

Collaborative Learning under Data Heterogeneity

Junki Mori

Abstract

Recent advances in deep learning show that model performance consistently improves with larger and more diverse datasets. However, in many practical scenarios, a single data holder lacks sufficient data to benefit from such scaling. Collaborative learning, in which multiple parties jointly train a model using their combined data resources, offers a promising solution, yet direct data sharing is often infeasible due to privacy and confidentiality constraints. Federated learning provides a privacy-preserving framework by enabling model training across distributed data without exchanging raw data. Despite its potential, collaboration remains fundamentally challenged by data heterogeneity, including differences in underlying data distributions across parties and inconsistencies in observable features or label availability, whether or not the data are centrally shared.

This thesis conducts a systematic study of data heterogeneity in collaborative learning, focusing on supervised learning across multiple parties under two primary forms of heterogeneity: (1) distribution heterogeneity, arising from differences in underlying data distributions (e.g., covariate and label shift), and (2) observation heterogeneity, stemming from discrepancies in observable features or label availability. The thesis develops practical algorithms to enable effective collaboration despite these discrepancies.

First, we propose a personalized federated learning method designed for scenarios with heterogeneous label distributions. By introducing a multi-branch architecture that jointly captures global shared patterns and party-specific behaviors, the method achieves robust performance under diverse label-shift conditions. Second, we investigate federated learning under feature-space heterogeneity, where parties possess only partially overlapping features. We develop a neural architecture that separates common and party-specific representations, allowing each party to exploit its unique features while benefiting from collaboration through shared features. Third, we address a federated adaptation scenario where all parties hold unlabeled data from distinct domains and only a pretrained source model is available. We intro-

duce a domain-aware aggregation framework that leverages pseudo-label refinement and similarity computed solely from shared model parameters, ensuring privacy-preserving and domain-sensitive model updates. Finally, we explore a two-party (source and target) domain adaptation setting with centrally aggregated but highly heterogeneous data, combining extreme label-shift conditions, mismatched feature spaces, and lack of labels. A novel algorithm is developed to integrate heterogeneous domain adaptation with positive-unlabeled learning, enabling effective cross-domain knowledge transfer in realistic applications such as cross-industry potential customer prediction.

Through comprehensive analysis and the development of algorithms tailored to diverse practical constraints, this thesis expands the possibilities for organizational data collaboration, enabling robust learning across realistically heterogeneous environments.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Hisashi Kashima, for his continuous support throughout my doctoral studies and insightful advice. I am also deeply grateful to the members of committee for this thesis, Professor Akihiro Yamamoto and Professor Ko Nishino, for their careful review of my thesis and for providing helpful comments that improved the quality of this thesis.

Furthermore, I would like to thank my collaborators, Professor Jun Sakuma, Dr. Isamu Teranishi, Mr. Ryo Furukawa, Mr. Taiki Miyagawa, Dr. Akinori Ebihara, Dr. Kosuke Kihara, and Mr. Tomoyuki Yoshiyama, for their insightful discussions, valuable advice, and dedicated efforts in improving our articles.

Finally, I would like to thank my parents and my wife for their continuous support in my daily life, as well as my friends for their encouragement.

List of Publications

Major Publications

This thesis is based on the following publications.

Refereed Conference Proceedings

1. (Chapter 3) **Junki Mori**, Tomoyuki Yoshiyama, Ryo Furukawa, and Isamu Teranishi. Personalized Federated Learning with Multi-branch Architecture. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1 – 8, 2023.
2. (Chapter 4) **Junki Mori**, Isamu Teranishi, and Ryo Furukawa. Continual Horizontal Federated Learning for Heterogeneous Data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1 – 8, 2022.
3. (Chapter 5) **Junki Mori**, Kosuke Kihara, Taiki Miyagawa, Akinori F. Ebihara, Isamu Teranishi, and Hisashi Kashima. Federated Source-Free Domain Adaptation for Classification: Weighted Cluster Aggregation for Unlabeled Data. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6879-6889, 2025.
4. (Chapter 6) **Junki Mori**, Ryo Furukawa, Isamu Teranishi, and Jun Sakuma. Heterogeneous Domain Adaptation with Positive and Unlabeled Data. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 778-787, 2023.

Refereed Journal Articles

1. (Chapter 4) **森隼基**, 寺西勇, 古川諒. 重複特徴量と非重複特徴量に対する継続学習による連合学習の改善. 人工知能学会論文誌, 39(3), A-N72_1-11, 2024.

Others

Refereed Conference Proceedings

1. Rishav Chourasia, Batnyam Enkhtaivan, Kunihiro Ito, **Junki Mori**, Isamu Teranishi, and Hikaru Tsuchida. Knowledge Cross-Distillation for Membership Privacy. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2, 362 – 377, 2022.
2. Masahiro Hayashitani, **Junki Mori**, and Isamu Teranishi. Survey of Privacy Threats and Countermeasures in Federated Learning. *The 3rd IEEE International Conference on Federated Learning Technologies and Applications (FLTA)*, 2025.

Refereed Conference Workshops

1. Kosuke Kihara, **Junki Mori**, Taiki Miyagawa, and Akinori F. Ebihara. Rethinking the Backbone in Class Imbalanced Federated Source Free Domain Adaptation: The Utility of Vision Foundation Models. Satellite Workshop 1 at IEEE ICIP 2025, 2025.

Contents

1	Introduction	21
1.1	Background and Motivation	21
1.2	Data Heterogeneity	22
1.2.1	Distribution Heterogeneity	24
1.2.2	Observation Heterogeneity	24
1.3	Related Works	25
1.3.1	Federated Learning under Distribution Heterogeneity	25
1.3.2	Federated Learning with Heterogeneous Features	25
1.3.3	Federated Learning with Unlabeled Data	26
1.3.4	Heterogeneous Unsupervised Domain Adaptation	26
1.4	Contributions and Organization	26
2	Preliminaries	29
2.1	Problem Setting	29
2.1.1	General Formulation	29
2.1.2	Data Heterogeneity	30
2.2	Federated Averaging (FedAvg)	32
3	Personalized Federated Learning with Multi-branch Architecture	35
3.1	Introduction	35
3.2	Related Works	36
3.2.1	Multi-branch Neural Networks	36
3.2.2	Federated Learning Based on Multi-branch Neural Networks	37
3.3	Problem Formulation	37
3.4	Proposed Method	38
3.4.1	Architecture	38
3.4.2	Algorithm	39
3.5	Experiments	41

3.5.1	Experimental Setup	41
3.5.2	Process of Personalization	43
3.5.3	Performance Comparison	45
3.5.4	Effects of Number of Branches	46
3.6	Discussion	47
3.7	Summary	48
3.8	Appendix	48
4	Federated Learning Enhanced by Continual Learning for Common and Uncommon Features	51
4.1	Introduction	51
4.2	Related Work	52
4.2.1	Horizontal Federated Learning under Feature-Space Heterogeneity	52
4.2.2	Continual Learning (CL)	53
4.3	Preliminaries	53
4.3.1	Progressive Neural Network (PNN)	53
4.3.2	Problem Formulation	54
4.4	Proposed Approach	55
4.4.1	Idea	55
4.4.2	Model Architecture	55
4.4.3	Algorithm	56
4.5	Experiments	58
4.5.1	Experimental Settings	58
4.5.2	Experimental Results	59
4.6	Summary	62
5	Federated Source-free Domain Adaptation for Classification: Weighted Cluster Aggregation for Unlabeled Data	63
5.1	Introduction	63
5.2	Related Work	65
5.2.1	Source-free Domain Adaptation	65
5.2.2	Federated Domain Adaptation	66
5.2.3	Federated Source-free Domain Adaptation	66
5.3	Problem Formulation	66
5.4	Method	67

5.4.1	Private and Parameter-free Client Clustering	67
5.4.2	Weighted Cluster Aggregation	68
5.4.3	Local Adaptation with Pseudo-Labeling	70
5.5	Experiment	73
5.5.1	Comparison with Baselines	74
5.5.2	Ablation Studies	75
5.6	Summary	77
5.7	Appendix	78
5.7.1	Notations	78
5.7.2	Details of Method	78
5.7.3	Detailed Experimental Settings	81
5.7.4	Limitations and Discussions	83
5.7.5	Supplementary Experimental Results	85

