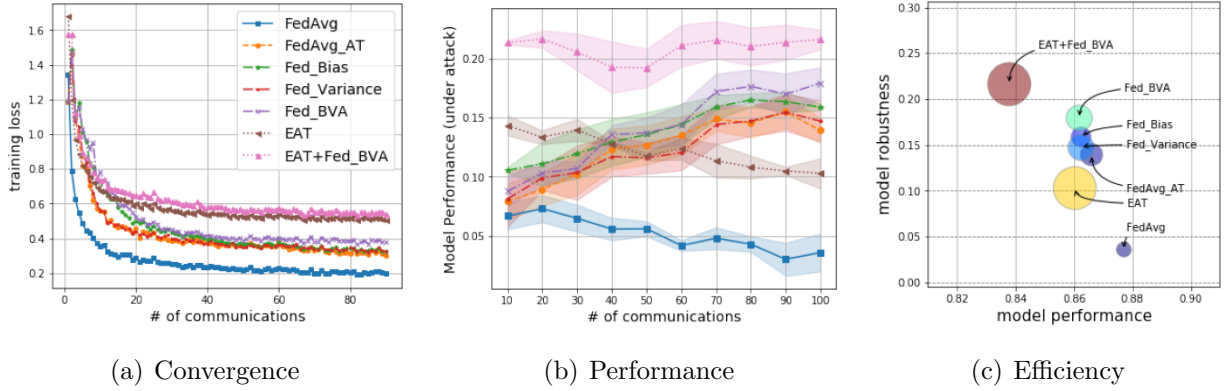


Figure 5.7: Model analysis on Fashion-MNIST (PGD-20 attack)

use sign-flipping attack [320] and additive noise attack [320] to manipulate the local model updates of 40% clients. Then we adopt the Byzantine-robust aggregation mechanisms in our Algorithm 5.2. In this case, we use three popular mechanisms, i.e., Krum [321], Median [322], and GeoMed [323], to aggregate the local model updates. The results are reported in Figure 5.6. It is observed that **Fed_BVA** is robust against adversarial and Byzantine attacks when using Krum as our aggregation strategy.

Ablation Study. The convergence plot of all baselines is presented in Fig. 5.7(a). We observe that **FedAvg** has the best convergence, and all robust training will have a slightly higher loss upon convergence. This matches the observations in [306] which states that training performance may be sacrificed to improve robustness. For the model performance shown in Fig. 5.7(b), we observe that the aggregation of federated learning is vulnerable to adversarial attacks since both **FedAvg** and **EAT** have decreased performance with an increasing

number of server-client communications. Other baselines that utilize asymmetrical communications have increasing robustness with more communication rounds although only a small number of perturbed examples ($n_{\text{share}} = 64$) are transmitted. We also observe that when communication rounds reach 40, **Fed_BVA** starts to outperform **EAT** while the latter is even more resource-demanding than **Fed_BVA** (shown in Fig. 5.7(c), where the pie plot size represents the running time). Overall, bias-variance based adversarial training via asymmetric communication is both effective and efficient for robust federated learning.

CHAPTER 6: PRIVACY AND FAIRNESS OF KNOWLEDGE TRANSFER

In addition to adversarial robustness, privacy, and fairness are two other crucial properties that demonstrate the trustworthiness of transfer learning systems. Therefore, in this chapter, we focus on understanding deep transfer learning from the perspective of data privacy and performance fairness [45, 107]. To this end, we study the privacy-preserving transfer learning framework through the lens of federated knowledge propagation. In this framework, data privacy for each domain is protected, as only the encrypted model parameters are shared. Besides, it is worth exploring whether all participating domains can benefit from federated collaboration in knowledge sharing. This results in the performance fairness concern among the participating domains. To be specific, Section 6.1 introduces a soft (implicit) clustering approach [45] for adaptive parameter propagation among all participating domains. Furthermore, Section 6.2 presents a hard (explicit) clustering approach [107] for optimizing the collaboration structure, such that a domain will only share knowledge with others within the same cluster.

Notation. Table 6.1 provides the notation used in this chapter. Let $\mathcal{X}$ and $\mathcal{Y}$ be the input space and output label space respectively. In this paper, we consider the personalized federated learning setting [188, 190], where there is a central server and K local clients. Each client has access to a private training set $\{x_{ik}, y_{ik}\}_{i=1}^{n_k}$ drawn from a data distribution $\mathbb{P}_k$ in the k^{th} client. Here $x_{ik} \in \mathcal{X}$ and $y_{ik} \in \mathcal{Y}$ denote the input example and output label, respectively. The data set $\{x_{ik}, y_{ik}\}_{i=1}^{n_k}$ is exclusively owned by the k^{th} client and will not be shared with the central server or other clients. We let $\mathcal{L}(\cdot, \cdot)$ denote the loss function. Then the expected prediction error on the k^{th} client is defined as $\mathcal{E}_k(\theta_k) = \mathbb{E}_{(x_k, y_k) \sim \mathbb{P}_k}[\mathcal{L}(h(x_k), y_k; \theta_k)]$ given a hypothesis $h \in \mathcal{H}$, where θ_k denotes the model parameters and $\mathcal{H}$ is a hypothesis space. The empirical error is then defined as $\hat{\mathcal{E}}_k(\theta_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(h(x_{ik}), y_{ik}; \theta_k)$. We denote $\mathbb{1}[a]$ by an indicator function that returns 1 when the argument a is true, otherwise it returns 0.

Notation	Definition
$\mathcal{X}, \mathcal{Y}$	Input space, output space
$\{x_{ik}, y_{ik}\}_{i=1}^{n_k}$	n_k labeled samples
$\mathcal{E}, \hat{\mathcal{E}}$	Expected and empirical prediction errors
$\mathcal{L}(\cdot, \cdot)$	Loss function
$\mathbb{1}[a]$	Indicator function
$\mathcal{H}$	Hypothesis space
$d_{\mathcal{H}\Delta\mathcal{H}}(\cdot, \cdot)$	$\mathcal{H}\Delta\mathcal{H}$ -discrepancy

Table 6.1: Notation used in Chapter 6

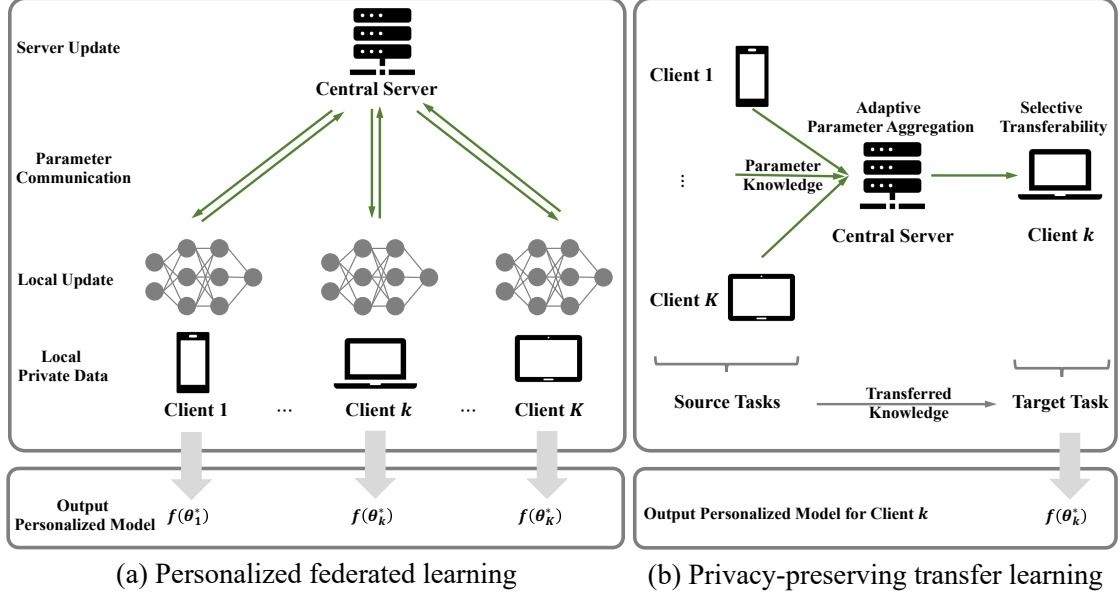


Figure 6.1: Illustration of personalized federated learning. (a) Personalized federated learning aims to find a personalized model for each client. (b) From the perspective of transfer learning, a client learns a personalized model by leveraging latent knowledge from other clients.

6.1 SOFT CLUSTERING FOR PRIVACY-PRESERVING KNOWLEDGE TRANSFER

In this section, we leverage a personalized federated learning framework to understand the data privacy and performance fairness of knowledge transfer among private clients/domains.

6.1.1 Problem Definition

Traditional federated learning algorithms [105, 178] are developed to build a global model by aggregating knowledge from all clients. But it is studied [180] that a single global model might not generalize well on the test data of each client when clients follow different data distributions. To solve this problem, personalized federated learning [182, 188] has been introduced by learning a personalized model for each client (shown in Figure 6.1(a)). Following [182, 188, 324], the problem of personalized federated learning is formally defined as follows.

Definition 6.1. Given a set of local clients with private training sets and a learning algorithm $f(\cdot)$, *personalized federated learning* aims to learn personalized prediction function for each client under the coordination of a central server.

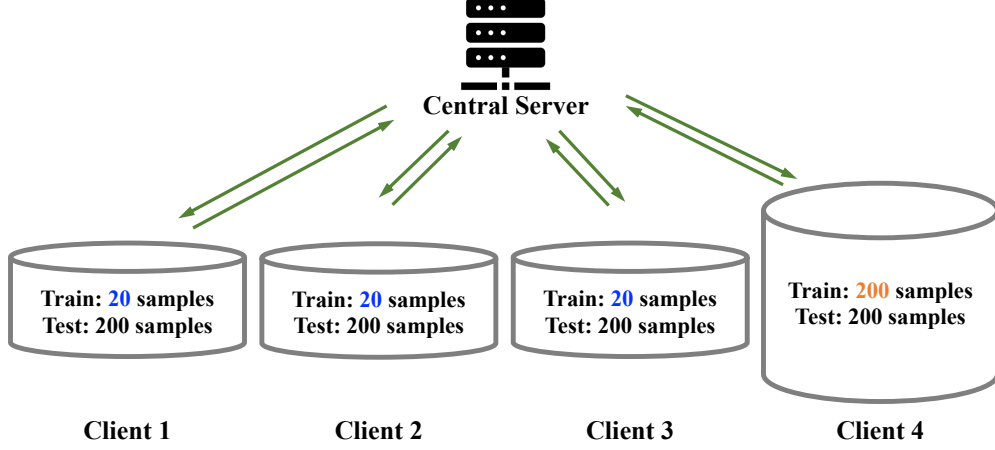
As shown in Figure 6.1, personalized federated learning can be decomposed into a group of

transfer learning problems. For any client k , it considers itself as the target and other clients ($k' \in \{1, \dots, k-1, k+1, \dots, K\}$) as the sources. From the perspective of transfer learning, each client aims to improve the generalization performance of a learning algorithm on this client, by transferring the knowledge from other clients. Nevertheless, we show in Figure 6.2 that some clients might have worse performance than their local training counterparts (i.e., each client trains the model over its own local data without communication across clients). As illustrated in Figure 6.2(a), four local clients collaborate in training models. It can be seen from Figure 6.2(b) that compared to local training (denoted as “LOCAL”), personalized federated learning approaches (e.g., LG-FedAvg [183], Ditto [190], FedAMP [193]) improve the overall prediction performance (e.g., test accuracy over all clients). However, we observe that not all clients can benefit from federated training, e.g., client 4 has lower accuracy than LOCAL. Intuitively, this result indicates that client 4 is not incentivized to participate in federated training, because it introduces communication costs (by sharing model parameters) and achieves no performance improvement.

The observation above motivates us to re-think personalized federated learning with the following fundamental research questions. *Q1: Are all the clients incentivized to participate in federated collaboration?* *Q2: How do clients maximally benefit from federated collaboration under data heterogeneity across clients?* These two research questions are strongly correlated. That is because it is revealed [26, 38] that the target task can benefit from transfer learning when it has a limited number of training samples (*Q1*) and it shares similar data distribution with the source task (*Q1&Q2*). More specifically, on one hand, a target client is incentivized to participate in federated collaboration when it has limited training samples and there exist other clients sharing similar data distributions. This also explains that in Figure 6.2, with adequate training samples, Client 4 is more likely to suffer from the negative transfer (i.e., worse performance compared to LOCAL). On the other hand, a target client would collaborate with a source client sharing similar data distributions (indicating they have some common knowledge). The transferred knowledge from the source client will have a negative impact on the target learner if the distribution shift between clients is large.

6.1.2 Proposed Algorithm

As pointed out in previous work [26, 38], when transferring the knowledge from a source domain to a target domain, the transfer performance is strongly correlated with both distribution differences between tasks and the number of training samples on the target domain. This motivates us to propose a federated parameter propagation framework (FEDORA) [45]



(a) Imbalanced training samples across clients

Model	Accuracy				Average Accuracy
	Client 1	Client 2	Client 3	Client 4	
LOCAL	0.5270	0.4840	0.4980	0.8110	0.5800
FedAvg	0.3755	0.4420	0.6455	0.7965	0.5649
LG-FedAvg	0.5440	0.5115	0.5430	0.8095	0.6020
Ditto	0.4095	0.4810	0.6465	0.8095	0.5866
FedAMP	0.5300	0.5210	0.5415	0.8105	0.6008
FEDORA (ours)	0.5565	0.5675	0.5850	0.8195	0.6321

(b) Results of personalized federated learning

Figure 6.2: Personalized federated learning on non-IID clients with imbalanced training samples. There are four clients with data drawn from Rotated MNIST [299]: Client 1 (0°), Client 2 (30°), Client 3 (60°), and Client 4 (90°), and several baselines, including FedAvg [105], LG-FedAvg [183], Ditto [190], and FedAMP [193]. The baselines suffer from the negative transfer in Client 4, i.e., lower accuracy than LOCAL.

for personalized federated learning. The overall objective function is formulated as follows.

$$\begin{aligned}
 \mathcal{J} \left(\{\theta_k\}_{k=1}^K, \{\hat{\theta}_k\}_{k=1}^K \right) = & \sum_{k=1}^K \frac{1}{\lambda_k n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \sum_{k=1}^K \|\theta_k - \hat{\theta}_k\|_2^2 \\
 & + \frac{\alpha}{2} \sum_{k=1}^K \sum_{k'=1}^K \frac{w_{kk'}}{D_{kk'}} \left\| \hat{\theta}_k - \hat{\theta}_{k'} \right\|_2^2
 \end{aligned} \tag{6.1}$$

where $w_{kk'}$ denotes the distribution similarity between client k and client k' and $D_{kk} = \sum_{k'=1}^K w_{kk'}$. θ_k is the personalized parameters and $\hat{\theta}_k$ is the auxiliary personalized parameters in client k . Here $\alpha > 0$ and $\lambda_k > 0$ balance different terms in our objective function.

Intuitively, the first term represents empirical prediction error for learning personalized parameters θ_k ($k = 1, \dots, K$). The second term enforces θ_k to approximate auxiliary per-

sonalized parameters $\hat{\theta}_k$. In this case, $\hat{\theta}_k$ would encode the knowledge transferred from other clients. λ_k indicates whether client k would benefit from the received knowledge. $\lambda_k \rightarrow 0$ implies that client k would focus more on local training over its own training samples, and the received knowledge $\hat{\theta}_k$ might hurt the generalization performance of client k . The third term of our objective function is to regularize the auxiliary personalized parameters based on distribution similarity between clients. That is, when two clients are distributionally similar, they would share similar model parameters.