6 Heterogeneous Domain Adaptation with Positive and Unlabeled

Data		93
6.1	Introduction	93
6.2	Related Works	96
6.2.1	Heterogeneous Unsupervised Domain Adaptation	96
6.2.2	PU Learning	97
6.2.3	Domain Adaptation in PU Learning Setting	97
6.3	Positive and Unlabeled Heterogeneous Domain Adaptation	98
6.3.1	Problem Formulation	98
6.3.2	Difficulty of PU-HUDA	98
6.4	Proposed Approach	99
6.4.1	Basic Idea	99
6.4.2	Naive Optimization Approach	101
6.4.3	Predictive Adversarial Domain Adaptation	101
6.4.4	Soft labeling mechanism to improve PADA	103
6.5	Experiments	104
6.5.1	Experimental Setup	104
6.5.2	Results and Analysis	107
6.5.3	Adaptation analysis of PADA	110
6.6	Summary	111

7	Conclusion and Future Direction	113
7.1	Conclusion	113
7.2	Future Direction	114
	Bibliography	117

List of Figures

1.1	An overview of data heterogeneity across two parties. Distribution heterogeneity refers to differences in the underlying data distributions, such as covariate shift or label shift, while observation heterogeneity arises from discrepancies in observable information, including feature-space heterogeneity and label-availability heterogeneity. . . .	23
3.1	Architecture of multi-branch layer.	39
3.2	Architecture of pFedMB.	39
3.3	Visualization of α at communication round r with/without α -weighted averaging using CIFAR10.	42
3.4	The mean test accuracy among 10 clients grouped into 5 pairs at each communication round using CIFAR10. Each method will be followed by local fine-tuning.	44
3.5	Variation of the mean test accuracy among 50 clients in the random 2 classes distribution setting using CIFAR10 with respect to the number of branches.	46
4.1	Description of a progressive neural network with three layers in the case of two tasks.	54
4.2	View of dataset structure in the two client scenario.	54
4.3	Architecture of FLEC for the case of $K = 2$ and $L = 3$. $z_{k,l}^c$ and $z_{k,l}^u$ denote the outputs of l -th layer of common and unique columns for client k 's input, respectively.	56
4.4	Variation of average test accuracy for covertedype with respect to the ratio of the common features.	60

5.1	Overview of our problem setting: FFREEDA [1]. A pre-trained model on a labeled <i>source</i> domain dataset exists on the server, which becomes inaccessible following pre-training (<i>source-free</i>). Clients possess unlabeled datasets from specific <i>target</i> domains, which may or may not be identical. We aim to train personalized models tailored to each client’s domain.	64
5.2	Weighted cluster aggregation (WCA). Our WCA consists of server-side (left) and client-side (right) operations. The client-side operations are represented by client 1 in cluster 1. Firstly, the server forms cluster models by averaging the local models within each cluster and develops soft cluster models by integrating models from other clusters. Next, the client merges these soft cluster models utilizing cluster weights determined by how each feature extractor aligns each sample with the respective classifier vector. Additionally, the client combines the merged model with the corresponding cluster model using Soft Neighborhood Density (SND) weights [2].	68
5.3	Pseudo-labeling strategy. The operation is signified by client 1 in cluster 1. Our strategy involves two phases: prototype-based pseudo-labeling with initial and cluster models, and mixup to generate samples with more reliable pseudo-labels. The first phase selects pseudo-labels closer to prototypes, while the second mixes samples with mismatched and matched pseudo-labels.	71
5.4	Comparison of average accuracy over clients within each target domain. The source domain is MNIST for Digit-Five (top) and Art Painting for PACS (bottom). The target domains are the others. FedWCA considerably enhances the accuracy of clients in specific target domains such as SVHN for Digit-Five and Sketch for PACS, while maintaining the performance of other domains.	74
5.5	Visualization of cluster weights for FedWCA-L and FedWCA when the communication round r is 2 and 50. Art Painting in PACS dataset is used for the source dataset. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. See the first row of Table 5.4 for the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. The number represents the weight for the corresponding cluster.	83

5.6	Images possessed by clients in each cluster. USPS of Digit-Five serves as the source domain. Our FedWCA clusters clients according to their local models, almost matching their data domains.	87
6.1	View of the data structure in two HUDA settings, (a) conventional one and (b) new one where the source domain has only positive data. Here, both of them assume that the two domains share some features.	94
6.2	Comparison of adaptation mechanisms in existing HUDA methods (top) and our proposed PU-HUDA method (bottom).	95
6.3	Overview of the proposed method, PADA using soft-labeling mechanism.	99
6.4	Effect of the number of selected common features for Movielens-Netflix	109

List of Tables

1.1	Overview of learning frameworks and types of data heterogeneity addressed in each chapter.	27
3.1	The mean test accuracy (%) among 10 clients grouped into 5 pairs using CIFAR10. Each method includes local fine-tuning for local epochs E	44
3.2	The mean test accuracy (%) among 15 clients and 50 clients in random 2 classes and Dirichlet distribution settings using CIFAR10. Client participation rate at each communication round is 100% and 20%, respectively. Each method includes local fine-tuning for local epochs E . “Local” is a non-federated local learning.	45
3.3	Tuned number of branches of pFedMB in each experimental setting. .	46
4.1	Types of datasets	58
4.2	Mean and standard deviation (%) of accuracy for the baselines and FLEC across the five datasets.	60
4.3	Mean and standard deviation (%) of accuracy for the baselines and FLEC with different numbers of clients on the covertedype dataset. . .	61
4.4	Mean and standard deviation (%) of accuracy for the baselines and FLEC under different correlation strengths on the helena dataset. . .	61

5.1	Related problem settings. Our study focuses on classification tasks, tackling three distinct problem types: "No Labels," "Domain Shift," and "Personalization." Δ denotes specific characteristics depending on FL paradigms: while most Personalized FL research concentrates on label shifts, certain studies like [3, 4] also tackle domain shifts. Semi-Supervised FL literature typically involves unlabeled data, presuming the presence of labeled data either on the server or within a subset of clients, a scenario also observed in some FDA studies.	65
5.2	Results (Digit-Five, PACS, and Office-Home). Each row represents the selected source domain denoted by the initial two letters for each dataset, and the numbers are the mean values $\pm$ standard deviations of averaged accuracy (%) across all clients and all target domains. Thees figures are computed from 15 trials. The best numbers in each row are denoted in bold and the second best are underlined.	72
5.3	Ablation on cluster weights calculation. The FedWCA's cluster weights calculation is replaced by various methods such as one-hot weights (One-hot), equal weights (Equal), a hybrid of them (One-Equal), an adaptive version of One-Equal (One-Equal-A), and a local calculation-only FedWCA variant described in Section 5.4.2 (FedWCA-L). The original FedWCA surpasses all other methods across all PACS source domains.	75
5.4	Ablation on layers used for clustering. Art Painting of PACS is used as the source domain, and ResNet-18 is employed. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. The table below reports the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. Ideally, each client group (C, C', C'') , (P, P', P'') , and (S, S', S'') should be grouped together, as is achieved by the first and second layers.	76
5.5	Ablation on pseudo-labeling strategy. Key components of FedWCA are PP : prototype-based pseudo-labeling, FP : fixed pseudo-labels in each round, TM : two-models-based pseudo-labeling using initial and cluster models, and MU : mixup. All components prove effective in most PACS source domains.	77

5.6	Comparison with extended LADD. The prior FFREEDA method LADD is extended by incorporating PP : prototype-based pseudo-labeling, FP : fixed pseudo-labels in each round, and IM : IM loss. Nevertheless, FedWCA outperforms it.	78
5.7	Notation.	79
5.8	Dataset information including number of samples, classes, and clients for three datasets.	81
5.9	Hyperparameters for our FedWCA including the learning rate (lr), balancing parameter for cross-entropy loss λ , balancing parameter for mixup μ , and temperature parameters T_a , T_b . Only for SYNTH of Digit-Five and Clipart of Office-Home source domains, T_a is set to 0.001 and 0.1, respectively.	82
5.10	Results for FedWCA modification reducing costs. The numbers are the mean values $\pm$ standard deviations of the averaged accuracy (%) across all clients and all target domains. In the latter $U - 1$ rounds of every U round, the initial model of the FedWCA can be computed server-side as opposed to client-side, thereby cutting communication, storage, and computational costs. This revised FedWCA boasts an equivalent performance to its original counterpart when $U = 5$	84
5.11	Full results for layer dependence of clustering. Art Painting of PACS is used as the source domain, and ResNet-18 is employed. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. The table below reports the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. Ideally, each client group (C, C', C'') , (P, P', P'') , and (S, S', S'') should be grouped together, as is achieved by the first and second layers.	86
5.12	Results of client clustering in Digit-Five for each source domain. The clustering algorithm FINCH is applied to Lenet. Each row represents client IDs. Clients 1 to 8, 9 to 16, 17 to 24 and 25 to 32 each belong to the same domain. The table below reports the cluster IDs (denoted by, e.g., $\bigcirc$, $\triangle$, $\square$, and $\diamond$) assigned to each client. Ideally, each client group $(1,2,3,4,5,6,7,8)$, $(9,10,11,12,13,14,15,16)$, $(17,18,19,20,21,22,23,24)$, and $(25,26,27,28,29,30,31,32)$ should be grouped together, as is achieved by SVHN source domain.	88

5.13	Ablation on clustering algorithm. Three clustering algorithms are applied to FedWCA. ‘True’ uses the hypothetical true clusters based on the domain labels. Even without specific cluster information, FedWCA offers similar performance to ‘True’.	89
5.14	Performance comparison with extended FL methods (Digit-Five and PACS). The existing FL methods, FedProx [5], FedAMP [6], and IFCA [7] are extended to FFREEDA by incorporating PP : prototype-based pseudo-labeling, FP : fixed pseudo-labels in each round, and IM : IM loss. FedWCA outperforms them in all source domains.	89
6.1	Setting comparison of this chapter with existing HUDA methods. . .	96
6.2	Detailed information about the processed datasets, Movielens-Netflix, 20-Newsgroups, and Default of credit card clients.	105
6.3	Performance comparison using accuracy (%) for Movielens-Netflix dataset. Four common features were randomly selected five times, which makes 5 experimental settings.	107
6.4	Performance comparison using accuracy (%) for 20-Newsgroups dataset. We selected two top categories and assigned them as positive and negative labels.	107
6.5	Performance comparison using accuracy (%) and AUC (%) for Default of credit card clients dataset. The positive ratio of target data was varied from 0.5 to 0.1.	108
6.6	The average test accuracy (%) of the test discriminator for each method using Movielens-Netflix.	111

Chapter 1

Introduction

1.1 Background and Motivation

Recent progress in deep learning has enabled high-performance models across various domains such as computer vision [8, 9], natural language processing [10, 11], audio and speech processing [12, 13], and tabular data [14, 15]. These advances have been driven largely by the availability of large-scale, high-quality training data, and numerous empirical studies have demonstrated that the performance of deep neural networks continues to improve as the size of training data increases, a phenomenon known as the *scaling law* [16, 17]. However, in many practical scenarios, a single data owner typically does not possess sufficient data diversity or quantity to fully exploit such scaling behavior.

One solution to this limitation is to aggregate data from multiple data owners (hereafter referred to as *parties*¹) and collaboratively train a model that leverages their collective knowledge. Indeed, the idea of exploiting data from a source domain to improve performance in a target domain with limited data has been widely studied under *domain adaptation* (DA) [18], which has proved effective across many applications [19–22]. Yet, in many real-world settings, data held by each party cannot be freely shared due to privacy regulations, confidentiality requirements, and institutional policies. To address these concerns, *federated learning* (FL) [23] has emerged as a privacy-preserving alternative. FL enables multiple parties (clients) to jointly train a machine learning model without exchanging their raw data. Instead, each party performs local training on its own data and shares only model parameters

¹*Party* is used in a general sense to denote independent data-holding entities such as organizations, institutions, devices, and *clients* in federated learning.

or gradients, which are then aggregated to update a global model in a central server. This paradigm maintains data locality while enabling multi-party collaboration, and it has already been deployed in applications such as mobile keyboard prediction [24], healthcare analytics [25, 26], and credit card fraud detection [27].

A fundamental challenge in such *collaborative learning*² is *data heterogeneity*. In practice, the data held by each party often differ in two key ways: (1) their underlying data-generating distributions may not be identical, and (2) some features or labels may be unobserved for some parties. We refer to the former as *distribution heterogeneity* and the latter as *observation heterogeneity*. These discrepancies can hinder naive collaborative approaches, sometimes even degrading performance due to distribution mismatch, or making it algorithmically difficult to construct a unified model when feature spaces or label availability differ. Although several studies have addressed specific aspects of heterogeneity, these efforts remain fragmented, and a systematic investigation of data heterogeneity in collaborative learning is still lacking.

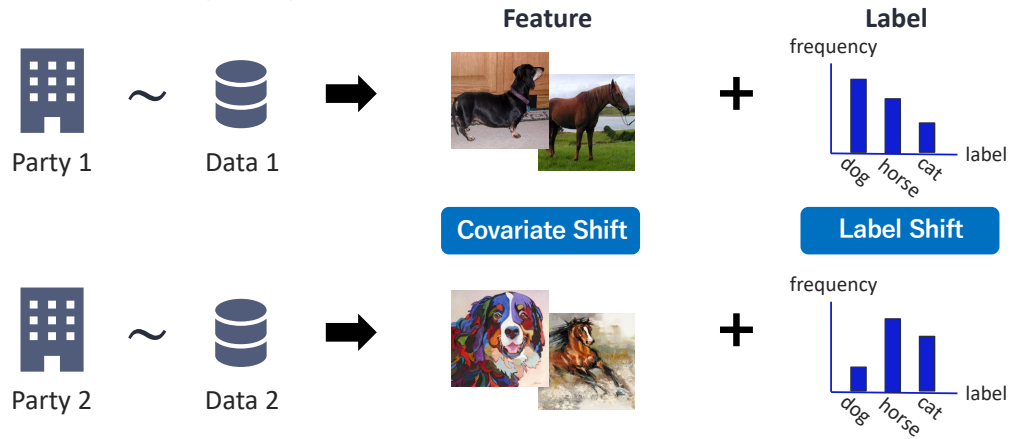
The main goal of this thesis is to provide such a systematic investigation and to develop methods that enable effective collaboration under diverse types of heterogeneity. Although FL and DA are often studied as separate research areas, both can be viewed under a unified perspective of collaborative learning, where multiple data sources collectively contribute to model training. This unified viewpoint provides a common foundation for the methods developed in this thesis.

1.2 Data Heterogeneity

Throughout this thesis, we focus on collaborative *supervised* learning, where each party possesses input-label pairs. When multiple parties collaborate, the data they collect may differ in several important ways. We categorize heterogeneity into two primary types: distribution heterogeneity and observation heterogeneity, illustrated in Figure 1.1. We provide a brief overview below and defer their formal definitions to Chapter 2.

²We use *collaborative learning* in a broad sense to refer to any setting where multiple parties jointly train a model, including both data-sharing approaches and federated learning.

Distribution Heterogeneity



Observation Heterogeneity

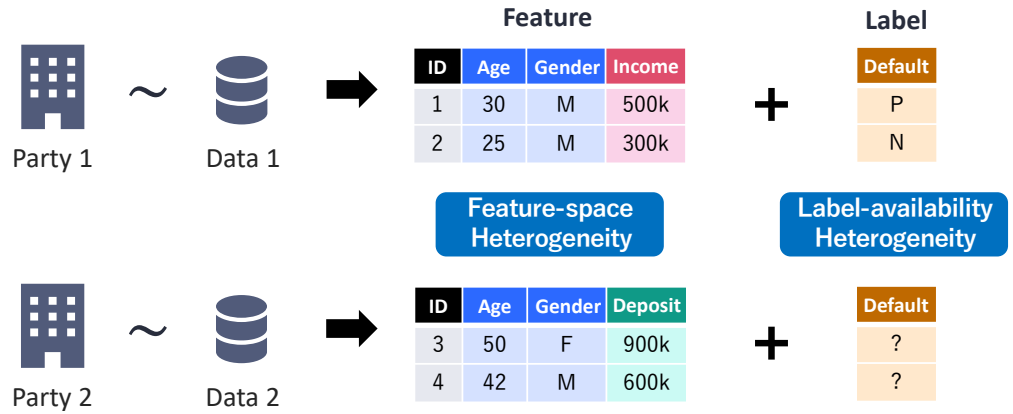


Figure 1.1: An overview of data heterogeneity across two parties. Distribution heterogeneity refers to differences in the underlying data distributions, such as covariate shift or label shift, while observation heterogeneity arises from discrepancies in observable information, including feature-space heterogeneity and label-availability heterogeneity.