It can be seen that in the special case where $\hat{\theta}_k = \theta_k$ for all $k \in \{1, \dots, K\}$ and $\lambda_1 = \dots = \lambda_K = \lambda$, our objective function is equivalent to existing federated multi-task learning algorithms, e.g., MOCHA [188], FedU [192], with the objective function.

$$\min_{\theta_1, \dots, \theta_K} \sum_{k=1}^K \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \frac{\alpha\lambda}{2} \sum_{k=1}^K \sum_{k'=1}^K \frac{w_{kk'}}{D_{kk}} \|\theta_k - \theta_{k'}\|_2^2 \quad (6.2)$$

Compared to previous works [188, 192], **FEDORA** is more flexible because client k can choose whether to collaborate by dynamically adjusting λ_k during model training.

By minimizing the objective function of Eq. (6.1), we iteratively update the parameters θ_k and $\hat{\theta}_k$, including (a) fixing $\hat{\theta}_k$ and updating θ_k for $k = 1, \dots, K$ in parallel, i.e.,

$$\min_{\theta_k} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \lambda_k \|\theta_k - \hat{\theta}_k\|_2^2 \quad (6.3)$$

and (b) fixing θ_k and updating $\hat{\theta}_k$, i.e.,

$$\min_{\hat{\theta}_1, \dots, \hat{\theta}_K} \sum_{k=1}^K \|\theta_k - \hat{\theta}_k\|_2^2 + \frac{\alpha}{2} \sum_{k=1}^K \sum_{k'=1}^K \frac{w_{kk'}}{D_{kk}} \|\hat{\theta}_k - \hat{\theta}_{k'}\|_2^2 \quad (6.4)$$

In this case, Eq. (6.3) optimizes the personalized parameters θ_k via empirical risk minimization with respect to local training data in client k . It implies that θ_k ($k = 1, \dots, K$) can be updated locally in parallel. In contrast, Eq. (6.4) updates $\hat{\theta}_k$ globally, because it requires the auxiliary parameters $\hat{\theta}_{k'}$ ($k' \neq k$) from other clients. Therefore, following [105, 106], we can roughly summarize the federated training procedures of **FEDORA** as follows (shown in Algorithm 6.1).

1. *Forward Communication*: The server broadcasts the auxiliary personalized parameters to clients, e.g., send $\hat{\theta}_k$ to client k ;
2. *Client Update*: Each client updates its own personalized parameters θ_k via Eq. (6.3);

Algorithm 6.1: FEDORA

Input: K private clients with $\{x_{ik}, y_{ik}\}_{i=1}^{n_k}$ ($k = 1, \dots, K$);
Output: Personalized model parameters $\{\theta_k\}_{k=1}^K$.
foreach *client* $k = 1, \dots, K$ *in parallel* **do**
 Compute base vectors $\mathbf{U}_k$ of subspace in client k ;
 Upload $\mathbf{U}_k$ to the central server;
end
Estimate client similarity $w_{kk'}$ via Eq. (6.6) on the server;
Initialize personalized parameters θ_k on local client;
Initialize auxiliary parameters $\theta_{k'}$ on central server;
foreach *each round* $r = 0, 1, \dots$, **do**
 foreach *client* $k = 1, \dots, K$ *in parallel* **do**
 Estimate λ_k using Eq. (6.22);
 foreach *local epoch* $i = 1, \dots, E$ **do**
 Update personalized parameters θ_k using Eq. (6.23);
 end
 Upload θ_k to the central server;
 end
 Update auxiliary parameters using Eq. (6.12) or Eq. (6.13);
 Send updated auxiliary parameters back to local clients;
end

3. *Backward Communication:* Each client uploads the personalized parameters θ_k back to the server;
4. *Server Update:* The server updates the auxiliary personalized parameters $\hat{\theta}_k$ ($k = 1, \dots, K$) via Eq. (6.4).

We see that similar to FedAvg [105], the communication cost of FEDORA is determined by $\hat{\theta}_k$ and θ_k during federated training, i.e., a client shares θ_k to the central server and receives $\hat{\theta}_k$ from the server. Thus, it is much cheaper than existing federated multi-task learning algorithms, e.g., MOCHA [188], FedFOMO [324]. This is because each client in these algorithms receives the model parameters from multiple clients.

Next, we present the detailed training procedures of the proposed FEDORA framework.

Preprocessing. The intuition behind FEDORA framework is that two clients share similar personalized parameters if they are distributionally similar. Before introducing the iterative optimization solution of FEDORA, we first estimate the data distribution similarity between clients in the context of federated learning.

Inspired by [325], we measure the client similarity (i.e., $w_{kk'}$ between client k and client k') by exploring the subspaces induced by training samples across clients. As shown in [26], the

distribution shift across clients can be induced by both input features and output labels. As a result, we focus on estimating the client similarity over the joint data distribution $\mathcal{X} \times \mathcal{Y}$. Given a training set $\mathbf{X}_k = \{x_{1k}, \dots, x_{n_kk}\}$ with associated labels $\mathbf{Y}_k = \{y_{1k}, \dots, y_{n_kk}\}$ in client k , we perform truncated SVD on $\mathbf{X}_k \circ \mathbf{Y}_k$ and obtain the subspace representation $\mathbf{U}_k = [u_1^k, \dots, u_p^k]$ ($p \ll \text{rank}(\mathbf{X}_k)$), i.e., $\mathbf{X}_k \circ \mathbf{Y}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$. Here $\circ$ denotes the vector concatenation. Then, the similarity of two orthogonal subspaces can be measured by the principal angles. To be specific, given two subspaces $\mathcal{U}_k = \text{span}\{u_1^k, \dots, u_p^k\}$ and $\mathcal{U}_{k'} = \text{span}\{u_1^{k'}, \dots, u_p^{k'}\}$, the principal angles [326] are formally defined as:

$$\begin{aligned} \zeta_1^{kk'} &= \min_{a_1^k \in \mathcal{U}_k, b_1^{k'} \in \mathcal{U}_{k'}} \arccos \left(\frac{\langle a_1^k, b_1^{k'} \rangle}{\|a_1^k\| \|b_1^{k'}\|} \right) \\ &\vdots \\ \zeta_p^{kk'} &= \min_{\substack{a_p^k \in \mathcal{U}_k, b_p^{k'} \in \mathcal{U}_{k'} \\ a_p^k \perp a_1^k, \dots, a_{p-1}^k \\ b_p^{k'} \perp b_1^{k'}, \dots, b_{p-1}^{k'}}} \arccos \left(\frac{\langle a_p^k, b_p^{k'} \rangle}{\|a_p^k\| \|b_p^{k'}\|} \right) \end{aligned} \quad (6.5)$$

The orthogonal subspaces $\mathcal{U}_k$ and $\mathcal{U}_{k'}$ are identical when $\zeta_1 = \dots = \zeta_p = 0$. Then based on the principal angles, we define the similarity between client k and client k' as follows.

$$w_{kk'} = \sum_{i=1}^p \cos \zeta_i^{kk'} \quad (6.6)$$

Previous work [326] shows that the principal angles can be efficiently calculated by the following matrix decomposition.

$$\mathbf{U}_k^T \mathbf{U}_{k'} = \mathbf{P} \left(\text{diag} \left(\cos \zeta_1^{kk'}, \dots, \cos \zeta_p^{kk'} \right) \right) \tilde{\mathbf{P}}^T \quad (6.7)$$

where $\mathbf{P}$ and $\tilde{\mathbf{P}}$ are two orthogonal matrices by performing SVD on $\mathbf{U}_k^T \mathbf{U}_{k'}$ and $\cos \zeta_1^{kk'}, \dots, \cos \zeta_p^{kk'}$ are the corresponding eigenvalues. The estimation of client similarity in federated learning is summarized in Algorithm 6.1. Each client extracts the orthogonal subspace representation $\mathbf{U}_k = [u_1^k, \dots, u_p^k]$ over its own training samples, and then uploads $\mathbf{U}_k$ to the central server. The central server would estimate the pair-wise client similarity using Eq. (6.6) and Eq. (6.7).

Compared to previous works [108, 187, 193, 324], the client similarity in Eq. (6.6) has the following advantages. First, our client similarity measure Eq. (6.6) is directly corre-

lated with the data distributions of clients. But previous works implicitly estimate the client similarity using local model parameters. Second, the estimation of Eq. (6.6) is computationally efficient, since it only requires one client-server communication. In contrast, previous works would have to calculate the client similarity in every training round of federated learning. Besides, Lemma 6.1 shows the connections between our framework and previous works [190, 191] by using constant client similarity.

Lemma 6.1. With different measures of client similarity, our objective function Eq. (6.1) has the following special cases.

- If $w_{kk'} = 1$ for $k = k'$ and $w_{kk'} = 0$ for $k \neq k'$, then we have $\hat{\theta}_k = \theta_k$. Moreover, the optimization problem of Eq. (6.1) becomes standard local training with the objective function

$$\min_{\{\theta_k\}_{k=1}^K} \sum_{k=1}^K \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) \quad (6.8)$$

- If $w_{kk'} = n_{k'}$ for $k, k' \in \{1, \dots, K\}$ and $\alpha \rightarrow \infty$, then we have $\hat{\theta}_k = \sum_{k'=1}^K \frac{n_{k'}}{\sum_{j=1}^K n_j} \theta_{k'}$. Moreover, the optimization problem of Eq. (6.1) becomes customized personalized federated learning (e.g., Ditto [190]) with the objective function

$$\min_{\{\theta_k\}_{k=1}^K} \sum_{k=1}^K \left(\frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \lambda_k \|\theta_k - w\|_2^2 \right) \quad (6.9)$$

where $w = \sum_{k'=1}^K \frac{n_{k'}}{\sum_{j=1}^K n_j} \theta_{k'}$ is the weighted personalized parameters.

Proof. In the first case, if $w_{kk'} = 1$ for $k = k'$ and $w_{kk'} = 0$ for $k \neq k'$, our objective function Eq. (6.1) becomes

$$\mathcal{J}(\{\theta_k\}_{k=1}^K, \{\hat{\theta}_k\}_{k=1}^K) = \sum_{k=1}^K \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \lambda_k \left(\sum_{k=1}^K \|\theta_k - \hat{\theta}_k\|_2^2 \right) \quad (6.10)$$

It can be seen that the optimal solution is achieved at $\hat{\theta}_k = \theta_k$ for all client k . Each client updates its personalized parameters θ_k locally without communication with others.

In the second case, if $w_{kk'} = n_{k'}$ for $k, k' \in \{1, \dots, K\}$ and $\alpha \rightarrow \infty$, using Lemma 6.2 we have

$$\hat{\theta}_k^{(m)} = \frac{\alpha}{(1 + \alpha)D_{kk}} \sum_{k'=1}^K w_{kk'} \hat{\theta}_{k'}^{(m-1)} + \frac{1}{1 + \alpha} \theta_k \rightarrow \sum_{k'=1}^K \frac{n_{k'}}{\sum_{j=1}^K n_j} \hat{\theta}_{k'}^{(m-1)} \quad (6.11)$$

Since $\hat{\theta}_{k'}^0 = \theta_{k'}$, we have $\hat{\theta}_k^{(m)} \rightarrow \sum_{k'=1}^K \frac{n_{k'}}{\sum_{j=1}^K n_j} \theta_{k'}$ when $\alpha \rightarrow \infty$. That is, all clients share the

same auxiliary parameters $w = \sum_{k'=1}^K \frac{n_{k'}}{\sum_{j=1}^K n_j} \theta_{k'}$. QED.

Parameter Propagation for Server Update. The central server updates the auxiliary personalized parameters $\hat{\theta}_k$ ($k = 1, \dots, K$) by minimizing the sub-problem Eq. (6.4) of our objective function. The following lemma shows that Eq. (6.4) has a closed-form solution.

Lemma 6.2. Let $\Theta = [\theta_1, \theta_2, \dots, \theta_K]^T \in \mathbb{R}^{K \times d_\theta}$ and $\hat{\Theta} = [\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_K]^T \in \mathbb{R}^{K \times d_\theta}$ be the personalized model parameters and auxiliary personalized model parameters respectively, where d_θ denotes the dimensionality of model parameters. The optimal solution to the objective of Eq. (6.4) satisfies the equation

$$\hat{\Theta}^* = (1 - \kappa) (I - \kappa D^{-1} W)^{-1} \Theta \quad (6.12)$$

where $\kappa = \frac{\alpha}{1+\alpha}$. It has an equivalent iterative solution:

$$\hat{\Theta}^{(m)} = (\kappa D^{-1} W) \hat{\Theta}^{(m-1)} + (1 - \kappa) \Theta \quad (6.13)$$

where $\hat{\Theta}_k^{(0)} = \Theta$. Moreover, $\hat{\Theta}^{(m)}$ converges, i.e., $\lim_{m \rightarrow \infty} \hat{\Theta}^{(m)} = \hat{\Theta}^*$, when m goes to infinity.

Proof. When removing the terms without $\hat{\theta}_k$ ($k = 1, \dots, K$), the optimization problem of Eq. (6.1) becomes

$$\min_{\hat{\theta}_k} \sum_{k=1}^K \|\theta_k - \hat{\theta}_k\|_2^2 + \frac{\alpha}{2} \cdot \sum_{k=1}^K \sum_{k'=1}^K \frac{w_{kk'}}{D_{kk}} \left\| \hat{\theta}_k - \hat{\theta}_{k'} \right\|_2^2 \quad (6.14)$$

It can be re-written as:

$$\text{Tr} \left((\Theta - \hat{\Theta})^T (\Theta - \hat{\Theta}) \right) + \alpha \cdot \text{Tr} \left(\hat{\Theta}^T (I - D^{-1} W) \hat{\Theta} \right) \quad (6.15)$$

Setting the derivative of Eq. (6.1) with respect to $\hat{\theta}_k$ ($k = 1, \dots, K$) to zero gives the result

$$\begin{aligned} 2 \left(\hat{\Theta} - \Theta \right) + 2\alpha \cdot (I - D^{-1} W) \hat{\Theta} &= 0 \\ (1 + \alpha) \hat{\Theta} - \alpha \cdot (D^{-1} W) \hat{\Theta} &= \Theta \\ \left(I - \frac{\alpha}{1 + \alpha} D^{-1} W \right) \hat{\Theta} &= \frac{1}{1 + \alpha} \Theta \end{aligned} \quad (6.16)$$

Thus, the following holds

$$\hat{\Theta}^* = (1 - \kappa) (I - \kappa D^{-1} W)^{-1} \Theta \quad (6.17)$$

where $\kappa = \frac{\alpha}{1+\alpha}$.

For the iterative solution, we have

$$\begin{aligned}
\hat{\Theta}^{(m)} &= (\kappa D^{-1}W) \hat{\Theta}^{(m-1)} + (1 - \kappa)\Theta \\
&= (\kappa D^{-1}W)^2 \hat{\Theta}^{(m-2)} + (\kappa D^{-1}W) (1 - \kappa)\Theta + (1 - \kappa)\Theta \\
&= \dots = (\kappa D^{-1}W)^m \hat{\Theta}^{(0)} + \sum_{i=0}^{m-1} (\kappa D^{-1}W)^i (1 - \kappa)\Theta
\end{aligned} \tag{6.18}$$

It is easy to show that for the spectral radius of $\kappa D^{-1}W$, we have:

$$\rho(\kappa D^{-1}W) \leq \alpha \|\kappa D^{-1}W\|_{\infty} = \kappa = \frac{\alpha}{1 + \alpha} < 1 \tag{6.19}$$

Therefore, when m goes to infinity, $\left(\kappa D^{-\frac{1}{2}}W D^{-\frac{1}{2}}\right)^m$ converges, and

$$\begin{aligned}
\lim_{m \rightarrow \infty} (\kappa D^{-1}W)^m &= \mathbf{0} \\
\lim_{m \rightarrow \infty} \sum_{i=0}^{m-1} (\kappa D^{-1}W)^i &= (I - \kappa D^{-1}W)^{-1}
\end{aligned} \tag{6.20}$$

which completes the proof. QED.