1.2.1 Distribution Heterogeneity

Distribution heterogeneity refers to differences in the statistical properties of data collected by different parties. Such differences may arise due to variations in demographics, environments, sensor characteristics, or data-collection processes. Two well-studied forms of distribution heterogeneity include:

- **Covariate Shift:** When the distributions of input features are different while the underlying relationship between inputs and labels remains consistent across parties, the discrepancy is known as *covariate shift* [28]. Covariate shift commonly appears when parties operate in different sensing environments or target different subpopulations.
- **Label Shift:** Conversely, parties may share similar class-conditional distributions but differ in the prevalence of each label. This type of mismatch is referred to as *label shift* [29]. Label shift often arises in applications with varying class frequencies across institutions, such as medical diagnosis or fraud detection.

Such differences in underlying distributions can degrade performance in FL, where model parameters are aggregated across parties [30].

1.2.2 Observation Heterogeneity

Observation heterogeneity arises when parties differ in what information they can observe. This affects the structure of the inputs or the availability of labels:

- **Feature-Space Heterogeneity:** Parties may record different sets of input features due to hardware differences, domain-specific sensor availability, or data collection policies. Some features may be available to all parties (common features), while others may be unique to each party (party-specific features).
- **Label-availability heterogeneity:** Parties may also vary in the availability of labeled data. Some may have fully labeled datasets; others may have only partially labeled data or exclusively unlabeled data due to annotation costs or limited domain expertise.

These forms of heterogeneity complicate data collaboration including FL and DA because the observable information space differs across parties.

1.3 Related Works

Research related to this thesis spans several areas of collaborative learning, including FL under distribution heterogeneity, FL with heterogeneous features, FL with unlabeled data, and heterogeneous DA. Below, we summarize these research threads to clarify how the contributions of this thesis relate to existing work.

1.3.1 Federated Learning under Distribution Heterogeneity

FL was introduced as the Federated Averaging (FedAvg) algorithm [31], which averages locally trained models to obtain a single global model. However, FedAvg suffers from significant degradation under distribution heterogeneity [23, 30, 32].

To address this issue, two broad research directions have emerged. One focuses on improving the robustness of a shared global model through modifications to optimization or aggregation [5, 33, 34]. The other line of work develops *personalized federated learning* (PFL), which aims to produce party-specific models that adapt to local distributions using local adaptation, regularization, or clustering-based strategies [6, 7, 35–41]. Despite these advances, existing approaches still struggle to balance global sharing and local specialization under complex label-shift conditions.

Chapter 3 addresses this challenge by proposing a simple yet effective multi-branch architecture that explicitly captures both shared and party-specific patterns.

1.3.2 Federated Learning with Heterogeneous Features

Most early FL studies assume a shared input feature space across all parties. *Vertical federated learning* (VFL) and *federated transfer learning* (FTL) relax this assumption by supporting disjoint features with aligned sample IDs or correlated features [42–48]. However, these assumptions rarely hold in practical scenarios where organizations observe different features without shared samples or strong correlations.

Recent approaches attempt to extend standard FL to heterogeneous feature spaces by mapping data from each party into a shared latent representation before federated training [49–51]. Such methods, however, depend heavily on the existence of a meaningful latent space and may suffer from negative transfer when private features are highly diverse.

Chapter 4 tackles these limitations by explicitly separating common and party-specific feature components within a unified architecture, enabling effective collaboration without requiring feature alignment or feature similarity across parties.

1.3.3 Federated Learning with Unlabeled Data

FL with limited labeled data has been studied in the contexts of semi-supervised FL [52–58] and unsupervised FL focusing on representation learning or clustering [59–63]. These works typically assume at least partial supervision or similar data distributions across parties.

A more challenging scenario arises when all parties hold only unlabeled data from different domains (covariate shift), and a pretrained source model is available. The only prior work in this setting focuses on semantic segmentation and requires sharing feature-derived representations, raising privacy concerns [1].

Chapter 5 introduces the first federated framework for classification in this regime, enabling domain-aware and privacy-preserving collaboration by estimating party similarity solely from model parameters rather than data-derived features.

1.3.4 Heterogeneous Unsupervised Domain Adaptation

Conventional DA assumes matched feature spaces between source and target domains. *Heterogeneous unsupervised domain adaptation* (HUDA) relaxes this assumption by allowing distinct feature spaces and mainly addresses observation heterogeneity via feature-space mismatch and limited label availability [64–70]. However, most HUDA studies implicitly assume negligible distribution shift, which rarely holds in practice.

Chapter 6 considers a more realistic setting where severe label shift coexists with observation heterogeneity: the source domain has only positive examples, the target is fully unlabeled, and their feature spaces partially overlap. By integrating HUDA with positive-unlabeled learning [71], this chapter enables collaboration under both observation heterogeneity and significant label shift.

1.4 Contributions and Organization

This thesis investigates heterogeneous collaborative learning across multiple parties under a variety of practical constraints. Before addressing the different forms of heterogeneity, Chapter 2 introduces the unified problem setting and notation used throughout the thesis, providing a common foundation for the subsequent chapters. The four main research chapters then tackle different forms of data heterogeneity from complementary perspectives. Table 1.1 summarizes the learning frameworks and heterogeneity types considered in each chapter.

Table 1.1: Overview of learning frameworks and types of data heterogeneity addressed in each chapter.

Chapter	Learning Framework	Data Heterogeneity	
		Distribution	Observation
Chapter 3	federated learning	label shift	-
Chapter 4	federated learning	-	feature space
Chapter 5	federated learning	covariate shift	label availability
Chapter 6	domain adaptation	label shift	feature space, label availability

Chapters 3–5 focus on FL under different heterogeneous conditions: Chapter 3 studies distribution heterogeneity in the form of label shift; Chapter 4 examines observation heterogeneity arising from heterogeneous feature spaces; and Chapter 5 considers a challenging combination of covariate shift and the absence of labels when a pretrained source model is available. While Chapters 3–5 focus on FL settings where collaboration is constrained by privacy and decentralization, Chapter 6 deliberately steps outside the federated paradigm to examine collaborative learning under extreme heterogeneity. In this chapter, data can be centrally aggregated, enabling an explicit analysis of scenarios where multiple forms of heterogeneity co-exist simultaneously, including extreme label shift and heterogeneity in both label availability and feature space.

Together, these chapters provide a unified and systematic perspective on collaborative learning under heterogeneous data conditions and develop practical algorithms that enable effective collaboration.

The concrete contributions of this thesis are summarized as follows:

- **Chapter 3:** We propose a FL method tailored to label-shift scenarios, enabling parties with different label distributions to collaboratively train personalized models. A simple multi-branch architecture is introduced to balance shared global structure with party-specific specialization, providing robustness under realistic label-shift scenarios.
- **Chapter 4:** We address FL under observation heterogeneity, where parties observe different subsets of features. We propose a neural architecture that explicitly separates common and party-specific feature components, allowing each party to benefit from collaboration without requiring feature alignment or feature-similarity assumptions.
- **Chapter 5:** We study a fully unlabeled federated setting under covariate shift with a pretrained source model. A privacy-preserving method is developed

that refines pseudo-labels locally and computes domain similarity solely from shared model parameters and local unlabeled data, enabling domain-aware aggregation without exchanging data-derived representations.

- **Chapter 6:** We consider a two-party DA setting characterized by extreme heterogeneity, where the source domain contains only positive-labeled data, the target domain is entirely unlabeled, and the two domains have partially overlapping features. This setting is motivated by real-world applications, such as cross-industry potential customer prediction. We develop an algorithm that integrates HUDA with positive-unlabeled learning, enabling effective collaboration despite partial label availability and feature heterogeneity.

Chapter 2

Preliminaries

This chapter introduces the formal mathematical framework and optimization algorithms used throughout this thesis, including the general collaborative-learning formulation and the Federated Averaging (FedAvg) [31] algorithm.

2.1 Problem Setting

In this thesis, we focus on collaborative supervised learning. Before discussing heterogeneous scenarios, we introduce a heterogeneity-agnostic general problem setting that defines the fundamental components common to all learning tasks addressed in this thesis.

2.1.1 General Formulation

We consider K parties indexed by $k \in \{1, \dots, K\}$ (or $k \in \{s, t\}$ in a two-party DA setting). Each party owns a local dataset $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$ of N_k samples drawn from its local distribution $P_k(x, y)$. Let N be the total number of data samples across clients. Each sample consists of an input feature vector $x_{k,i} \in \mathcal{X}_k$ and an associated label $y_{k,i} \in \mathcal{Y}$, where $\mathcal{X}_k$ denotes a party-specific feature space and $\mathcal{Y}$ denotes a shared label space. Throughout this thesis, we consider M -class classification problem. Each party trains a predictive model (neural network) $h_{\theta_k}: \mathcal{X}_k \rightarrow \mathcal{Y}$, parametrized by $\theta_k \in \Theta$. The local supervised loss for party k is defined as

$$\mathcal{L}_k(\theta_k) = \mathbb{E}_{(x,y) \sim P_k}[\ell(h_{\theta_k}(x), y)], \quad (2.1)$$

where $\ell(\cdot, \cdot)$ is a task-specific loss such as the cross-entropy loss. In practice, the empirical loss on dataset D_k is used:

$$\widehat{\mathcal{L}}(D_k; \theta_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell(h_{\theta_k}(x_{k,i}), y_{k,i}). \quad (2.2)$$

Collaborative learning commonly considers two objectives.