Lemma 6.2 illustrates the intuition behind server updates of **FEDORA**. To be specific, Eq. (6.13) can be rewritten as

$$\hat{\theta}_k^{(m)} = \frac{\alpha}{(1 + \alpha)D_{kk}} \sum_{k'=1}^K w_{kk'} \hat{\theta}_{k'}^{(m-1)} + \frac{1}{1 + \alpha} \theta_k \tag{6.21}$$

We see that client k would be more likely to iteratively aggregate the knowledge from client k' , when they have higher distribution similarity $w_{kk'}$. Moreover, similar to personalized PageRank [327], $\hat{\theta}_k$ has probability $\frac{1}{1+\alpha}$ of being updated via its counterpart θ_k , and probability $\frac{\alpha}{1+\alpha}$ of being updated using other clients.

Personalized Training for Client Update. When receiving the auxiliary personalized parameters $\hat{\theta}_k$ from the server, client k would update its personalized model parameters θ_k by minimizing the sub-problem Eq. (6.3) of our objective function. In this case, λ_k can be considered as an indicator to show whether the transferred knowledge $\hat{\theta}_k$ can benefit the personalized model training of client k . From the perspective of transfer learning [2], the goal of personalized learning on client k is to improve the generalization performance by leveraging the knowledge from other clients (encoded by $\hat{\theta}_k$). Therefore, we define the

selection parameter λ_k by empirically evaluating the generalization performance of auxiliary personalized parameters $\hat{\theta}_k$.

$$\lambda_k = \max \left(\epsilon, \tilde{\mathcal{L}}_k(\theta_k) - \tilde{\mathcal{L}}_k(\hat{\theta}_k) \right) \quad (6.22)$$

where $\tilde{\mathcal{L}}_k(\cdot)$ denotes the prediction error on the validation set of client k and $\epsilon > 0$ is a constant. The intuition behind Eq. (6.22) is that the auxiliary personalized parameters $\hat{\theta}_k$ will guide the personalized model training of client k , only when $\hat{\theta}_k$ can generalize better than θ_k . In the experiments, we show that this simple selection strategy over λ_k can largely mitigate the negative transfer of local clients.

When λ_k is learned, client k would update its personalized parameters θ_k using standard gradient descent as follows.

$$\theta_k \leftarrow \theta_k - \eta \nabla_{\theta_k} \left(\frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(x_{ik}, y_{ik}; \theta_k) + \lambda_k \|\theta_k - \hat{\theta}_k\|_2^2 \right) \quad (6.23)$$

where $\eta > 0$ is the learning rate.

The overall training procedures of **FEDORA** are summarized in Algorithm 6.1. It first estimates the client similarity based on the data distributions of local clients. Then **FEDORA** iteratively updates the personalized parameters θ_k and auxiliary parameters $\hat{\theta}_k$ by minimizing the overall objective function in Eq. (6.1).

6.1.3 Generalization to New Clients

In addition to standard model training, we show that the proposed **FEDORA** framework can easily generalize to new clients. Specifically, we consider two federated learning scenarios: (1) new clients have labeled training samples, and (2) new clients only have unlabeled training samples. In the first case, the new client extracts the orthogonal subspace representation $\mathbf{U}_{K+1}$ and then uploads $\mathbf{U}_{K+1}$ to the central server. The server would estimate the distribution similarity between this new client and other clients. Based on the distribution similarity, the server calculates the auxiliary parameters $\hat{\theta}_{K+1}$ via Eq. (6.12) and sends it back to the new client. Finally, the new client can optimize its personalized parameters θ_{K+1} (see Eq. (6.23)) by regularizing θ_{K+1} with the received auxiliary parameters $\hat{\theta}_{K+1}$. We summarize the training procedures in Algorithm 6.2.

In the second scenario where new clients only have unlabeled training samples, these clients do not support building pure locally trained models or fine-tuning the received parameters from the federated learning system. We show that our **FEDORA** framework can generate

Algorithm 6.2: Generalization to a New Client

Input: A new client with data $\{x_{i(K+1)}, y_{i(K+1)}\}_{i=1}^{n_{K+1}}$;
Output: Personalized model parameter θ_{K+1} .
Compute base vectors $\mathbf{U}_{K+1}$ of subspace in the new client;
Upload $\mathbf{U}_{K+1}$ to the central server;
Estimate client similarity between this new client and old ones;
Calculate the auxiliary parameters $\hat{\theta}_{K+1}$;
Send the auxiliary parameters $\hat{\theta}_{K+1}$ to the new client;
Update personalized parameters θ_{K+1} using Eq. (6.23).

the auxiliary parameters $\hat{\theta}_{K+1}$ for the new client under mild assumptions. If all the clients follow the covariate shift assumption [2], i.e., they have the same labeling function $\mathbb{P}_k(y|x) = \mathbb{P}_{k'}(y|x)$ but different marginal distributions $\mathbb{P}_k(y|x) \neq \mathbb{P}_{k'}(y|x)$, we can estimate the client similarity based on the subspace representation $\mathbf{U}_k$ induced by features $\mathbf{X}_k$ (i.e., $\mathbf{X}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$). In this case, when a new client with unlabeled training samples appears, FEDORA can estimate the client similarity between this new client and other ones based on the feature-guided subspace representation. Then the parameter propagation mechanism in Eq. (6.12) would generate the auxiliary parameters $\hat{\theta}_{K+1}$ for the new client. The objective function of Eq. (6.3) shows that when no labeled training samples are available on the new client, it achieves the optimal solution at $\theta_{K+1} = \hat{\theta}_{K+1}$.

6.1.4 Experiment

In the experiments, we evaluate the effectiveness of FEDORA in mitigating negative transfer across various personalized federated learning settings.

Data Sets. We use the following data sets: MNIST [299], Fashion-MNIST [314], CIFAR-10 [315], Yearbook [328], GTSRB [329], and agriculture data [223, 287]. Following [105, 109], we consider two methods to partition the non-IID data over clients. For MNIST, Fashion-MNIST, and GTSRB, we partition the training images into K clients, where the images in each client are rotated with a certain angle. For the rotated MNIST, Fashion-MNIST, and GTSRB, the data heterogeneity among clients is induced by the feature shift. There are 36, 72, and 10 clients in rotated MNIST, Fashion-MNIST, and GTSRB respectively. For CIFAR-10, we follow the pathological non-IID setting where each client has data with at most two classes. In addition, Yearbook consists of 37921 frontal-facing American high school yearbook photos from 1930 to 2013 for gender classification. Agriculture data sets contain maize and soybean data collected from Illinois and Nebraska over the years. The task

in the agriculture data sets is to predict diverse traits (e.g., Nitrogen) of plants related to the plants' growth using leaf hyperspectral reflectance. Therefore, Yearbook and agriculture data sets can be naturally partitioned into different clients based on the data collection time. Then Yearbook has 84 clients and Agriculture has 11 clients. Data heterogeneity exists in Yearbook and agriculture data sets because the underlying sampling distribution might be changing over time.

Baselines. The baselines used in the experiments include (global) federated learning approaches: FedAvg [105], FedProx [179] and their variants with fine-tuning (FedAvg+FT and FedProx+FT), and the following personalized federated learning approaches.

- **LOCAL:** Each client trains its own personalized model without knowledge communication.
- **Parameter Decoupling:** LG-FedAvg [183], FedPer [184] and pFedHN [194] partially share the model parameters indicating the shared common knowledge among clients.
- **Model Interpolation:** APFL [186] and Ditto [190] learn personalized models using a mixture of global and local models.
- **Clustering:** IFCA [109] and FeSEM [187] alternately estimate the cluster identities and optimize model parameters for each cluster.
- **Multi-Task Learning:** FedFOMO [324], FedAMP [193] and FedU [192] explicitly capture relationships among the clients with different data distributions.

Evaluation Metric. When evaluating the efficacy of a federated learning system, the commonly used metrics in previous works [188, 190, 192] are average prediction results over all the clients, e.g., average classification accuracy for image classification [105]. In addition to average accuracy indicating the overall performance of the federated learning system, we also consider two additional evaluation metrics as follows.

Definition 6.2. (Relative Accuracy) Given a target client k , the relative accuracy of personalized federated learning is defined as

$$\text{R-Acc}(\theta_k^*) = \frac{\text{Acc}(\theta_k^*) - \text{Acc}(\theta_k^{\text{LOCAL}})}{\text{Acc}(\theta_k^{\text{LOCAL}})} \quad (6.24)$$

where $\text{Acc}(\cdot)$ denotes the test accuracy, θ_k^* denotes the parameters learned by personalized federated learning (i.e., with federated collaboration) on client k , and θ_k^{LOCAL} denotes the parameters learned by local training (i.e., without federated collaboration) on client k .

Model	Rotated MNIST			Rotated Fashion-MNIST			CIFAR-10			Yearbook			Rotated GTSRB		
	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$
LOCAL	0.7642	-	-	0.7057	-	-	0.7617	-	-	0.8068	-	-	0.5531	-	-
FedAvg	0.6889	-0.0976	0	0.6441	-0.0847	0.1250	0.6531	-0.1382	0.3000	0.8165	0.0191	0.5119	0.6375	0.2006	0.8000
FedAvg+FT	0.7411	-0.0293	0.3056	0.6848	-0.0283	0.3472	0.7992	0.0513	0.9000	0.8180	0.0195	0.5595	0.6375	0.2185	0.8000
FedProx	0.5375	-0.2962	0	0.5968	-0.1521	0	0.6984	-0.0799	0.2000	0.7995	-0.0027	0.4405	0.7031	0.3384	0.9000
FedProx+FT	0.6893	-0.0973	0.0278	0.6788	-0.0358	0.3056	0.7953	0.0460	0.9000	0.8227	0.0258	0.5595	0.7312	0.3984	0.9000
LG-FedAvg	0.7804	0.0214	0.9444	0.7137	0.0115	0.7361	0.7656	0.0054	0.8000	0.8072	0.0007	0.8095	0.5938	0.1044	0.8000
FedPer	0.7741	0.0135	0.6389	0.6725	-0.0457	0.1389	0.8352	0.0990	1.0000	0.7974	-0.0096	0.4167	0.6687	0.2607	0.8000
pFedHN	0.8004	0.0486	0.8611	0.7215	0.0249	0.6944	0.7766	0.0221	0.6000	0.8263	0.0313	0.6310	0.4500	-0.1778	0.2000
APFL	0.7871	0.0303	0.8889	0.7134	0.0112	0.7639	0.8258	0.0866	0.9000	0.8128	0.0081	0.7619	0.6469	0.1995	0.9000
Ditto	0.7806	0.0220	0.7222	0.7212	0.0232	0.7361	0.8078	0.0630	0.9000	0.8148	0.0112	0.6429	0.7063	0.3345	0.8000
IFCA	0.7915	0.0365	0.6944	0.7305	0.0370	0.7639	0.8227	0.0828	0.9000	0.8122	0.0076	0.5238	0.7344	0.3852	1.0000
FeSEM	0.7720	0.0110	0.6111	0.7074	0.0051	0.5278	0.8547	0.1255	1.0000	0.7821	-0.0258	0.3810	0.6562	0.2274	0.8000
FedFOMO	0.7749	0.0140	0.9167	0.7110	0.0076	0.7639	0.8242	0.0797	1.0000	0.8111	0.0059	0.7619	0.6156	0.1321	1.0000
FedU	0.7837	0.0260	0.8889	0.7208	0.0225	0.8056	0.7836	0.0295	0.9000	0.8092	0.0047	0.5357	0.5625	0.0549	0.6000
FedAMP	0.7869	0.0298	1.0000	0.7203	0.0213	0.8056	0.7953	0.0457	0.8000	0.8111	0.0059	0.6905	0.5625	0.0233	0.6000
FEDORA	0.8251	0.0806	1.0000	0.7433	0.0548	0.9028	0.8570	0.1288	1.0000	0.8341	0.0386	0.9167	0.7375	0.3773	1.0000

Table 6.2: Results on image data sets (Acc: average accuracy, R-Acc: average relative accuracy, PTR: positive transferability ratio)

Definition 6.3. (Positive Transferability Ratio) Given a federated learning system with K clients, the positive transferability ratio of personalized federated learning is defined as

$$\text{PTR}(\theta_1^*, \dots, \theta_K^*) = \frac{\sum_{k=1}^K \mathbb{1}[\text{Acc}(\theta_k^*) - \text{Acc}(\theta_k^{\text{LOCAL}})]}{K} \quad (6.25)$$

where $\mathbb{1}[\cdot]$ is an indicator function that returns 1 when the argument is true, otherwise it returns 0.

Both relative accuracy and positive transferability ratio measure the knowledge transfer performance when client k participates in federated collaboration. Notably, the positive transferability ratio $\text{PTR}(\theta_1^*, \dots, \theta_K^*)$ indicates whether negative transfer happens in the federated learning system. Higher $\text{PTR}(\theta_1^*, \dots, \theta_K^*)$ implies that most clients benefit from federated collaboration. $\text{R-Acc}(\theta_k^*)$ indicates fine-grained performance improvement/degradation that client k achieves from federated collaboration.

Model Configuration. In the experiments, we use a 3-layer MLP for MNIST, Fashion-MNIST, Yearbook, and agriculture data sets. A 5-layer CNN is adopted for CIFAR-10 and GTSRB. We use the cross-entropy loss for image classification (MNIST, Fashion-MNIST, Yearbook, CIFAR-10, and GTSRB) and mean square error as the loss function for agriculture analysis. In addition, we set $p = 1$ and $\alpha = 1$ in the experiments.

Main Results. Table 6.2 and Table 6.3 provide the results of personalized federated learning on image and agriculture data sets (the best results are indicated in bold) where each client has the same number of training samples. We report the average classification accuracy, average relative accuracy, and positive transferability ratio for image classification in

Model	MAE ↓	R-MAE ↑	PTR ↑
LOCAL	0.5576	-	-
FedPer	0.4399	0.1224	0.6364
pFedHN	0.4262	0.1289	0.6364
APFL	0.4263	0.1398	0.8182
Ditto	0.4331	0.1281	0.8182
FedFOMO	0.4488	0.0973	0.8182
FedU	0.5421	0.0289	0.8182
FedAMP	0.4433	0.1103	0.8182
FEDORA	0.4185	0.1499	0.9091

Table 6.3: Results on agriculture data set



Figure 6.3: Relative performance improvement of each client on Rotated MNIST

Table 6.2. For the regression task in the agriculture data set, we use the Mean Absolute Error (MAE) between the predicted outputs and the ground-truth outputs (i.e., Nitrogen content). Similarly, we report the average MAE, average relative MAE, and positive transferability ratio in Table 6.3. We have the following observations. (1) Higher accuracy does not imply that all the clients benefit from federated collaboration (e.g., IFCA [109] obtains much higher accuracy on Rotated MNIST than LG-FedAvg [183], but it suffers from negative transfer on local clients). Relative accuracy (relative MAE) and positive transferability ratio can better characterize whether the negative transfer happens in local clients. (2) The proposed FEDORA framework achieves comparable average accuracy (MAE) but a much better positive transferability ratio than state-of-the-art baselines.

The number of training samples in one client might affect whether it can benefit from federated collaborations. As a result, we empirically investigate the impact of the number of training samples in personalized federated learning. Table 6.4 reports the image classification results on Rotated MNIST and Fashion-MNIST when one client (e.g., client 18 in Rotated MNIST) has much more training samples than others. It shows that when using the average model parameters (e.g., learned by FedAvg [105]) to regularize the local

Model	Rotated MNIST			Rotated Fashion-MNIST		
	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$
LOCAL	0.7736	-	-	0.7079	-	-
FedAvg	0.5961	-0.2310	0.1944	0.5156	-0.2697	0.0694
FedAvg+FT	0.7631	-0.0131	0.3333	0.6671	-0.0559	0.2083
FedProx	0.5564	-0.2826	0.1389	0.4529	-0.3590	0.0417
FedProx+FT	0.7111	-0.0805	0.1944	0.6387	-0.0955	0.1111
LG-FedAvg	0.7916	0.0240	0.8611	0.7187	0.0154	0.7500
FedPer	0.7676	-0.0071	0.4722	0.6599	-0.0654	0.1806
pFedHN	0.7927	0.0254	0.6667	0.7254	0.0275	0.7083
APFL	0.7934	0.0262	0.8889	0.7219	0.0204	0.8056
Ditto	0.7908	0.0231	0.5000	0.6738	-0.0464	0.2083
IFCA	0.8263	0.0693	0.8889	0.7356	0.0414	0.7500
FeSEM	0.7876	0.0183	0.5833	0.7171	0.0154	0.6389
FedFOMO	0.7959	0.0293	0.8611	0.7225	0.0213	0.7639
FedU	0.7928	0.0255	0.8333	0.7229	0.0226	0.7778
FedAMP	0.7906	0.0228	0.9167	0.7228	0.0217	0.8194
FEDORA	0.8366	0.0828	1.0000	0.7466	0.0562	0.9444

Table 6.4: Impact of imbalanced samples among clients

model, personalized federated learning approaches (e.g., Ditto [190], FedPer [184]) has lower positive transferability ratio. This is because the shared average model parameters would significantly bias toward the client with a large number of training samples. We visualize the relative performance improvement of local clients in Figure 6.3. It confirms that Ditto and FedPer can only achieve satisfactory performance on clients when they have similar distributions as client 18. In contrast, **FEDORA** can mitigate the negative transfer for all the clients.

We show in Subsection 6.1.3 that **FEDORA** can be adapted to new clients associated with either labeled or unlabeled training samples. Figure 6.4 provides the results on Rotated MNIST by adapting the federated learning models to new clients, where the classification accuracy on new clients is reported. We observe from the results that **FEDORA** achieves better prediction performance when the new client has labeled training samples for fine-tuning (Figure 6.4(a)). When the new client only has unlabeled training samples, most existing approaches [109, 194] fail to adapt the trained federated learning system to the new client. FedAvg [105] can simply share the global model with the new client, but it does not consider the data heterogeneity between the new client and the old ones. In contrast, **FEDORA** learns the personalized auxiliary parameters based on the client similarity. Figure 6.4(b) confirms the superior performance of **FEDORA** over FedAvg.

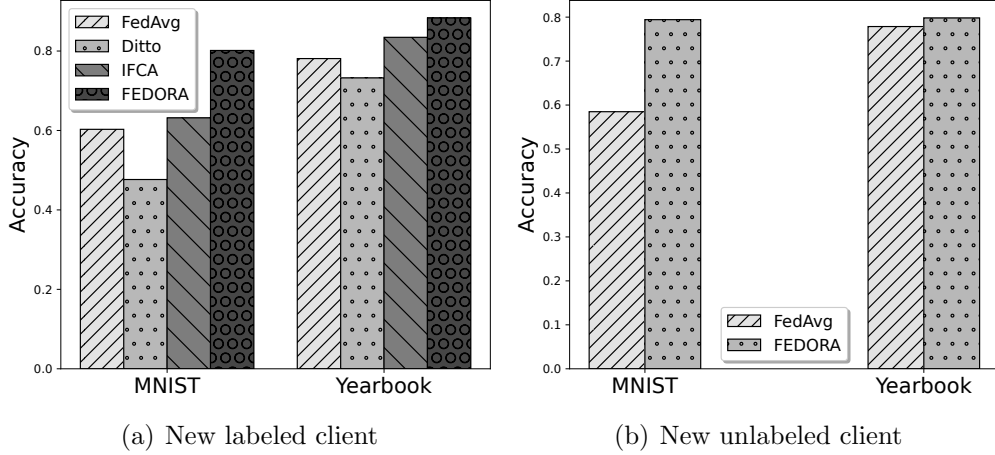


Figure 6.4: Generalization to the new client where the new client has (a) labeled training samples; (b) unlabeled training samples

Model	Acc $\uparrow$	R-Acc $\uparrow$	PTR $\uparrow$
FEDORA with random similarity	0.7657	0.0020	0.7222
FEDORA with parameter similarity	0.8003	0.0477	0.9444
FEDORA with gradient similarity	0.8125	0.0637	0.9722
FEDORA	0.8251	0.0806	1.0000

Table 6.5: Impact of client similarity measure on FEDORA

Ablation Study. We study the impact of client similarity measurements on the proposed FEDORA framework. More specifically, we consider several methods to estimate client similarity, including random client similarity [192], parameter similarity [193], and gradient similarity [108]. Table 6.5 shows the personalized federated learning results. It can be seen that when estimating the client similarity based on data distribution, FEDORA achieves better prediction performance and positive transferability ratio. The coordinate-wise parameter/-gradient similarity cannot accurately measure the client similarity due to the permutation invariance of neural network parameters [308].

6.2 HARD CLUSTERING FOR PRIVACY-PRESERVING KNOWLEDGE TRANSFER

In Section 6.1, we study personalized federated learning through the lens of transfer learning. The key idea is to adaptively transfer knowledge among clients based on the similarities in their data distributions. This thus can be viewed as a soft clustering method for identifying the collaboration structure among clients. Furthermore, in this section, we will explore clients’ collaboration structure from a hard clustering perspective [107]. Specifically, we can

find an optimal collaboration structure where each client only collaborates with clients in the same cluster for knowledge sharing.

6.2.1 Problem Definition

Following [108, 109], we formally define the problem of clustered federated learning.

Problem 6.1. Given a set of local clients with private training sets and a learning algorithm $f(\cdot)$, *clustered federated learning* aims to partition clients into non-overlapping coalitions such that each client only collaborates with clients in the same coalition for training models.

In clustered federated learning, clients in the same coalition share the same model parameters, while clients in different coalitions have different model parameters. To this end, a client k in coalition $\mathcal{S}$ trains the model with other clients in $\mathcal{S}$ to minimize the following weighted empirical prediction error.

$$\hat{h}_{\alpha_i} = \arg \min_{h \in \mathcal{H}} \sum_{k=1}^K \alpha_{kk'} \hat{\mathcal{E}}_{k'}(h) \quad (6.26)$$

where $\alpha_{kk'} = \frac{\beta_{k'} \mathbb{1}[k' \in \mathcal{S}]}{\sum_{k \in \mathcal{S}} \beta_k}$ and $\beta = [\beta_1, \dots, \beta_K] = [\frac{n_1}{\sum_{k=1}^K m_k}, \dots, \frac{n_K}{\sum_{k=1}^K m_k}]$. Notice that the objective function of Eq. (6.26) subsumes both local training and global federated learning. As discussed in Subsection 6.1, local training indicates that each client independently trains its own model without sharing knowledge with any other clients. In contrast, global federated learning [105] allows all the clients to collaborate in training a globally shared model. The objective function of Eq. (6.26) can recover local training by setting α_k as a one-hot vector (i.e., $\alpha_{kk} = 1$ and $\alpha_{kk'} = 0$ for $k' \neq k$) and global federated learning [105] by setting $\alpha_k = \beta$. This demonstrates that clustered federated learning lies between two extremes: no knowledge sharing and global knowledge sharing.

The key challenge of clustered federated learning is to find the optimal collaboration structure among clients. For example, given K local clients, there are at least 2^{K-1} different coalitions for local client k , it is prohibitively expensive to enumerate every option and pick the best model. Instead, we propose a principle framework based on the distribution discrepancy and the data quantity to optimize the collaboration structure in practice.

6.2.2 Theoretical Analysis

Theorem 6.1. Given a hypothesis space $\mathcal{H}$, we let $\hat{h}_{\alpha_k}$ be the empirical error minimizer defined in Eq. (6.26) and h_k^* be client k 's expected error minimizer. If we use VC dimension

d_{VC} to quantify the complexity of hypothesis space $\mathcal{H}$, for any $\delta \in (0, \frac{1}{2})$, with probability at least $1 - 2\delta$, the following holds

$$\begin{aligned} \mathcal{E}_k(\hat{h}_{\alpha_k}) &\leq \mathcal{E}_k(h_k^*) + 2\sqrt{\left(\sum_{k'=1}^K \frac{\alpha_{kk'}^2}{\beta_{k'}}\right) \left(\frac{2d_{\text{VC}} \log(2n+2) + \log(4/\delta)}{n}\right)} \\ &\quad + \sum_{k' \neq k} \alpha_{kk'} (d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_k, \mathbb{P}_{k'}) + 2\lambda_{kk'}^*) \end{aligned} \quad (6.27)$$

where $\mathcal{E}_k(h_k^*) = \min_{h \in \mathcal{H}} \mathcal{E}_k(h)$ is the expected prediction error on the client k , $n = \sum_{k=1}^K n_k$, $\lambda_{kk'}^* = \min_{h \in \mathcal{H}} \mathcal{E}_k(h) + \mathcal{E}_{k'}(h)$, and $d_{\mathcal{H}\Delta\mathcal{H}}(\cdot, \cdot)$ is the $\mathcal{H}\Delta\mathcal{H}$ -divergence [38].

Proof. It can be seen that this error bound generalizes Theorem 4 in [38] to clustered federated learning with multiple domains. QED.

Theorem 6.1 reveals that the optimal coalition for client k should consider both quantity information (β, m) and distribution difference $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_k, \mathbb{P}_{k'})$ to minimize the error $\mathcal{E}_k(\hat{h}_{\alpha_k})$. Furthermore, it is flexible to generalize the observation of Theorem 6.1 by considering various domain discrepancy measures, e.g., Margin Disparity Discrepancy [53], f -divergence [31], $\mathcal{C}$ -divergence [39, 256], etc.

6.2.3 Proposed Algorithm

We propose a novel approach **FEDCOLLAB** to optimize the collaboration structure. The key idea of **FEDCOLLAB** is to minimize the error bounds of all clients. Given a collaboration structure $\{\mathcal{S}_1, \dots, \mathcal{S}_C\}$ with C non-overlapping coalitions where C is the number of coalitions, clients from the same coalition have the same collaboration vector α_i as defined in Eq. (6.26) since they share the same global model. Here we define the collaboration matrix $\mathbf{A} = [\alpha_1^T, \dots, \alpha_K^T]^T$ as follows.

$$\mathbf{A}_{kk'} = \alpha_{kk'} = \begin{cases} \frac{\beta_{k'}}{\sum_{l \in \mathcal{S}_c} \beta_l}, & \text{if } k, k' \in \mathcal{S}_c, \exists c \\ 0, & \text{otherwise} \end{cases} \quad (6.28)$$

Then, the **FEDCOLLAB** objective can be formulated as

$$\mathcal{L} = \sum_{k=1}^K \left(\frac{\lambda}{\sqrt{n}} \sqrt{\sum_{k'=1}^K \frac{\alpha_{kk'}^2}{\beta_{k'}}} + \sum_{k'=1}^K \alpha_{kk'} d_{kk'} \right) \quad (6.29)$$

Algorithm 6.3: FEDCOLLAB optimizer

Input: $\beta, n, d_{kk'}$;

Output: Coalition assignment $p(\cdot)$.

Initialize $p(k) = k$ for all clients (local training);

while *not converged* **do**

foreach *client k in a permutation of $[1, \dots, K]$* **do**

 Evaluate the objective of Eq. (6.29) with the new collaboration structure
 after setting $p(k) = 1, \dots, K$;

 Update $p(k)$ to the coalition with lowest value of Eq. (6.29);

end

end

where $\lambda > 0$ is a hyper-parameter (e.g., $2\sqrt{2d_{\text{VC}} \log(2n+2) + \log(4/\delta)}$ in Theorem 6.1) and $d_{kk'}$ denotes the domain discrepancy between client k and client k' .

Due to the discrete nature of the hard clustering constraint in Eq. (6.28), it is challenging to optimize the objective function in Eq. (6.29) using gradient descent. Therefore, we propose an efficient algorithm to solve the problem in discrete space. We optimize the coalition assignment $p(\cdot)$ which maps the client index to a coalition index (e.g., $p(1) = 2$ means assigning client 1 to coalition 2). We initialize the coalition assignment with local training, i.e., $p(k) = k, \forall k \in \{1, 2, \dots, K\}$, and iteratively assign clients to a new coalition that can further minimize the FEDCOLLAB objective in Eq (6.29) (see Algorithm 6.3 for pseudo-code of the proposed optimizer).

The optimizer guarantees to converge to the local optimum, since the objective function has finite values and strictly decreases in each iteration. In practice, since greedy methods generally do not guarantee the global optimum, we re-run Algorithm 6.3 multiple times with different random seeds to further refine the collaboration structure. Different from most clustered federated learning algorithms [109, 187], where re-optimizing the collaboration structure requires re-training FL models and introduces large computation and communication costs, the collaboration optimization process of FEDCOLLAB is purely on the server and does not require training any ML model. As a result, our optimizer is very efficient and only takes a few seconds to run.

After solving the collaboration structure, FEDCOLLAB fixes the collaboration and trains federated learning models within each coalition separately. Notice that since the collaboration structure and the federated learning model are optimized independently, FEDCOLLAB can be seamlessly integrated with any federated learning algorithms in this stage.

6.2.4 Generalization to New Clients

An additional advantage of **FEDCOLLAB** is that unlike typical federated learning systems [188, 190], which are costly to accommodate new clients after training, the proposed **FEDCOLLAB** framework allows new clients to join without requiring re-clustering and re-training of all federated learning models. In particular, **FEDCOLLAB** assigns new clients to existing coalitions that minimize the objective in Eq. (6.29) by estimating the distribution distance between the new client and existing clients, thus requiring only one coalition to fine-tune or re-train the FL model for each new client.

6.2.5 Experiment

In the experiments, we evaluate the effectiveness of the proposed **FEDCOLLAB** framework in optimizing the collaboration structure among clients and further validate how this optimized collaboration structure mitigates negative transfer.

Data Sets. We evaluate our framework on three models and datasets: we train a 3-layer MLP for Fashion-MNIST[314], a 5-layer CNN for CIFAR-10 [315], and an ImageNet pre-trained ResNet-18 [69] for CIFAR-100 (with 20 coarse labels). We simulate three typical scenarios of non-IIDness [105] on three datasets respectively, to show that our algorithm can handle a wide range of non-IIDness. For all scenarios, we simulate 20 clients with four types.

- **Label shift** [330]. Each client has a different label distribution. Different from Dirichlet partition [331] where the distribution distance between any two clients has the same expectation, we create multiple levels of distribution distances. For example, client 0’s label distribution is most close to clients 1-4, less close to clients 5-9, and very distinct to clients 10-19.
- **Feature shift** [109]. Each client’s image is rotated for a given angle: $+25^\circ$, -25° , $+155^\circ$, -155° for clients 0-4, 5-9, 10-14, and 15-19, respectively. Multiple levels of distribution distances also exist in this scenario: client 0’s images have 0° angle difference from client 1-4, 50° from client 5-9, 130° from client 10-14, and 180° from client 15-19.
- **Concept shift** [108]. Each client’s label indices are permuted. Similar multiple levels of distribution distances are constructed: client 0 has all labels aligned with clients 1-4, 14 labels aligned with clients 5-9, and no label aligned with clients 10-19.