Learning a Single Global Model: A single model shared by all parties is obtained by solving

$$\min_{\theta} \sum_{k=1}^K \mathcal{L}_k(\theta). \quad (2.3)$$

Standard FL approaches such as FedAvg [31] optimize this objective by exchanging model parameters while keeping all raw data local. We introduce the detailed algorithm of FedAvg in Section 2.2. In DA settings where data from source and target domains can be centrally collected (Chapter 6), the same objective is optimized to learn a target-effective model, possibly with additional domain regularization.

Learning Multiple Personalized Models: When parties exhibit substantial differences in data characteristics, it may be preferable to learn models specialized to each party while still leveraging shared information:

$$\min_{\theta_1, \dots, \theta_K} \sum_{k=1}^K \mathcal{L}_k(\theta_k). \quad (2.4)$$

This objective captures a variety of personalized FL approaches addressed in most chapters in this thesis (Chapter 3,4,5).

We next formalize the types of heterogeneity that later chapters build upon.

2.1.2 Data Heterogeneity

We consider two primary forms of heterogeneity: distribution heterogeneity and observation heterogeneity.

2.1.2.1 Distribution Heterogeneity

Distribution heterogeneity occurs when parties have different data distributions:

$$P_k(x, y) \neq P_{k'}(x, y). \quad (2.5)$$

Decomposing the joint distribution as

$$P_k(x, y) = P_k(x) P_k(y|x) = P_k(y) P_k(x|y) \quad (2.6)$$

allows us to characterize two types of distributional discrepancies.

Covariate Shift (Chapters 5): Parties share the same conditional distribution $P_k(y|x)$ but differ in the marginal $P_k(x)$.

Label Shift (Chapters 3, 6): Parties share the same class-conditional distribution $P_k(x|y)$ but differ in the marginal $P_k(y)$.

The degree of distribution heterogeneity can be quantified by the Kullback–Leibler (KL) divergence $D_{\text{KL}}(\cdot\|\cdot)$ between party-wise distributions P_k . Chapters 3 and 6 focus on label-shift scenarios in which the supports of $P_k(y)$ differ across clients, resulting in $D_{\text{KL}}(P_k\|P_{k'}) = \infty$. Chapter 3 further considers a milder Dirichlet-based label-shift setting with $D_{\text{KL}}(P_k\|P_{k'}) < \infty$. In contrast, Chapter 5 addresses covariate shift, where analytical computation of $D_{\text{KL}}(\cdot\|\cdot)$ is generally intractable for high-dimensional image data, and instead examines heterogeneous visual domains such as art paintings and photos.

2.1.2.2 Observation Heterogeneity

Observation heterogeneity concerns differences in the information available to each party.

Feature-Space Heterogeneity (Chapters 4, 6): Parties observe different feature spaces $\mathcal{X}_k \neq \mathcal{X}_{k'}$. A common formulation decomposes each feature space as

$$\mathcal{X}_k = \mathcal{X}^c \times \mathcal{X}_k^u,$$

where $\mathcal{X}^c$ represents features shared across all parties and $\mathcal{X}_k^u$ represents party-specific (unique) features.

Label-Availability Heterogeneity (Chapters 5, 6): Parties may differ in the amount of supervision: some have labeled datasets $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$, while others have only unlabeled data $D_{k'} = \{x_{k',i}\}_{i=1}^{N_{k'}}$ due to annotation costs or lack of expertise.

2.2 Federated Averaging (FedAvg)

FL is typically organized in a *server-client* architecture, where each party acts as a client performing local optimization, and a central server coordinates the overall training process. FedAvg [31] is the most standard method for iteratively optimizing the global objective in such a distributed setting while keeping all data local. The detailed algorithm is shown in Algorithm 1.

At communication round r , the central server sends the current global model parameters θ^r to all participated clients. Each client k performs E local epochs to update θ^r and obtain its local model θ_k^r by minimizing the empirical loss $\widehat{\mathcal{L}}(D_k; \theta_k)$ via gradient descent:

$$\theta_k^r \leftarrow \theta_k^r - \eta \nabla \widehat{\mathcal{L}}(D_k; \theta_k^r), \quad (2.7)$$

where η is the learning rate. After completing the local updates, client k returns θ_k^r to the server, which averages the local models to update the global parameters:

$$\theta^{r+1} \leftarrow \sum_{k=1}^K \frac{N_k}{N} \theta_k^r. \quad (2.8)$$

This procedure is iterated for R iterations, and the final global model θ^R is then used for inference. However, such a single global model does not necessarily optimize performance for the heterogeneous data distributions of individual clients.

The generalization behavior of FedAvg has been theoretically studied. Recent stability-based analyses show that the generalization bound of FedAvg scales as $\mathcal{O}(\frac{1}{N})$ with the total number of samples across clients [72]. This result suggests that increasing the number of participating clients can reduce the generalization error, providing theoretical justification for the benefits of data collaboration discussed in Chapter 1.

Algorithm 1: FedAvg: Federated Averaging [31]

Input: K clients with local datasets $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$, initial parameters θ^0 , number of communication rounds R , number of local epochs E , learning rate η , size of sampled clients S

Output: Trained parameters θ^R

```
1 Server executes:
2 for  $r = 0$  to  $R - 1$  do
3   Sample a subset of clients  $\mathcal{S}^r$  of size  $S$ 
4   for each  $k \in \mathcal{S}^r$  in parallel do
5     Send  $\theta^r$  to client  $k$ 
6      $\theta_k^r \leftarrow \text{ClientLocalLearning}(k, \theta^r)$ 
7    $\theta^{r+1} \leftarrow \sum_{k=1}^K \frac{N_k}{N} \theta_k^r$ 
8 return  $\theta^R$ 

9 ClientLocalLearning( $k, \theta^r$ ):
10   $\theta_k^r \leftarrow \theta^r$ 
11  for  $e = 1$  to  $E$  do
12     $\theta_k^r \leftarrow \theta_k^r - \eta \nabla \hat{\mathcal{L}}(D_k; \theta_k^r)$ 
13  return  $\theta_k^r$ 
```

Chapter 3

Personalized Federated Learning with Multi-branch Architecture

3.1 Introduction

As discussed in Chapters 1, 2, federated learning (FL) enables multiple clients to collaborate without sharing data, but its effectiveness is significantly affected by *distribution heterogeneity*. In particular, *label shift*, a key form of distribution heterogeneity defined in Chapter 2, makes a single global model suboptimal, thereby motivating the need for *personalized federated learning* (PFL), which aims to learn client-specific models that better capture local data characteristics.

Existing PFL methods can be broadly categorized into two groups based on how personalization is achieved: local-customization methods and similarity-based methods [6]. Local-customization methods train a global model and locally customize it to build a personalized model. Local fine-tuning is a representative example [35]. However, these methods usually construct the global model with equal contributions from all clients, even when some clients' data distributions differ substantially. This can lead to suboptimal or even invalid global updates.

To mitigate this issue, similarity-based PFL methods have been proposed. Early approaches in this category rely on client clustering [7, 40, 73, 74], training a shared model for each cluster of similar clients. Such methods are only applicable when clients are clearly partitioned according to their distributions and not suitable for more complex data distributions. More recent methods, such as FedAMP [6] and FedFomo [41], compute pairwise client similarities and update each client's model as a weighted combination of other clients' models with the similarities. Although

these methods outperform local-customization approaches in simple label-shift settings, e.g., when each client has a small subset of classes or clients are grouped in advance by their data distributions, our experiments show that FedAMP, FedFomo, and clustering-based methods perform poorly in more complex label-shift scenarios. This is because directly estimating inter-client similarities is difficult due to the complexity of data distribution.