To simulate quantity shift while maintaining explainability, we let clients 0-9 be “large” clients with more data, and clients 10-19 be “small” clients with less data. As a result, large

Method	Label Shift (FashionMNIST)			Feature Shift (CIFAR-10)			Concept Shift (CIFAR-100)		
	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$
Local Training	86.05 (0.28)	-	-	38.65 (0.44)	-	-	29.82 (0.56)	-	-
FedAvg	46.64 (0.12)	46.00 (2.24)	41.03 (0.24)	44.31 (0.98)	86.00 (4.18)	4.62 (0.58)	26.62 (0.12)	50.00 (0.00)	11.54 (0.45)
+FEDCOLLAB	92.45 (0.07)	100.00 (0.00)	5.99 (0.41)	52.61 (0.60)	100.00 (0.00)	3.30 (0.63)	40.94 (0.22)	100.00 (0.00)	2.78 (0.30)
FedProx	46.70 (0.08)	45.00 (5.00)	41.09 (0.29)	44.45 (0.58)	87.00 (4.47)	4.74 (0.56)	26.78 (0.14)	50.00 (0.00)	11.66 (0.36)
+FEDCOLLAB	92.39 (0.15)	100.00 (0.00)	6.02 (0.37)	52.73 (0.64)	100.00 (0.00)	3.16 (0.61)	40.99 (0.17)	100.00 (0.00)	2.79 (0.34)
FedNova	75.92 (1.14)	45.00 (3.54)	12.38 (1.25)	46.98 (0.57)	99.00 (2.24)	3.42 (0.22)	26.46 (0.13)	50.00 (0.00)	10.57 (0.32)
+FEDCOLLAB	92.47 (0.13)	100.00 (0.00)	5.97 (0.39)	52.72 (0.57)	100.00 (0.00)	3.18 (0.63)	40.92 (0.36)	100.00 (0.00)	2.75 (0.43)
Finetune	67.32 (3.17)	48.00 (2.74)	22.97 (2.82)	44.17 (0.99)	82.00 (2.74)	5.14 (0.32)	33.30 (4.79)	50.00 (0.00)	13.95 (0.57)
+FEDCOLLAB	92.57 (0.15)	99.00 (2.24)	6.07 (0.30)	51.53 (0.61)	100.00 (0.00)	2.92 (0.46)	40.94 (2.36)	100.00 (0.00)	2.54 (0.30)
Per-FedAvg	51.13 (4.10)	49.00 (2.24)	37.35 (4.15)	43.78 (0.69)	83.00 (9.08)	4.74 (0.65)	27.39 (0.24)	50.00 (0.00)	12.24 (0.46)
+FEDCOLLAB	92.16 (0.25)	97.00 (6.71)	6.00 (0.25)	52.64 (0.45)	100.00 (0.00)	3.03 (0.30)	41.04 (0.26)	100.00 (0.00)	2.85 (0.49)
pFedMe	55.31 (3.45)	47.00 (4.47)	33.71 (3.11)	39.74 (0.85)	60.00 (12.25)	4.81 (0.74)	27.04 (0.39)	48.00 (2.74)	10.39 (0.47)
+FEDCOLLAB	92.18 (0.43)	99.00 (2.24)	6.40 (0.81)	47.20 (1.29)	97.00 (2.74)	3.02 (0.30)	37.47 (0.31)	100.00 (0.00)	3.04 (0.23)
Ditto	68.73 (1.40)	48.00 (2.74)	20.29 (2.06)	47.04 (0.30)	97.00 (2.74)	3.85 (0.35)	32.50 (0.40)	50.00 (0.00)	12.22 (0.36)
+FEDCOLLAB	92.55 (0.08)	99.00 (2.24)	6.11 (0.30)	50.97 (0.75)	99.00 (2.24)	3.38 (1.55)	40.33 (0.33)	100.00 (0.00)	2.16 (0.30)

Table 6.6: Results on various data sets and base federated learning algorithms, where we report the mean and standard deviation for each evaluation metric in percentage (%) after five runs.

clients are more picky and perform best when they only collaborate with the same type of client. However, small clients prefer larger coalitions.

Evaluation Metric. To comprehensively evaluate the FL algorithms, besides the accuracy score (Acc), we use *incentivized participation rate* (IPR) [181] to evaluate how many clients get accuracy gains compared to local training, and *reward standard deviation* (RSD) to evaluate the fairness of accuracy gains. Both metrics are defined with local model $\hat{h}_i^{\text{LOCAL}}$ and FL model $\hat{h}_i^{\text{FL}}$.

$$\text{IPR} = \frac{1}{K} \sum_{k=1}^K \mathbb{1} \left[\text{Acc}(\hat{h}_k^{\text{FL}}) - \text{Acc}(\hat{h}_k^{\text{LOCAL}}) > 0 \right] \quad (6.30)$$

$$\text{RSD} = \text{SD}(\{\text{Acc}(\hat{h}_i^{\text{FL}}) - \text{Acc}(\hat{h}_i^{\text{LOCAL}})\}_{i=1}^N) \quad (6.31)$$

where $\mathbb{1}[\cdot]$ is an identity function and SD represents the standard deviation. For performance fairness, all clients should obtain similar accuracy gains indicated by a large IPR and small RSD.

Main Results. We demonstrate that although federated learning algorithms experience negative transfer, this issue can be alleviated after integration with FEDCOLLAB. To this end, we study a wide range of state-of-the-art federated learning algorithms. For global federated learning, in addition to FedAvg [105], we compare to FedProx [179] and FedNova [332].

Method	Label Shift (FashionMNIST)			Feature Shift (CIFAR-10)			Concept Shift (CIFAR-100)		
	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$
IFCA	91.49 (0.61)	95.00 (5.00)	5.62 (0.54)	49.78 (1.01)	100.00 (0.00)	3.13 (0.52)	30.74 (4.46)	60.00 (22.36)	11.28 (5.04)
FedCluster	92.07 (0.47)	95.00 (7.07)	6.14 (0.49)	44.86 (1.90)	79.00 (17.10)	5.64 (1.81)	29.23 (2.18)	62.00 (12.55)	9.55 (0.69)
FeSEM	56.79 (6.71)	45.00 (11.18)	36.12 (2.08)	42.73 (0.37)	82.00 (5.70)	4.10 (0.62)	31.92 (3.12)	72.00 (12.55)	9.81 (1.77)
KMeans	69.30 (0.81)	72.00 (2.74)	35.87 (1.22)	48.61 (1.15)	96.00 (4.18)	4.54 (0.74)	34.24 (3.01)	85.00 (13.69)	6.47 (3.06)
FEDCOLLAB	92.45 (0.07)	100.00 (0.00)	5.99 (0.41)	52.61 (0.60)	100.00 (0.00)	3.30 (0.63)	40.94 (0.22)	100.00 (0.00)	2.78 (0.30)

Table 6.7: Comparison with Clustered FL

Method	Acc $\uparrow$	IPR $\uparrow$	RSD $\downarrow$
FEDCOLLAB	92.45 (0.07)	100.00 (0.00)	5.99 (0.41)
w/o quantity term	90.31 (0.11)	94.00 (5.48)	4.30 (0.46)
w/o discrepancy term	67.79 (0.89)	19.00 (4.18)	18.97 (0.83)

Table 6.8: Ablation study on FashionMNIST with label shift

For personalized federated learning, we include Finetune (where each client locally finetunes the FedAvg model), a meta-learning-based method Per-FedAvg [189], a regularization-based method pFedMe [191], and a fair-and-robust method Ditto [190]. As shown in Table 6.6, the proposed FEDCOLLAB framework enhances the performance of all seven base federated learning algorithms in terms of accuracy, IPR, and fairness (RSD). In particular, in scenarios involving label shift and concept shift, all base federated learning algorithms suffer from negative transfer. Although personalized federated learning algorithms introduce accuracy gain compared to FedAvg, they do not solve the negative transfer problem since small clients still do not benefit from federated collaboration. However, when combined with our FEDCOLLAB, all base federated learning algorithms can reach a near 100% IPR with much better accuracy and performance fairness.

In addition, we compare FEDCOLLAB (combined with FedAvg) to baseline clustered federated learning algorithms, including a loss-based algorithm IFCA [109], a gradient-based algorithm FedCluster [108], and two parameter-based algorithms FeSEM [187] and KMeans [333]. We report the results in Table 6.7. Across all scenarios, FEDCOLLAB has the highest accuracy and IPR, with RSD among the lowest. This demonstrates the effectiveness of the proposed FEDCOLLAB framework in optimizing the collaboration structure.

Ablation Study we investigate the impact of distribution discrepancy and quantity on the optimization of collaboration structure. To this end, we consider two variants of FEDCOLLAB. “FEDCOLLAB w/o quantities” replaces the real quantity distribution β with a uniform vector $\frac{1}{K}\mathbf{1}$, while “FEDCOLLAB w/o discrepancy” considers the distribution distance $d_{kk'}$ between clients as a constant. Table 6.8 summarizes the results on FashionMNIST with label shift.

When ignoring distribution discrepancy, we observe that FEDCOLLAB assigns clients with no overlapping labels to the same coalition, which results in worse performance. When ignoring the data quantity, we observe that FEDCOLLAB forms multiple coalitions for small clients, instead of a large coalition for clients 10-19. Therefore, small clients get smaller performance gains compared to the original FEDCOLLAB.

CHAPTER 7: CONCLUSION AND FUTURE RESEARCH

In this chapter, we conclude this thesis and discuss future research directions in trustworthy transfer learning.

7.1 CONCLUSION

This thesis dives into understanding deep transfer learning from the perspectives of knowledge transferability and trustworthiness. Concisely, it explores the following research problems: *knowledge transferability* which focuses on quantitatively measuring and enhancing the transfer of knowledge or information between source and target domains, and *knowledge trustworthiness* which investigates the reliability and trustworthiness of the transferred knowledge in the context of deep transfer learning.

To address the first problem, we provide a suite of theoretical tools and algorithms to understand knowledge transferability in both IID and non-IID settings.

IID Transferability. It assumes that the samples within each domain are independent and identically drawn (IID). Based on the IID assumption in each domain at any time stamp, we study both static transfer learning scenarios where the source and target domains have static data distributions, and dynamic transfer learning scenarios where the data distribution of the source or target domain is continuously evolving over time. To be specific, in static transfer learning scenarios, we present a distribution-informed neural network (**DINO**) [18] for single-source knowledge transfer and a distributional network of networks (**TENON**) [13] for multi-source knowledge transfer. Furthermore, in dynamic transfer learning scenarios, we introduce a meta-learning framework (**L2E**) [8] for dynamic knowledge transfer with closed-set targets, and a positive-unlabeled learning framework (**OuterAdapter**) [10] for dynamic knowledge transfer with open-set targets.

Non-IID Transferability. It considers a relaxed assumption that the samples within each domain are not independent and identically drawn (non-IID). In this thesis, we study graph transfer learning, which involves the transfer of knowledge or information from a source graph to a relevant target graph. In addition to individual node attributes or features, the graph structure can affect the knowledge transferability across graphs. We start by studying the graph transfer learning scenario where the source and target graphs follow the homophilic structure assumption. In this scenario, we present a graph adaptive network (**GRADE**) by quantitatively measuring the distribution shifts based on the subtree representations. Furthermore, we study a more challenging yet realistic scenario where the source and target

graphs can follow different structure assumptions. A graph-structured Gaussian process model (**GraphGP**) [14] is then introduced by adaptively measuring the graph similarity based on the infinitely wide graph neural network induced graph kernel.

To address the second problem, we study the impact of adversarial robustness, privacy, and fairness on transfer learning frameworks.

Adversarial Robustness. We investigate attack and defense mechanisms to understand the adversarial robustness of knowledge transfer. To demonstrate the adversarial vulnerability of existing transfer learning algorithms, we present a generic indirect invisible poisoning framework **I2Attack** [44] for generating the poisoned source data such that existing algorithms could be easily fooled when predicting the target samples. Furthermore, we study the adversarial robustness of knowledge transfer in federated learning systems. To this end, we present an adversarially robust federated learning framework (**Fed.BVA**) [104] by building the connection between bias-variance decomposition and adversarial training.

Privacy & Fairness. We also explore data privacy and performance fairness in the context of transfer learning. Specifically, we formulate personalized federated learning as a group of privacy-preserving transfer learning problems. In this case, each private client associated with a specific domain aims to leverage the knowledge from other clients to improve the model performance. However, we show [45, 107] that some clients might suffer from negative transfer when participating in federated collaboration using existing personalized federated learning algorithms. To alleviate negative transfer, we present a soft clustering approach (**FEDORA**) [45] for adaptive knowledge transfer across clients, and a hard clustering approach (**FEDCOLLAB**) [107] for optimizing the collaboration structure among clients.

7.2 VISION FOR THE FUTURE

The philosophy underlying trustworthy transfer learning is to mimic human intelligence by leveraging the previously acquired knowledge to learn new concepts and tasks in a trustworthy manner. My short-term research aims to understand the trade-off between knowledge transferability and trustworthiness. In the long term, I am interested in building a computational foundation for trustworthy artificial intelligence with both theoretical insights and practical algorithms.

7.2.1 Short-Term Plans

Unified Knowledge Transferability. In my previous work, I have studied the knowledge transferability in IID and non-IID learning tasks [13, 14, 18, 24]. This prompts a natural

question: *Is it feasible to handle both IID and non-IID data within a unified transfer learning framework?* Furthermore, this question can be decomposed into several sub-questions. How can a unified transfer learning framework be designed when the source and target domains have different data modalities? When can the positive transfer across domains be theoretically guaranteed? What knowledge is transferred in noisy, dynamic, and open-world environments? With these questions in mind, I would like to develop transferable Gaussian processes that unify the sample generation process of IID and non-IID data from heterogeneous domains. It is crucial to design the transfer kernel to accurately capture the relationships between samples in noisy, dynamic, and open-world environments.

Transparency under Distribution Shifts. The adversarial robustness, privacy, and fairness have been studied in my previous work [44, 45, 294], in order to improve the security, privacy, and ethical aspects of transfer learning approaches. Going one step further, my research will develop a theoretical understanding of transparency in the context of deep transfer learning. Transparency aims at revealing how knowledge is transferred from the source domain to the target domain, thereby providing insights into the decision-making process within the target domain. Specifically, it is feasible to investigate the transparency of knowledge transfer from two aspects. One is the counterfactual explanation of the transferred knowledge in terms of both distribution-oriented and label-oriented feature factors. It detects these factors for explaining prediction results in the target domain. The other one is the uncertainty associated with model predictions. The quantification of prediction uncertainty is closely correlated with the distribution discrepancy across domains. The counterfactual explanation and prediction uncertainty provide complementary information to enhance model transparency.

Inherent Trade-off between Transferability and Trustworthiness. It is shown that trustworthiness (e.g., adversarial robustness) may be at odds with transfer accuracy. This raises a subsequent question: *what is the fundamental trade-off between knowledge transferability and trustworthiness?* To answer this question, I plan to investigate the impact of trustworthiness properties on the generalization error bounds of transfer learning. For example, the existence of such trade-offs can be verified by a lower error bound in terms of trustworthiness properties. It is worth noting that answering this question becomes more challenging when multiple trustworthiness properties are involved. The interconnections of different trustworthiness properties complicate the assessment of the trustworthiness of any hypothesis. Therefore, I will start by deriving the potential trade-off between knowledge transferability and individual trustworthiness property (e.g., adversarial robustness, fairness, etc.). Then, the impact of various trustworthiness properties can be disentangled in the context of deep transfer learning. In this case, the disentangled trustworthiness properties (e.g.,

unfairness arising from protected attributes and adversarial vulnerability induced by noisy class-discriminative regions) are closely associated with the residual information captured between the input sample and its corresponding domain-invariant representation.

7.2.2 Long-Term Plans

Physics-Informed Transfer Learning. The generalization performance of a machine learning algorithm is determined by the quality and quantity of training samples. Recently, data-centric transfer learning has achieved remarkable success in computer vision and neutral language processing communities. This success can be attributed to the utilization of numerous high-quality samples collected from diverse sources. Nevertheless, this could potentially restrict the applicability of transfer learning techniques in real-world situations, where collecting high-quality source samples is often costly and time-consuming. Alternatively, one can inject physical rules or simulations as the source knowledge to guide the knowledge transferability in the target domain. This physics-informed transfer learning paradigm extends beyond the traditional data-centric transfer learning that explicitly relies on a large number of high-quality source samples. Moreover, the source knowledge can be provided from multiple facets, including both labeled samples and physical rules. Therefore, one of my long-term plans is to generalize the theoretical understanding and empirical algorithms of transfer learning from data-centric to physics-informed (or multi-faceted) scenarios.

Computational Foundation of Trustworthy Artificial Intelligence. Trustworthy Artificial Intelligence (AI) has gained substantial attention for building reliable, ethical, transparent, fair, and accountable systems. From the perspective of trustworthy transfer learning, customized AI systems can be developed by transferring the previously acquired knowledge to learn user-specific tasks. On top of trustworthy transfer learning, I will further investigate the universal knowledge (e.g., domain-invariant representation shared by both seen and unseen domains) and trustworthiness (e.g., commonsense trustworthy properties) of trustworthy AI. Concisely, universal knowledge characterizes commonsense knowledge that can be readily applied to new domains without the need for re-training. This universal knowledge can be discovered by domain-invariant representation learning and then applied to out-of-distribution generalization. Theoretical analysis of universal knowledge identifies the conditions under which a learning algorithm can generalize well. On the other hand, universal trustworthiness guarantees the reliability of a learning algorithm in new unseen domains. In general, the computational foundation of trustworthy AI can provide valuable insights into improving the effectiveness and trustworthiness of AI systems in learning novel concepts and tasks.

REFERENCES

- [1] J. McCarthy, “What is artificial intelligence?” 2004.
- [2] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [4] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 3320–3328.
- [5] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III*, vol. 11141. Springer, 2018, pp. 270–279.
- [6] B. Neyshabur, H. Sedghi, and C. Zhang, “What is being transferred in transfer learning?” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [7] N. Tripuraneni, M. I. Jordan, and C. Jin, “On the theory of transfer learning: The importance of task diversity,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [8] J. Wu and J. He, “A unified meta-learning framework for dynamic transfer learning,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2022, pp. 3573–3579.
- [9] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, vol. 37, 2015, pp. 97–105.
- [10] J. Wu and J. He, “Domain adaptation with dynamic open-set targets,” in *KDD ’22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. ACM, 2022, pp. 2039–2049.
- [11] J. He, Y. Liu, and R. D. Lawrence, “Graph-based transfer learning,” in *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. ACM, 2009, pp. 937–946.

- [12] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee, E. David, I. Stavness, W. Guo, B. Earnshaw, I. S. Haque, S. M. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang, “WILDS: A benchmark of in-the-wild distribution shifts,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 5637–5664.
- [13] J. Wu, J. He, and H. Tong, “Distributional network of networks for modeling data heterogeneity,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.
- [14] J. Wu, L. Ainsworth, A. Leakey, H. Wang, and J. He, “Graph-structured gaussian processes for transferable graph learning,” in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [15] M. Liu, Z. Fang, Z. Zhang, M. Gu, S. Zhou, X. Wang, and J. Bu, “Rethinking propagation for unsupervised graph domain adaptation,” in *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, 2024, pp. 13 963–13 971.
- [16] S. Wang, K. Guan, C. Zhang, C. Jiang, Q. Zhou, K. Li, Z. Qin, E. Ainsworth, J. He, J. Wu, D. Schaefer, L. Gentry, A. Margenot, and L. Herzberger, “Airborne hyperspectral imaging of cover crops through radiative transfer process-guided machine learning,” *Remote Sensing of Environment*, vol. 285, p. 113386, 02 2023.
- [17] Y. Ma, S. Chen, S. Ermon, and D. Lobell, “Transfer learning in environmental remote sensing,” *Remote Sensing of Environment*, vol. 301, p. 113924, 02 2024.
- [18] J. Wu, J. He, S. Wang, K. Guan, and E. A. Ainsworth, “Distribution-informed neural networks for domain adaptation regression,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [19] B. Mieth, J. Hockley, N. Görnitz, M. Höhne, K.-R. Müller, A. Gutteridge, and D. Ziemek, “Using transfer learning from prior reference knowledge to improve the clustering of single-cell rna-seq data,” *Scientific Reports*, vol. 9, p. 20353, 12 2019.
- [20] M. Lotfollahi, M. Naghipourfar, M. D. Luecken, M. Khajavi, M. Büttner, M. Wagenvoetter, Žiga Avsec, A. Gayoso, N. Yosef, M. Interlandi, S. Rybakov, A. V. Misharin, and F. J. Theis, “Mapping single-cell data to reference atlases by transfer learning,” *Nature Biotechnology*, vol. 40, pp. 121 – 130, 2021.

- [21] C. Park, A. Awadalla, T. Kohno, and S. N. Patel, “Reliable and trustworthy machine learning for health using dataset shift detection,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 3043–3056.
- [22] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi, “Cross-domain recommender systems,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Springer, 2015, pp. 919–959.
- [23] G. Hu, Y. Zhang, and Q. Yang, “Conet: Collaborative cross networks for cross-domain recommendation,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 2018, pp. 667–676.
- [24] J. Wu, J. He, and E. A. Ainsworth, “Non-iid transfer learning on graphs,” in *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 2023, pp. 10 342–10 350.
- [25] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, “To transfer or not to transfer,” in *NIPS 2005 workshop on transfer learning*, vol. 898, no. 3, 2005.
- [26] Z. Wang, Z. Dai, B. Póczos, and J. G. Carbonell, “Characterizing and avoiding negative transfer,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 11 293–11 302.
- [27] I. Kuzborskij and F. Orabona, “Stability and hypothesis transfer learning,” in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, vol. 28, 2013, pp. 942–950.
- [28] S. S. Du, J. Koushik, A. Singh, and B. Póczos, “Hypothesis transfer learning via transformation functions,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 574–584.
- [29] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, “Analysis of representations for domain adaptation,” in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, 2006, pp. 137–144.
- [30] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman, “Learning bounds for domain adaptation,” in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, 2007, pp. 129–136.

- [31] D. Acuna, G. Zhang, M. T. Law, and S. Fidler, “ f -domain adversarial learning: Theory and algorithms,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 66–75.
- [32] G. Blanchard, G. Lee, and C. Scott, “Generalizing from several related classification tasks to a new unlabeled sample,” in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 2011, pp. 2178–2186.
- [33] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, “Undoing the damage of dataset bias,” in *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 7572. Springer, 2012, pp. 158–171.
- [34] K. Muandet, D. Balduzzi, and B. Schölkopf, “Domain generalization via invariant feature representation,” in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, vol. 28, 2013, pp. 10–18.
- [35] I. Gulrajani and D. Lopez-Paz, “In search of lost domain generalization,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.
- [36] G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *CoRR*, vol. abs/1503.02531, 2015.
- [37] M. Phuong and C. Lampert, “Towards understanding knowledge distillation,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 5142–5151.
- [38] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, “A theory of learning from different domains,” *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, 2010.
- [39] J. Wu, H. Tong, E. A. Ainsworth, and J. He, “Adaptive knowledge transfer on evolving domains,” in *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*. IEEE, 2022, pp. 1389–1394.
- [40] A. Kumar, T. Ma, and P. Liang, “Understanding self-training for gradual domain adaptation,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 5468–5479.
- [41] L. Ruiz, L. F. O. Chamon, and A. Ribeiro, “Graphon neural networks and the transferability of graph neural networks,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

- [42] P. P. Busto and J. Gall, “Open set domain adaptation,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 754–763.
- [43] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada, “Open set domain adaptation by backpropagation,” in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, vol. 11209. Springer, 2018, pp. 156–171.
- [44] J. Wu and J. He, “Indirect invisible poisoning attacks on domain adaptation,” in *KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2021, pp. 1852–1862.
- [45] J. Wu, W. Bao, E. A. Ainsworth, and J. He, “Personalized federated learning with parameter propagation,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*. ACM, 2023, pp. 2594–2605.
- [46] B. An, Z. Che, M. Ding, and F. Huang, “Transferring fairness under distribution shifts via fair consistency regularization,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [47] J. Wu and J. He, “Trustworthy transfer learning: Transferability and trustworthiness,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*. ACM, 2023, pp. 5829–5830.
- [48] C. R. Yendrek, T. Tomaz, C. M. Montes, Y. Cao, A. M. Morse, P. J. Brown, L. M. McIntyre, A. D. Leakey, and E. A. Ainsworth, “High-Throughput Phenotyping of Maize Leaf Physiological and Biochemical Traits Using Hyperspectral Reflectance,” *Plant Physiology*, vol. 173, no. 1, pp. 614–626, 2016.
- [49] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, “Moment matching for multi-source domain adaptation,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1406–1415.
- [50] H. Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.
- [51] Z. C. Lipton, Y. Wang, and A. J. Smola, “Detecting and correcting for label shift with black box predictors,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 3128–3136.

- [52] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, “A unifying view on dataset shift in classification,” *Pattern recognition*, vol. 45, no. 1, pp. 521–530, 2012.
- [53] Y. Zhang, T. Liu, M. Long, and M. I. Jordan, “Bridging theory and algorithm for domain adaptation,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 7404–7413.
- [54] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, “Transfer learning of graph neural networks with ego-graph information maximization,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 1766–1779.
- [55] S. Liu, T. Li, Y. Feng, N. Tran, H. Zhao, Q. Qiu, and P. Li, “Structural re-weighting improves graph domain adaptation,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 21 778–21 793.
- [56] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 1024–1034.
- [57] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.
- [58] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70. PMLR, 2017, pp. 1263–1272.
- [59] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, “Beyond homophily in graph neural networks: Current limitations and effective designs,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [60] B. Eshete, “Making machine learning trustworthy,” *Science*, vol. 373, no. 6556, pp. 743–744, 2021.
- [61] K. R. Varshney, *Trustworthy Machine Learning*. Chappaqua, NY, USA: Independently Published, 2022.
- [62] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi, “Trustworthy artificial intelligence: A review,” *ACM Computing Surveys*, vol. 55, no. 2, pp. 39:1–39:38, 2023.

- [63] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, “Trustworthy graph neural networks: Aspects, methods, and trends,” *Proceedings of the IEEE*, vol. 112, no. 2, pp. 97–139, 2024.
- [64] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [65] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- [66] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, 2009*, pp. 248–255.
- [67] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.
- [68] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, H. Alobeidli, A. Cappelli, B. Pannier, E. Almazrouei, and J. Launay, “The refinedweb dataset for falcon LLM: outperforming curated corpora with web data only,” in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [69] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778.
- [70] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.
- [71] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

- [72] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Nee-lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [73] I. Dayan, H. Roth, A. Zhong, A. Harouni, A. Gentili, A. Abidin, A. Liu, A. Costa, B. Wood, C.-S. Tsai, C.-H. Wang, C.-N. Hsu, C. Lee, P. Ruan, D. Xu, D. Wu, E. Huang, F. Kitamura, G. Lacey, and Q. Li, “Federated learning for predicting clinical outcomes in patients with covid-19,” *Nature Medicine*, vol. 27, pp. 1–9, 10 2021.
- [74] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” *CoRR*, vol. abs/1811.03604, 2018.
- [75] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel, “Fairness through awareness,” in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*. ACM, 2012, pp. 214–226.
- [76] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, “Certifying and removing disparate impact,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 259–268.
- [77] M. Hardt, E. Price, and N. Srebro, “Equality of opportunity in supervised learning,” in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, 2016*, pp. 3315–3323.
- [78] D. Pessach and E. Shmueli, “Algorithmic fairness,” in *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*. Springer, 2023, pp. 867–886.
- [79] S. Caton and C. Haas, “Fairness in machine learning: A survey,” *ACM Computing Surveys*, vol. 56, no. 7, pp. 166:1–166:38, 2024.
- [80] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva, “Counterfactual fairness,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017*, pp. 4066–4076.
- [81] T. Li, M. Sanjabi, A. Beirami, and V. Smith, “Fair resource allocation in federated learning,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

- [82] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70. PMLR, 2017, pp. 1885–1894.
- [83] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. V. Dillon, J. Ren, and Z. Nado, “Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 13969–13980.
- [84] J. Wu, E. A. Ainsworth, S. Wang, K. Guan, and J. He, “Adaptive Transfer Learning for Plant Phenotyping,” in *Third International Workshop on Machine Learning for Cyber-Agricultural Systems*, 2021.
- [85] Y. Zhou, J. Xu, J. Wu, Z. T. Nasrabadi, E. Körpeoglu, K. Achan, and J. He, “PURE: positive-unlabeled recommendation with generative adversarial network,” in *KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2021, pp. 2409–2419.
- [86] S. Roach, C. Ni, A. Kopylov, T. Lu, J. Xu, S. Zhang, B. Du, D. Zhou, J. Wu, L. Liu, Y. Yan, J. He, and H. Tong, “CANON: complex analytics of network of networks for modeling adversarial activities,” in *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*. IEEE, 2020, pp. 1634–1643.
- [87] J. Wu and J. He, “Dynamic transfer learning with progressive meta-task scheduler,” *Frontiers Big Data*, vol. 5, 2022.
- [88] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, “Domain-adversarial training of neural networks,” *Journal of Machine Learning Research*, vol. 17, pp. 59:1–59:35, 2016.
- [89] X. Chen, S. Wang, J. Wang, and M. Long, “Representation subspace distance for domain adaptation regression,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 1749–1759.
- [90] B. Cao, S. J. Pan, Y. Zhang, D. Yeung, and Q. Yang, “Adaptive transfer learning,” in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010, pp. 407–412.
- [91] A. Jacot, C. Hongler, and F. Gabriel, “Neural tangent kernel: Convergence and generalization in neural networks,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 8580–8589.

- [92] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 855–864.
- [93] J. Wu, J. He, and Y. Liu, “Imverde: Vertex-diminished random walk for learning imbalanced network representation,” in *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*. IEEE, 2018, pp. 871–880.
- [94] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2013, pp. 3111–3119.
- [95] R. Wang, Y. Dong, S. O. Arik, and R. Yu, “Koopman neural operator forecaster for time-series with temporal distributional shifts,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [96] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [97] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 3837–3845.
- [98] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [99] J. Wu, J. He, and J. Xu, “DEMO-Net: degree-specific graph neural networks for node and graph classification,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 406–415.
- [100] A. Leman and B. Weisfeiler, “A reduction of a graph to a canonical form and an algebra arising during this reduction,” *Nauchno-Tekhnicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.
- [101] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-Lehman graph kernels,” *Journal of Machine Learning Research*, vol. 12, pp. 2539–2561, 2011.