In this chapter, we aim to overcome this limitation of similarity-based methods by extending the core idea behind clustering-based methods. Specifically, clustering-based approaches restrict each client to a single cluster model, but in complex label-shift settings where clear clustering does not exist, clients should ideally leverage weighted combinations of multiple models. Motivated by this insight, we propose a new PFL method called *personalized Federated learning with Multi-Branch architecture* (pFedMB). pFedMB decomposes each neural network layer into multiple branches and assigns a set of client-specific coefficients to these branches. Each client obtains a personalized model by learning optimized client-specific coefficients and combining all the branches via a convex weighting to form its effective layer. Furthermore, pFedMB introduces a novel aggregation mechanism that updates each branch through weighted averaging, using the client-specific coefficients associated with that branch. This design enables clients with similar data distributions to implicitly share more information without explicitly computing similarity measures as in FedAMP or FedFomo since similar clients naturally learn similar client-specific coefficients. As a consequence, the method enhances both communication efficiency and model performance.

Our empirical results demonstrate that pFedMB consistently outperforms state-of-the-art PFL methods across a range of benchmarks in label shift settings and exhibits strong robustness to both simple and highly complex label-shift settings, making it a well-balanced and widely applicable PFL solution.

3.2 Related Works

3.2.1 Multi-branch Neural Networks

Our research is inspired by multi-branch convolutional networks such as conditionally parameterized convolutions (CondConv) [75], which enables us to increase the size and capacity of a network, while maintaining efficient inference. In a CondConv layer, a convolutional kernel for each example is computed as a weighted

linear combination of multiple branches by example-dependent coefficients. We extend this architecture to the FL setting, where the coefficients associated to each branch depend on clients’ data distributions, rather than examples.

Other type of multi-branch neural networks has also been proposed [76, 77]. BranchyNet [76] inserts early exit branches, that is, side branches in the middle of the neural network and achieves fast inference by allowing test samples to exit the network early via these branches when samples can already be inferred with high confidence.

3.2.2 Federated Learning Based on Multi-branch Neural Networks

Several studies have incorporated multi-branch neural network architectures into FL [78–82]. Broadly, these methods use branching to adapt model capacity or structure across clients, and are often motivated by practical heterogeneity in client resources or operating conditions. In particular, a common use case is heterogeneous FL with varying computation capacities, where the model is structured so that different clients can train and/or deploy different portions of the network [78–81]. Another line of work considers dynamic environments, where specialized branches help accommodate changing data conditions over time [82].

Our method differs from these approaches in two important aspects. First, the proposed multi-branch architecture is structurally distinct. Our method splits each layer into multiple branches and computes their weighted linear combination, similar in spirit to CondConv [75]. In contrast, existing works [78–81] adopt a multi-exit architecture inspired by BranchyNet [76], which does not allow branch superposition. FedTEM [82] is closely related to our method but branches only at the final layer and does not support branch superposition. Second, unlike all the above methods, our approach is explicitly designed for PFL. To the best of our knowledge, this is the first method to employ a multi-branch neural network architecture specifically for personalization in FL.

3.3 Problem Formulation

We consider a FL setting in which K clients each hold private labeled datasets drawn from different label distributions (see Section 2.1.1). Our goal is to learn personalized models $\{\theta_1, \dots, \theta_K\}$ by the PFL objective (2.4) without assuming grouping structure

or prior knowledge of similarities among clients, which is precisely the challenge addressed by the proposed pFedMB method.

In this chapter, we denote learning model parameters of neural networks as $\theta = (W_1, \dots, W_L)$, where each W_l represents the weight matrix of the l -th layer. Bias vectors are omitted for simplicity.

3.4 Proposed Method

In this section, we provide details of our proposed method, *personalized Federated learning with Multi-Branch architecture* (pFedMB).

3.4.1 Architecture

We first introduce the architecture of pFedMB. Our method splits each linear layer of a neural network into B branches, similar to CondConv [75]. Specifically, W_l is represented as

$$W_l = \alpha_{1,l}W_{1,l} + \dots + \alpha_{B,l}W_{B,l}, \quad (3.1)$$

where $W_{b,l}$ is the weight matrix of the b -th branch, and $\alpha_{b,l}$ is its non-negative coefficient satisfying $\sum_{b=1}^B \alpha_{b,l} = 1$. In the multi-branch layer, each branch receives the same input x and outputs $W_{b,l} * x$, and then the outputs of all the branches are linearly combined, weighted by $\alpha_{b,l}$. This is mathematically equivalent to the operation to multiply x by W_l :

$$W_l * x = \alpha_{1,l}(W_{1,l} * x) + \dots + \alpha_{B,l}(W_{B,l} * x). \quad (3.2)$$

Therefore, expressing W_l as above shows that the multi-branch layer is as computationally efficient as a standard layer during inference. The multi-branch architecture is illustrated in Fig. 3.1.

In pFedMB, all clients use the same multi-branch architecture. While CondConv determines branch coefficients as functions of the input x , we adapt this idea to the PFL setting by making the branch coefficients client-dependent, reflecting each client's data distribution, rather than input-dependent. To obtain personalized models, the coefficients $\{\alpha_{b,l}\}_{b,l}$ are decoupled from globally shared parameters as in FedPer [83], allowing each client to optimize its own coefficients locally and obtain a model personalized to their data distribution. Unlike FedPer, which decouples the parameters of entire layers, pFedMB shares knowledge across clients through the

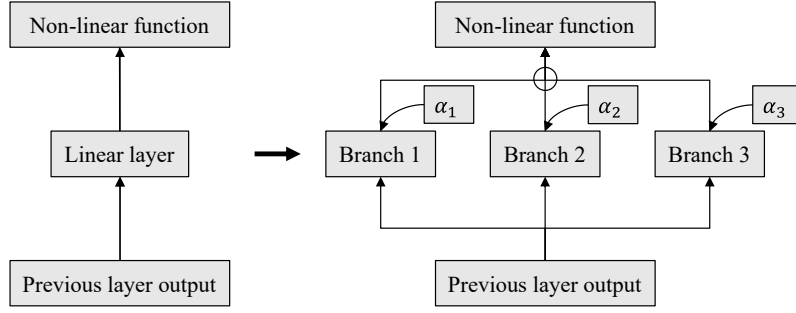


Figure 3.1: Architecture of multi-branch layer.

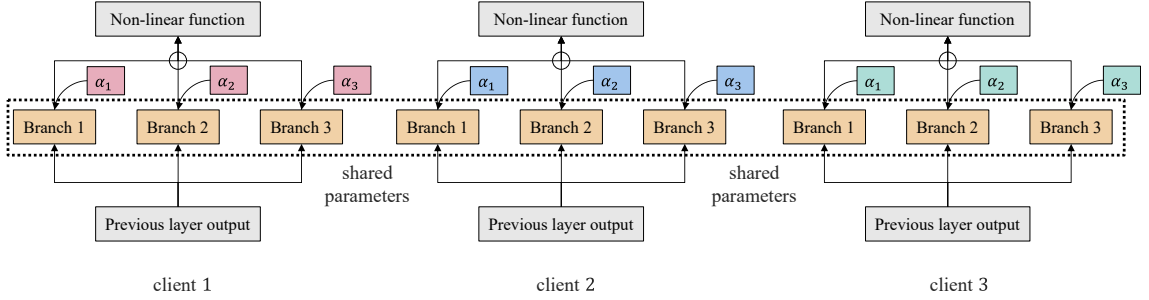


Figure 3.2: Architecture of pFedMB.

branch parameters and personalizes only the branch coefficients, thereby reducing overfitting on client-specific data. The pFedMB architecture is shown in Fig. 3.2.

3.4.2 Algorithm

We review the algorithm of pFedMB in detail. Algorithm 2 shows its complete procedure. We denote the model parameters of each client k by $\theta_k = (W^k, \alpha^k) = (\{W_{l,b}^k\}_{l,b}, \{\alpha_{l,b}^k\}_{l,b})$. The algorithm follows the general structure of standard FL methods such as FedAvg [31]: the server updates the global model at each communication round, while clients perform local training between rounds. However, pFedMB introduces distinct server and client-side modifications, which we outline below.