- [102] S. Ben-David, T. Lu, T. Luu, and D. Pál, “Impossibility theorems for domain adaptation,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, vol. 9, 2010, pp. 129–136.
- [103] H. Zhao, R. T. des Combes, K. Zhang, and G. J. Gordon, “On learning invariant representations for domain adaptation,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 7523–7532.
- [104] Y. Zhou, J. Wu, H. Wang, and J. He, “Adversarial robustness through bias variance decomposition: A new perspective for federated learning,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*. ACM, 2022, pp. 2753–2762.
- [105] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, vol. 54. PMLR, 2017, pp. 1273–1282.
- [106] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, “Advances and open problems in federated learning,” *Foundations and Trends® in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.
- [107] W. Bao, H. Wang, J. Wu, and J. He, “Optimizing the collaboration structure in cross-silo federated learning,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 1718–1736.
- [108] F. Sattler, K. Müller, and W. Samek, “Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.
- [109] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, “An efficient framework for clustered federated learning,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [110] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in domain adaptation theory*. Elsevier, 2019.

- [111] Y. Mansour, M. Mohri, and A. Rostamizadeh, “Domain adaptation: Learning bounds and algorithms,” in *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.
- [112] J. Wen, R. Greiner, and D. Schuurmans, “Domain aggregation networks for multi-source domain adaptation,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 10 214–10 224.
- [113] C. Cortes, M. Mohri, and A. M. Medina, “Adaptation algorithm and theory based on generalized discrepancy,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 169–178.
- [114] C. Cortes, M. Mohri, and A. Muñoz Medina, “Adaptation based on generalized discrepancy,” *Journal of Machine Learning Research*, vol. 20, pp. 1:1–1:30, 2019.
- [115] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.
- [116] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *CoRR*, vol. abs/1412.3474, 2014.
- [117] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, “Joint distribution optimal transportation for domain adaptation,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 3730–3739.
- [118] J. Shen, Y. Qu, W. Zhang, and Y. Yu, “Wasserstein distance guided representation learning for domain adaptation,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018, pp. 4058–4065.
- [119] A. T. Nguyen, T. Tran, Y. Gal, P. H. S. Torr, and A. G. Baydin, “KL guided domain adaptation,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.
- [120] W. Yin, S. Yu, Y. Lin, J. Liu, J.-J. Sonke, and E. Gavves, “Domain adaptation with Cauchy-Schwarz divergence,” in *Uncertainty in Artificial Intelligence*, 2024.
- [121] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 2962–2971.

- [122] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “CyCADA: cycle-consistent adversarial domain adaptation,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 1994–2003.
- [123] M. Long, Z. Cao, J. Wang, and M. I. Jordan, “Conditional adversarial domain adaptation,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018*, pp. 1647–1657.
- [124] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon, “Adversarial multiple source domain adaptation,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018*, pp. 8568–8579.
- [125] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014*, pp. 2672–2680.
- [126] A. Maurer, M. Pontil, and B. Romera-Paredes, “The benefit of multitask representation learning,” *Journal of Machine Learning Research*, vol. 17, pp. 81:1–81:32, 2016.
- [127] A. Watkins, E. Ullah, T. Nguyen-Tang, and R. Arora, “Optimistic rates for multi-task representation learning,” in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.
- [128] N. Tripuraneni, C. Jin, and M. I. Jordan, “Provable meta-learning of linear representations,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 10 434–10 443.
- [129] S. S. Du, W. Hu, S. M. Kakade, J. D. Lee, and Q. Lei, “Few-shot learning via learning the representation, provably,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.
- [130] Y. Bengio, A. C. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [131] Y. Bao, Y. Li, S. Huang, L. Zhang, L. Zheng, A. Zamir, and L. J. Guibas, “An information-theoretic approach to transferability in task transfer learning,” in *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*. IEEE, 2019, pp. 2309–2313.

- [132] A. T. Tran, C. V. Nguyen, and T. Hassner, “Transferability and hardness of supervised classification tasks,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1395–1405.
- [133] Y. Li, X. Jia, R. Sang, Y. Zhu, B. Green, L. Wang, and B. Gong, “Ranking neural checkpoints,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, 2021, pp. 2663–2673.
- [134] A. Agostinelli, J. R. R. Uijlings, T. Mensink, and V. Ferrari, “Transferability metrics for selecting source model ensembles,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 7926–7936.
- [135] M. Pándy, A. Agostinelli, J. R. R. Uijlings, V. Ferrari, and T. Mensink, “Transferability estimation using bhattacharyya class separability,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 2022, pp. 9162–9172.
- [136] C. V. Nguyen, T. Hassner, M. W. Seeger, and C. Archambeau, “LEEP: A new measure to evaluate transferability of learned representations,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 7294–7305.
- [137] K. You, Y. Liu, J. Wang, and M. Long, “LogME: practical assessment of pre-trained models for transfer learning,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 12 133–12 143.
- [138] L. Huang, J. Huang, Y. Rong, Q. Yang, and Y. Wei, “Frustratingly easy transferability estimation,” in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 9201–9225.
- [139] J. Wu and J. He, “Scalable manifold-regularized attributed network embedding via maximum mean discrepancy,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2019, pp. 2101–2104.
- [140] J. Y. Lee, F. Deroncourt, and P. Szolovits, “Transfer learning for named-entity recognition with neural networks,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018.
- [141] V. Kuznetsov and M. Mohri, “Discrepancy-based theory and algorithms for forecasting non-stationary time series,” *Annals of Mathematics and Artificial Intelligence*, vol. 88, no. 4, pp. 367–399, 2020.

- [142] S. Krishnagopal and L. Ruiz, “Graph neural tangent kernel: Convergence on large graphs,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 17 827–17 841.
- [143] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, “Unsupervised domain adaptive graph convolutional networks,” in *WWW ’20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 2020, pp. 1457–1467.
- [144] Q. Dai, X. Wu, J. Xiao, X. Shen, and D. Wang, “Graph transfer learning via adversarial domain adaptation with graph convolution,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4908–4922, 2023.
- [145] Y. You, T. Chen, Z. Wang, and Y. Shen, “Graph domain adaptation via theory-grounded spectral regularization,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [146] S. Liu, T. Li, Y. Feng, N. Tran, H. Zhao, Q. Qiu, and P. Li, “Structural re-weighting improves graph domain adaptation,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 21 778–21 793.
- [147] B. Zoph and K. Knight, “Multi-source neural translation,” in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. The Association for Computational Linguistics, 2016, pp. 30–34.
- [148] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, “Google’s multilingual neural machine translation system: Enabling zero-shot translation,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.
- [149] M. Artetxe, G. Labaka, E. Agirre, and K. Cho, “Unsupervised neural machine translation,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [150] H. Zhao, J. Hu, and A. Risteski, “On learning language-invariant representations for universal machine translation,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 11 352–11 364.

- [151] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, “Llama 2: Open foundation and fine-tuned chat models,” *CoRR*, vol. abs/2307.09288, 2023.
- [152] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, O. Abend, R. Alon, T. Asida, A. Bergman, R. Glozman, M. Gokhman, A. Manevich, N. Ratner, N. Rozen, E. Shwartz, M. Zusman, and Y. Shoham, “Jamba: A hybrid transformer-mamba language model,” *CoRR*, vol. abs/2403.19887, 2024.
- [153] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for NLP,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 2790–2799.
- [154] X. Jin, Y. Park, D. C. Maddix, H. Wang, and Y. Wang, “Domain adaptation for time series forecasting via attention sharing,” in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 10 280–10 297.
- [155] S. Purushotham, W. Carvalho, T. Nilanon, and Y. Liu, “Variational recurrent adversarial deep domain adaptation,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- [156] Y. Özyurt, S. Feuerriegel, and C. Zhang, “Contrastive learning for unsupervised domain adaptation of time series,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [157] H. He, O. Queen, T. Koker, C. Cuevas, T. Tsiligkaridis, and M. Zitnik, “Domain adaptation for time series under feature and label shifts,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, vol. 202. PMLR, 2023, pp. 12 746–12 774.
- [158] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 5998–6008.

- [159] D. Fernández Llorca and E. Gómez, “Trustworthy autonomous vehicles,” *Publications Office of the European Union, Luxembourg, EUR*, vol. 30942, 2021.
- [160] S. Fritz-Morgenthal, B. Hein, and J. Papenbrock, “Financial risk management and explainable, trustworthy, responsible AI,” *Frontiers in Artificial Intelligence*, vol. 5, p. 779799, 2022.
- [161] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, “A survey on adversarial attacks and defences,” *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021.
- [162] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.
- [163] A. Shafahi, W. R. Huang, M. Najibi, O. Suci, C. Studer, T. Dumitras, and T. Goldstein, “Poison frogs! targeted clean-label poisoning attacks on neural networks,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 6106–6116.
- [164] N. Carlini and D. A. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 39–57.
- [165] B. Wang, Y. Yao, B. Viswanath, H. Zheng, and B. Y. Zhao, “With great training comes great vulnerability: Practical attacks against transfer learning,” in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, 2018, pp. 1281–1297.
- [166] S. Rezaei and X. Liu, “A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [167] Y. Zhang, Y. Song, J. Liang, K. Bai, and Q. Yang, “Two sides of the same coin: White-box and black-box attacks for transfer learning,” in *KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 2989–2997.
- [168] A. Mehra, B. Kailkhura, P. Chen, and J. Hamm, “Understanding the limits of unsupervised domain adaptation via data poisoning,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 17 347–17 359.
- [169] D. Hendrycks, K. Lee, and M. Mazeika, “Using pre-training can improve model robustness and uncertainty,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 2712–2721.

- [170] A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, D. W. Jacobs, and T. Goldstein, “Adversarially robust transfer learning,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [171] M. Awais, F. Zhou, H. Xu, L. Hong, P. Luo, S. Bae, and Z. Li, “Adversarial robustness for unsupervised domain adaptation,” in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 8548–8557.
- [172] D. Chen, H. Hu, Q. Wang, Y. Li, C. Wang, C. Shen, and Q. Li, “CARTL: cooperative adversarially-robust transfer learning,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 1640–1650.
- [173] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, “Do adversarially robust imagenet models transfer better?” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [174] F. Utrera, E. Kravitz, N. B. Erichson, R. Khanna, and M. W. Mahoney, “Adversarially-trained deep nets transfer better: Illustration on image classification,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.
- [175] Z. Deng, L. Zhang, K. Vodrahalli, K. Kawaguchi, and J. Y. Zou, “Adversarial training helps transfer learning via better representations,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 25 179–25 191.
- [176] X. Xu, J. Y. Zhang, E. Ma, H. H. Son, S. Koyejo, and B. Li, “Adversarially robust models may not transfer better: Sufficient conditions for domain transferability from the view of regularization,” in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 24 770–24 802.
- [177] W. Bao, J. Wu, and J. He, “BOBA: Byzantine-robust federated learning with label skewness,” in *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, vol. 238. PMLR, 2024, pp. 892–900.
- [178] M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 4615–4625.
- [179] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” in *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, 2020.

- [180] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” *CoRR*, vol. abs/1806.00582, 2018.
- [181] Y. J. Cho, D. Jhunjhunwala, T. Li, V. Smith, and G. Joshi, “To federate or not to federate: incentivizing client participation in federated learning,” in *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.
- [182] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, “Three approaches for personalization with applications to federated learning,” *CoRR*, vol. abs/2002.10619, 2020.
- [183] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. Morency, “Think locally, act globally: Federated learning with local and global representations,” *CoRR*, vol. abs/2001.01523, 2020.
- [184] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, “Federated learning with personalization layers,” *CoRR*, vol. abs/1912.00818, 2019.
- [185] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, “Exploiting shared representations for personalized federated learning,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 2089–2099.
- [186] Y. Deng, M. M. Kamani, and M. Mahdavi, “Adaptive personalized federated learning,” *CoRR*, vol. abs/2003.13461, 2020.
- [187] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, “Multi-center federated learning: clients clustering for better personalization,” *World Wide Web (WWW)*, vol. 26, no. 1, pp. 481–500, 2023.
- [188] V. Smith, C. Chiang, M. Sanjabi, and A. Talwalkar, “Federated multi-task learning,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 4424–4434.
- [189] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, “Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [190] T. Li, S. Hu, A. Beirami, and V. Smith, “Ditto: Fair and robust federated learning through personalization,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 6357–6368.
- [191] C. T. Dinh, N. H. Tran, and T. D. Nguyen, “Personalized federated learning with moreau envelopes,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

- [192] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, “A new look and convergence rate of federated multitask learning with laplacian regularization,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [193] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, “Personalized cross-silo federated learning on non-iid data,” in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, 2021*, pp. 7865–7873.
- [194] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, “Personalized federated learning using hypernetworks,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139. PMLR, 2021, pp. 9489–9502.
- [195] R. Ye, Z. Ni, F. Wu, S. Chen, and Y. Wang, “Personalized federated learning with inferred collaboration graphs,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 39 801–39 817.
- [196] J. Kang, J. He, R. Maciejewski, and H. Tong, “Inform: Individual fairness on graph mining,” in *KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 379–389.
- [197] H. Zhao and G. J. Gordon, “Inherent tradeoffs in learning fair representations,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 15 649–15 659.
- [198] D. Mukherjee, F. Petersen, M. Yurochkin, and Y. Sun, “Domain adaptation meets individual fairness. and they get along,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [199] D. Madras, E. Creager, T. Pitassi, and R. S. Zemel, “Learning adversarially fair and transferable representations,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 3381–3390.
- [200] C. Schumann, X. Wang, A. Beutel, J. Chen, H. Qian, and E. H. Chi, “Transfer of machine learning fairness across domains,” *CoRR*, vol. abs/1906.09688, 2019.
- [201] Y. Chen, R. Raab, J. Wang, and Y. Liu, “Fairness transferability subject to bounded distribution shift,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

- [202] J. Schrouff, N. Harris, S. Koyejo, I. M. Alabdulmohsin, E. Schnider, K. Opsahl-Ong, A. Brown, S. Roy, D. Mincu, C. Chen, A. Dieng, Y. Liu, V. Natarajan, A. Karthikesalingam, K. A. Heller, S. Chiappa, and A. D’Amour, “Diagnosing failures of fairness transfer across distribution shift in real-world medical settings,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [203] A. Coston, K. N. Ramamurthy, D. Wei, K. R. Varshney, S. Speakman, Z. Mustahsan, and S. Chakraborty, “Fair transfer learning with missing protected attributes,” in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27-28, 2019*. ACM, 2019, pp. 91–98.
- [204] M. T. Ribeiro, S. Singh, and C. Guestrin, ““why should I trust you?”: Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 1135–1144.
- [205] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 618–626.
- [206] U. Bhatt, J. Antorán, Y. Zhang, Q. V. Liao, P. Sattigeri, R. Fogliato, G. G. Melançon, R. Krishnan, J. Stanley, O. Tickoo, L. Nachman, R. Chunara, M. Srikumar, A. Weller, and A. Xiang, “Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty,” in *AIES ’21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*. ACM, 2021, pp. 401–413.
- [207] Y. Lee, A. S. Chen, F. Tajwar, A. Kumar, H. Yao, P. Liang, and C. Finn, “Surgical fine-tuning improves adaptation to distribution shifts,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.
- [208] S. Park, O. Bastani, J. Weimer, and I. Lee, “Calibrated prediction with covariate shift via unsupervised domain adaptation,” in *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, vol. 108. PMLR, 2020, pp. 3219–3229.
- [209] X. Wang, M. Long, J. Wang, and M. I. Jordan, “Transferable calibration with lower bias and variance in domain adaptation,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [210] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, “Deep neural networks as gaussian processes,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

- [211] R. M. Neal, “Bayesian learning for neural networks,” Ph.D. dissertation, University of Toronto, Canada, 1995.
- [212] A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani, “Gaussian process behaviour in wide deep neural networks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [213] G. Yang, “Wide feedforward or recurrent neural networks of any architecture are gaussian processes,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 9947–9960.
- [214] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.
- [215] B. Sun and K. 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*, vol. 9915, 2016, pp. 443–450.
- [216] F. Hiai and M. Lin, “On an eigenvalue inequality involving the hadamard product,” *Linear Algebra and its Applications*, vol. 515, pp. 313–320, 2017.
- [217] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, “Correcting sample selection bias by unlabeled data,” in *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, 2006, pp. 601–608.
- [218] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh, “Sample selection bias correction theory,” in *Algorithmic Learning Theory, 19th International Conference, ALT 2008, Budapest, Hungary, October 13-16, 2008. Proceedings*, vol. 5254, 2008, pp. 38–53.
- [219] C. Cortes, Y. Mansour, and M. Mohri, “Learning bounds for importance weighting,” in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, 2010, pp. 442–450.
- [220] A. de Mathelin, G. Richard, F. Deheeger, M. Mougeot, and N. Vayatis, “Adversarial weighting for domain adaptation in regression,” in *33rd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2021, Washington, DC, USA, November 1-3, 2021*, 2021, pp. 49–56.
- [221] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, “dsprites: Disentanglement testing sprites dataset,” <https://github.com/deepmind/dsprites-dataset/>, 2017.