At each communication round r , the server distributes the global parameters $W^r = \{W_{l,b}^r\}$ to all clients. The local update phase consists of two steps: updating α and updating W . First, each client k updates their client-specific coefficients $\alpha^{r,k}$ for E local epochs via gradient descent while keeping W^r fixed:

$$\alpha^{r,k} \leftarrow \alpha^{r,k} - \eta_\alpha \nabla_{\alpha^{r,k}} \widehat{\mathcal{L}}(D_k; W^r, \alpha^{r,k}), \quad (3.3)$$

Algorithm 2: pFedMB: Personalized Federated Learning with Multi-Branch Architecture

Input: K clients with local datasets $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$, initial model parameters W^0 and $\{\alpha^{0,k}\}_k$, number of communication rounds R ; number of local epochs E , learning rates η_α and η_W ; size of sampled clients S , number of layers L and number of branches B

Output: N personalized models $\theta_k^R = (W^R, \alpha^{R,k})$

1 **Server executes:**

2 **for** $r = 0$ **to** $R - 1$ **do**

3 Send W^r to all clients

4 Sample a subset of clients $\mathcal{S}_r$ with size S

5 **for each** $k \in \mathcal{S}_r$ **in parallel do**

6 $(W^{r,k}, \alpha^{r,k}) \leftarrow \text{ClientLocalLearning}(k, r, W^r)$

7 **for** $l = 1$ **to** L **do**

8 **for** $b = 1$ **to** B **do**

9 $W_{l,b}^{r+1} \leftarrow \frac{\sum_{k=1}^K N_k \alpha_{l,b}^{r,k} W_{l,b}^{r,k}}{\sum_{k'=1}^K N_{k'} \alpha_{l,b}^{r,k'}}$

10 **return** W^R

11 **ClientLocalLearning**(k, r, W^r):

12 $W^{r,k} \leftarrow W^r$

13 Update $\alpha^{r,k}$ for E epochs via Eq. (3.3)

14 Update $W^{r,k}$ for E epochs via Eq. (3.4)

15 $\alpha^{r+1,k} \leftarrow \alpha^{r,k}$

16 **return** $W^{r,k}, \alpha^{r,k}$

where $\hat{\mathcal{L}}$ is the empirical loss function defined by Eq. (2.2). This step enables each client to adjust the branch-wise coefficients to better match their local data distribution. Next, client k fixes $\alpha^{r,k}$ and updates W^r for E local epochs to obtain the local version $W^{r,k}$:

$$W^{r,k} \leftarrow W^{r,k} - \eta_W \nabla_{W^{r,k}} \hat{\mathcal{L}}(D_k; W^{r,k}, \alpha^{r,k}). \quad (3.4)$$

After local training, each client sends $(\alpha^{r,k}, W^{r,k})$ to the server. The server then aggregates the weights to form the next global model via an α -weighted aggregation scheme:

$$W_{l,b}^{r+1} = \frac{\sum_{k=1}^K N_k \alpha_{l,b}^{r,k} W_{l,b}^{r,k}}{\sum_{k'=1}^K N_{k'} \alpha_{l,b}^{r,k'}}. \quad (3.5)$$

This aggregation weights each branch not only by the number of data points N_k

but also by the corresponding client-specific coefficient $\alpha_{l,b}^{r,i}$, allowing clients that rely more on branch b of layer l to have a stronger contribution to $W_{l,b}^{r+1}$. Consequently, clients with similar α values collaborate more closely, as in FedFomo [41] and FedAmp [6]. Unlike these methods, however, pFedMB induces such collaboration automatically because each client optimizes only its branch-wise coefficients, without explicitly computing pairwise similarities.

3.5 Experiments

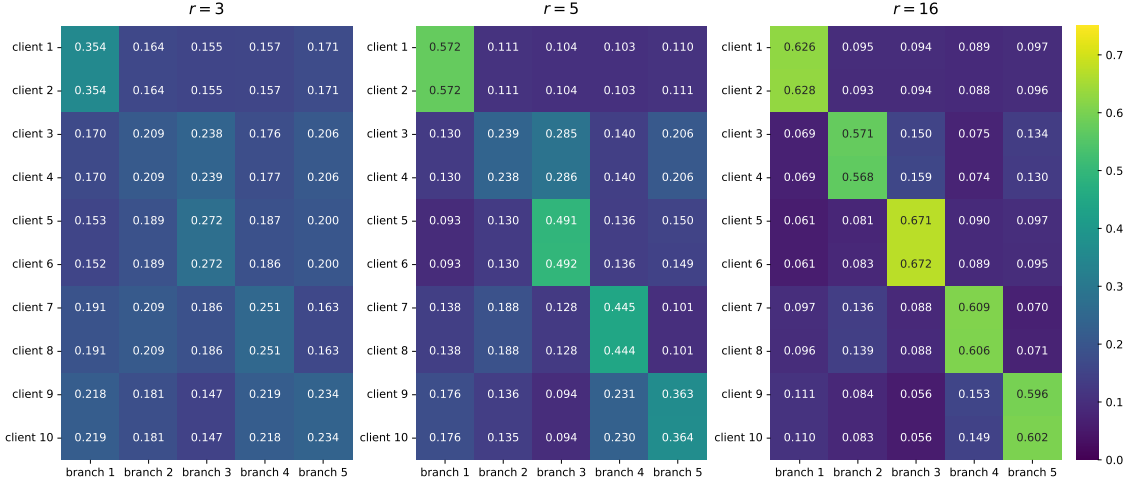
In this section, we evaluate the performance of pFedMB in label-shift settings. First, we observe how pFedMB achieves personalization in a simple setting. Next, we compare the performance of pFedMB with the state-of-the-art PFL methods in more complex settings.

3.5.1 Experimental Setup

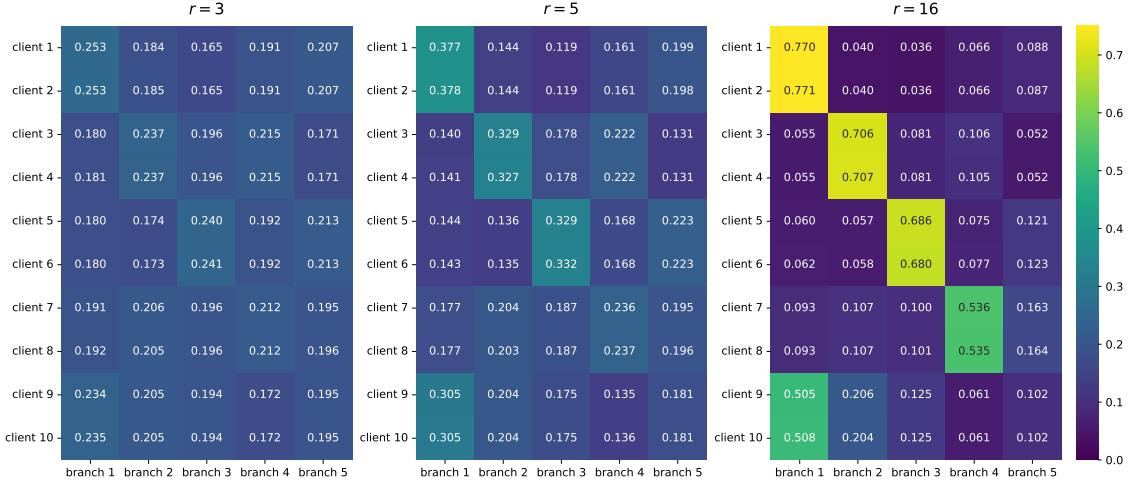
Datasets In this chapter, we numerically evaluate the performance of pFedMB using CIFAR10 and CIFAR100 [84]. CIFAR10/CIFAR100 dataset has 50,000 training images and 10,000 test images with 10/100 classes. We split the training images into 40,000 training images and 10,000 validation images used to tune the hyperparameters for each method (e.g. learning rate η_α, η_W and number of branches for pFedMB). Each type of data (i.e., training, validation, and test) is distributed across clients in the same way.

Data heterogeneity For CIFAR10, we consider two label-shift settings. The first is a commonly used scenario where each client is randomly assigned 2 classes with the same data size. Since the label supports differ across clients, the KL divergence between client-wise label distributions is infinite, i.e., $D_{\text{KL}}(P_k \| P_{k'}) = \infty$. However, this setting is overly simplistic and does not reflect realistic data distributions. To model a milder and more realistic label shift, we also adopt a Dirichlet-based partition with concentration parameter $\beta = 0.4$ [34, 85, 86], where all clients share the same label support and the KL divergence is finite. Specifically, when M -class label proportions are independently sampled from a symmetric Dirichlet distribution, the expected KL divergence between two clients is $\mathbb{E}[D_{\text{KL}}(P_k \| P_{k'})] = \frac{M-1}{M\beta}$, whose derivation is provided in the Appendix 3.8.

For CIFAR100, applying a Dirichlet partition would result in too few samples



(a) pFedMB with α -weighted averaging.



(b) pFedMB without α -weighted averaging.