- [222] M. W. Gondal, M. Wuthrich, D. Miladinovic, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer, “On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 15 714–15 725.
- [223] S. Wang, K. Guan, Z. Wang, E. A. Ainsworth, T. Zheng, P. A. Townsend, K. Li, C. Moller, G. Wu, and C. Jiang, “Unique contributions of chlorophyll and nitrogen to predict crop photosynthetic capacity from leaf spectroscopy,” *Journal of experimental botany*, vol. 72, no. 2, pp. 341–354, 2021.
- [224] B. He, B. Lakshminarayanan, and Y. W. Teh, “Bayesian deep ensembles via the neural tangent kernel,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [225] W. J. Maddox, S. Tang, P. G. Moreno, A. G. Wilson, and A. C. Damianou, “Fast adaptation with linearized neural networks,” in *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, vol. 130. PMLR, 2021, pp. 2737–2745.
- [226] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz, “Neural tangents: Fast and easy infinite neural networks in python,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [227] J. Lee, S. S. Schoenholz, J. Pennington, B. Adlam, L. Xiao, R. Novak, and J. Sohl-Dickstein, “Finite versus infinite neural networks: an empirical study,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [228] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha, “Marginalized denoising autoencoders for domain adaptation,” in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.
- [229] H. Li, S. J. Pan, S. Wang, and A. C. Kot, “Domain generalization with adversarial feature learning,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 5400–5409.
- [230] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, “Invariant risk minimization,” *CoRR*, vol. abs/1907.02893, 2019.
- [231] X. Peng, Z. Huang, X. Sun, and K. Saenko, “Domain agnostic learning with disentangled representations,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 5102–5112.

- [232] M. Bui, T. Tran, A. Tran, and D. Q. Phung, “Exploiting domain-specific features to enhance domain generalization,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 21 189–21 201.
- [233] T. Xu, W. Chen, P. Wang, F. Wang, H. Li, and R. Jin, “CDTrans: cross-domain transformer for unsupervised domain adaptation,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.
- [234] Y. Zhang and D. Yeung, “A convex formulation for learning task relationships in multi-task learning,” in *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, 2010, pp. 733–442.
- [235] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Label propagation for deep semi-supervised learning,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 5070–5079.
- [236] J. Ni, H. Tong, W. Fan, and X. Zhang, “Inside the atoms: ranking on a network of networks,” in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 2014, pp. 1356–1365.
- [237] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, 2003, pp. 321–328.
- [238] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using gaussian fields and harmonic functions,” in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, 2003, pp. 912–919.
- [239] L. Li and H. Tong, “The child is father of the man: Foresee the success at the early stage,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. ACM, 2015, pp. 655–664.
- [240] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, “Wide neural networks of any depth evolve as linear models under gradient descent,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 8570–8581.
- [241] R. Novak, J. Sohl-Dickstein, and S. S. Schoenholz, “Fast finite width neural tangent kernel,” in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, vol. 162. PMLR, 2022, pp. 17 018–17 044.

- [242] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, “Understanding traffic density from large-scale web camera data,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 4264–4273.
- [243] R. Misra, “News category dataset,” *CoRR*, vol. abs/2209.11429, 2022.
- [244] H. Yao, C. Choi, B. Cao, Y. Lee, P. W. Koh, and C. Finn, “Wild-time: A benchmark of in-the-wild distribution shift over time,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [245] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” *CoRR*, vol. abs/1910.01108, 2019.
- [246] C. B. Clement, M. Bierbaum, K. P. O’Keeffe, and A. A. Alemi, “On the use of arxiv as a dataset,” *CoRR*, vol. abs/1905.00075, 2019.
- [247] O. Delalleau, Y. Bengio, and N. L. Roux, “Efficient non-parametric function induction in semi-supervised learning,” in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, 2005.
- [248] Z. Xu, H. He, G. Lee, B. Wang, and H. Wang, “Graph-relational domain adaptation,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.
- [249] M. Pezeshki, S. Kaba, Y. Bengio, A. C. Courville, D. Precup, and G. Lajoie, “Gradient starvation: A learning proclivity in neural networks,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 1256–1272.
- [250] Y. Shi, J. Seely, P. H. S. Torr, S. Narayanaswamy, A. Y. Hannun, N. Usunier, and G. Synnaeve, “Gradient matching for domain generalization,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.
- [251] C. Eastwood, A. Robey, S. Singh, J. von Kügelgen, H. Hassani, G. J. Pappas, and B. Schölkopf, “Probable domain generalization via quantile risk minimization,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.
- [252] R. P. Lencevicius, “An empirical analysis of the laplace and neural tangent kernels,” M.S. thesis, California State Polytechnic University, Pomona, August 2022, <http://hdl.handle.net/20.500.12680/d504rr81v>.

- [253] A. Bobu, E. Tzeng, J. Hoffman, and T. Darrell, “Adapting to continuously shifting domains,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018.
- [254] H. Liu, M. Long, J. Wang, and Y. Wang, “Learning to adapt to evolving domains,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [255] C. Cortes and M. Mohri, “Domain adaptation in regression,” in *Algorithmic Learning Theory - 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings*, vol. 6925. Springer, 2011, pp. 308–323.
- [256] M. Mohri and A. M. Medina, “New analysis and algorithm for learning with drifting distributions,” in *Algorithmic Learning Theory - 23rd International Conference, ALT 2012, Lyon, France, October 29-31, 2012. Proceedings*, vol. 7568. Springer, 2012, pp. 124–138.
- [257] J. Hoffman, T. Darrell, and K. Saenko, “Continuous manifold based adaptation for evolving visual domains,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 2014, pp. 867–874.
- [258] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70. PMLR, 2017, pp. 1126–1135.
- [259] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains,” in *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, vol. 6314, 2010, pp. 213–226.
- [260] Z. Fang, J. Lu, F. Liu, J. Xuan, and G. Zhang, “Open set domain adaptation: Theoretical bound and algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4309–4322, 2021.
- [261] Y. Luo, Z. Wang, Z. Huang, and M. Baktashmotlagh, “Progressive graph learning for open-set domain adaptation,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 6468–6478.
- [262] P. Bickel and E. Levina, “Some theory for fisher’s linear discriminant function, ‘naive bayes’, and some alternatives when there are many more variables than observations,” *Bernoulli*, vol. 10, 12 2004.
- [263] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, “Positive-unlabeled learning with non-negative risk estimator,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 1675–1685.

- [264] Y. Xu, C. Xu, C. Xu, and D. Tao, “Multi-positive and unlabeled learning,” in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 2017, pp. 3182–3188.
- [265] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5385–5394.
- [266] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko, “Syn2real: A new benchmark for synthetic-to-real visual domain adaptation,” *CoRR*, vol. abs/1806.09755, 2018.
- [267] T. Shermin, G. Lu, S. W. Teng, M. M. Murshed, and F. Sohel, “Adversarial network with multiple classifiers for open set domain adaptation,” *IEEE Transactions on Multimedia*, vol. 23, pp. 2732–2744, 2021.
- [268] H. Liu, Z. Cao, M. Long, J. Wang, and Q. Yang, “Separate to adapt: Open set domain adaptation via progressive separation,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019, pp. 2927–2936.
- [269] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [270] F. Geerts, F. Mazowiecki, and G. A. Pérez, “Let’s agree to degree: Comparing graph convolutional networks in the message-passing framework,” in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 3640–3649.
- [271] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 1725–1735.
- [272] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, “*struc2vec*: Learning node representations from structural identity,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2017, pp. 385–394.
- [273] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “ArnetMiner: extraction and mining of academic social networks,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008, pp. 990–998.

- [274] X. Shen, Q. Dai, F. Chung, W. Lu, and K. Choi, “Adversarial deep network embedding for cross-network node classification,” in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2020, pp. 2991–2999.
- [275] J. Li, X. Hu, J. Tang, and H. Liu, “Unsupervised streaming feature selection in social media,” in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. ACM, 2015, pp. 1041–1050.
- [276] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 6861–6871.
- [277] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. G. Lanckriet, “Hilbert space embeddings and metrics on probability measures,” *Journal of Machine Learning Research*, vol. 11, pp. 1517–1561, 2010.
- [278] E. Chien, J. Peng, P. Li, and O. Milenkovic, “Adaptive universal generalized pagerank graph neural network,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.
- [279] T. Wang, D. Jin, R. Wang, D. He, and Y. Huang, “Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily,” in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 2022, pp. 4210–4218.
- [280] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 4805–4815.
- [281] A. Bietti and J. Mairal, “On the inductive bias of neural tangent kernels,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 12 873–12 884.
- [282] Z. Niu, M. Anitescu, and J. Chen, “Graph neural network-inspired kernels for gaussian processes in semi-supervised learning,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023.

- [283] Y. Cho and L. K. Saul, “Kernel methods for deep learning,” in *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, 2009, pp. 342–350.
- [284] E. L. Snelson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” in *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, 2005, pp. 1257–1264.
- [285] K. M. A. Chai, “Generalization errors and learning curves for regression with multi-task gaussian processes,” in *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, 2009, pp. 279–287.
- [286] B. Rozemberczki, C. Allen, and R. Sarkar, “Multi-scale attributed node embedding,” *Journal of Complex Networks*, vol. 9, no. 2, 2021.
- [287] C. M. Montes, C. Fox, Á. Sanz-Sáez, S. P. Serbin, E. Kumagai, M. D. Krause, A. Xavier, J. E. Specht, W. D. Beavis, C. J. Bernacchi et al., “High-throughput characterization, correlation, and mapping of leaf photosynthetic and functional traits in the soybean (glycine max) nested association mapping population,” *Genetics*, vol. 221, no. 2, 2022.
- [288] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, “Geom-GCN: geometric graph convolutional networks,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [289] Y. C. Ng, N. Colombo, and R. Silva, “Bayesian semi-supervised learning with graph gaussian processes,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 1690–1701.
- [290] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. Lim, “Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 20 887–20 902.
- [291] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, “MixHop: higher-order graph convolutional architectures via sparsified neighborhood mixing,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, vol. 97. PMLR, 2019, pp. 21–29.

- [292] J. R. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, “Gpytorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 7587–7597.
- [293] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [294] J. Wu and J. He, “A unified framework for adversarial attacks on multi-source domain adaptation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 039–11 050, 2023.
- [295] N. M. Müller and K. Markert, “Identifying mislabeled instances in classification datasets,” in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*. IEEE, 2019, pp. 1–8.
- [296] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, “Towards poisoning of deep learning algorithms with back-gradient optimization,” in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*. ACM, 2017, pp. 27–38.
- [297] S. Mei and X. Zhu, “Using machine teaching to identify optimal training-set attacks on machine learners,” in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, 2015, pp. 2871–2877.
- [298] A. Gretton, B. K. Sriperumbudur, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, and K. Fukumizu, “Optimal kernel choice for large-scale two-sample tests,” in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, 2012, pp. 1214–1222.
- [299] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, 1998.
- [300] J. J. Hull, “A database for handwritten text recognition research,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.
- [301] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng et al., “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, vol. 2011, no. 5, 2011, p. 7.
- [302] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, “VisDA: the visual domain adaptation challenge,” *CoRR*, vol. abs/1710.06924, 2017.

- [303] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. ACM, 2017, pp. 506–519.
- [304] G. Zizzo, A. Rawat, M. Sinn, and B. Buesser, “FAT: federated adversarial training,” *CoRR*, vol. abs/2012.01791, 2020.
- [305] D. Shah, P. Dube, S. Chakraborty, and A. Verma, “Adversarial training in communication constrained federated learning,” *CoRR*, vol. abs/2103.01319, 2021.
- [306] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [307] G. Valentini and T. G. Dietterich, “Bias-variance analysis of support vector machines for the development of svm-based ensemble methods,” *Journal of Machine Learning Research*, vol. 5, pp. 725–775, 2004.
- [308] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [309] C. Xie and A. L. Yuille, “Intriguing properties of adversarial training at scale,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
- [310] P. Domingos, “A unified bias-variance decomposition and its applications,” in *International Conference on Machine Learning*, 2000, pp. 231–238.
- [311] Z. Yang, Y. Yu, C. You, J. Steinhardt, and Y. Ma, “Rethinking bias-variance trade-off for generalization of neural networks,” in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, vol. 119. PMLR, 2020, pp. 10 767–10 777.
- [312] S. Geman, E. Bienenstock, and R. Doursat, “Neural networks and the bias/variance dilemma,” *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.
- [313] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, “A survey on homomorphic encryption schemes: Theory and implementation,” *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–35, 2018.
- [314] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *CoRR*, vol. abs/1708.07747, 2017.
- [315] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” *Master’s thesis, Department of Computer Science, University of Toronto*, 2009.

- [316] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [317] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [318] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 1800–1807.
- [319] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 4510–4520.
- [320] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, “Learning to detect malicious clients for robust federated learning,” *CoRR*, vol. abs/2002.00211, 2020.
- [321] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 119–129.
- [322] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 5636–5645.
- [323] Y. Chen, L. Su, and J. Xu, “Distributed statistical machine learning in adversarial settings: Byzantine gradient descent,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.
- [324] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Álvarez, “Personalized federated learning with first order model optimization,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.
- [325] S. Vahidian, M. Morafah, C. Chen, M. Shah, and B. Lin, “Rethinking data heterogeneity in federated learning: Introducing a new notion and standard benchmarks,” *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 3, pp. 1386–1397, 2024.
- [326] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU press, 2013.
- [327] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web.” Stanford InfoLab, Technical Report, 1999.

- [328] S. Ginosar, K. Rakelly, S. Sachs, B. Yin, and A. A. Efros, “A century of portraits: A visual historical record of american high school yearbooks,” in *2015 IEEE International Conference on Computer Vision Workshop, ICCV Workshops 2015, Santiago, Chile, December 7-13, 2015*, 2015, pp. 652–658.
- [329] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “The german traffic sign recognition benchmark: A multi-class classification competition,” in *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*. IEEE, 2011, pp. 1453–1460.
- [330] J. Ma, G. Long, T. Zhou, J. Jiang, and C. Zhang, “On the convergence of clustered federated learning,” *CoRR*, vol. abs/2202.06187, 2022.
- [331] T. H. Hsu, H. Qi, and M. Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” *CoRR*, vol. abs/1909.06335, 2019.
- [332] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [333] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, “Robust federated learning in a heterogeneous environment,” *CoRR*, vol. abs/1906.06629, 2019.