Figure 3.3: Visualization of α at communication round r with/without α -weighted averaging using CIFAR10.

per class. We therefore follow [37, 87], where data is heterogeneous in terms of not only classes but also data size. First, we randomly assign 20 classes for each client, yielding $D_{\text{KL}}(P_k(y)||P_{k'}(y)) = \infty$. Next, for each client k and chosen class c , we sample $u_{k,c} \in U(0.3, 0.7)$ (uniform distribution between 0.3 and 0.7) and allocate $\frac{u_{k,c}}{\sum_{k'} u_{k',c}}$ of the samples for class c to client k .

Implementation details The performance of all the methods is evaluated by mean test accuracy among clients. Moreover, the experiments were conducted three times and the average of the test accuracies from the three experiments are reported. In each method, we train the models until the convergence. The number of clients

is set to 15 and 50 in the main results and the client participation rate at each communication round is 100% and 20%, respectively. In all the experiments, we use a CNN network, which has two 5x5 convolution layers followed by 2x2 max pooling (the first with 6 channels and the second with 16 channels) and two fully connected layers with ReLU activation (the first with 120 units and the second with 84 units). pFedMB extends each linear layer to a multi-branch layer. The initial coefficients α for each client are set to the same value for all branches. Throughout the experiments, the number of local epochs and batch size are set to 5 and 64, respectively. All methods are implemented in PyTorch.

Compared methods We compare pFedMB with other various methods classified into two categories. The first category is the improved FL approach to train a single global model that performs well under label shift. In this category, we use FedAvg [31] and FedProx [5]. The second category is PFL to train multiple models personalized to each client’s data distribution. We compare our method with non-federated local learning, FedPer [83], pFedMe [37], IFCA [7], FedFomo [41], and FedAMP [6]. In all the methods, we perform local fine-tuning for local epochs. Therefore, in our experiments, FedAvg and FedProx also eventually produce multiple models.

3.5.2 Process of Personalization

Before moving on to the main result, we will see how our method works in a simple setting. We use CIFAR10 and consider 10 clients grouped into 5 pairs, each with the same two classes for simplicity (e.g. clients 1, 2 have class 0 and 1). Furthermore, we suppose client-specific coefficients α are common for all the layers. In this experiment, the number of branches is set to 5. We compare pFedMB with local learning and FedAvg as baselines. Moreover, in order to see the effect of an aggregation method using weighted averaging by α (3.5) in global model updating phase, we implement pFedMB that updates global models by a standard averaging as in FedAvg:

$$W_{l,b}^{r+1} = \frac{\sum_{k=1}^K N_k W_{l,b}^{r,k}}{\sum_{k'=1}^N N_{k'}}. \quad (3.6)$$

Table 3.1 shows the mean test accuracy for each method. First, from Table 3.1, we can see that pFedMB improves FedAvg and α -weighted averaging is effective in the perspective of accuracy.

Table 3.1: The mean test accuracy (%) among 10 clients grouped into 5 pairs using CIFAR10. Each method includes local fine-tuning for local epochs E .

Local	FedAvg	pFedMB (w/o α -weighted avg.)	pFedMB
87.26	87.99	89.08	89.49

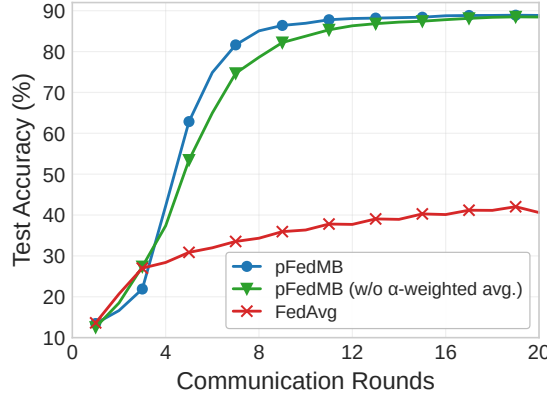


Figure 3.4: The mean test accuracy among 10 clients grouped into 5 pairs at each communication round using CIFAR10. Each method will be followed by local fine-tuning.

Next, the process of getting the optimal α for each client is visualized in Fig. 3.3a, 3.3b, each of which is the result of pFedMB with/without α -weighted averaging, respectively. Fig. 3.3a shows that every pair of clients who have the same classes gets almost the same α and different pairs of clients focus on different branches. From this result, we can see that pFedMB with α -weighted averaging achieves personalization by facilitating similar clients to collaborate more. In contrast, in pFedMB without α -weighted averaging, the clients are clustered slowly, which results in different pairs of clients concentrating on the same branch. This leads to inefficient collaborating. Fig. 3.4 shows that α -weighted averaging is also effective in terms of communication efficiency. Fig. 3.4 presents the test accuracy at each communication round. pFedMB with α -weighted averaging is clearly faster to convergence than pFedMB without α -weighted averaging as well as FedAvg, that is, reduces communication cost. Therefore, it is revealed that α -weighted averaging (3.5) is effective to promote collaboration among similar clients, which leads to good performance for each client and fast convergence.

Table 3.2: The mean test accuracy (%) among 15 clients and 50 clients in random 2 classes and Dirichlet distribution settings using CIFAR10. Client participation rate at each communication round is 100% and 20%, respectively. Each method includes local fine-tuning for local epochs E . “Local” is a non-federated local learning.

Methods	CIFAR10				CIFAR100	
	Random 2 classes		Dirichlet		Random 20 classes with different size	
	15 clients	50 clients	15 clients	50 clients	15 clients	50 clients
Local	90.01	83.79	67.54	61.27	39.10	29.42
FedAvg [31]	89.69	85.68	73.52	70.32	43.49	38.53
FedProx [5]	90.51	86.07	73.38	70.98	44.81	36.07
FedPer [83]	89.73	84.56	71.97	67.57	41.52	29.67
pFedMe [37]	90.04	83.09	72.92	71.00	45.57	39.04
IFCA [7]	90.17	85.68	72.42	69.84	41.46	35.01
FedFomo [41]	90.12	85.01	67.55	62.62	38.19	28.18
FedAmp [6]	90.73	85.90	72.91	65.98	41.27	29.53
pFedMB (ours)	90.69	86.73	73.53	72.14	44.73	39.78

3.5.3 Performance Comparison

Table 3.2 shows the performance comparison of pFedMB and the other PFL methods in six settings. We can see that pFedMB is the best in almost all the settings. Even in 15 clients with random 2 classes setting using CIFAR10 and 15 clients setting using CIFAR100, pFedMB is comparable to the best methods FedAMP and pFedMe, respectively. We note that FedAvg and FedProx, originally not personalized FL methods also perform well especially in random 2 classes distribution setting using CIFAR10. It is because in our experiments, all the methods include local fine-tuning. That is, local fine-tuning has enough personalization power especially in a simple setting where each client has only 2 classes, which is a fact not often pointed out in the previous studies. It is also worth noting that the state-of-the-art methods such as FedAMP and FedFomo do not perform well except in random 2 classes distribution setting using CIFAR10. As mentioned in the introduction, this is probably because the similarity calculation between clients does not work well when each client’s data is sampled from a complex distribution such as Dirichlet distribution and random 20 classes with different size. In contrast, pFedMB, which does not directly calculate the similarity, shows good performance in both distribution settings. In other words, pFedMB is the most balanced method that is effective in every setting.

Table 3.3: Tuned number of branches of pFedMB in each experimental setting.

Settings			# of branches
CIFAR10	Random 2 classes	15 clients	6
		50 clients	10
	Dirichlet	15 clients	7
		50 clients	6
CIFAR100	Random 20 classes with different size	15 clients	3
		50 clients	3

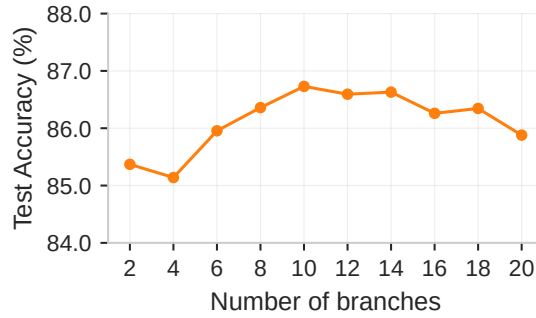


Figure 3.5: Variation of the mean test accuracy among 50 clients in the random 2 classes distribution setting using CIFAR10 with respect to the number of branches.

3.5.4 Effects of Number of Branches

We next investigate the effects of number of branches on the performance of pFedMB. Table 3.3 shows tuned number of branches of pFedMB in each experimental setting. From Table 3.3, we can see that the required number of branches is at most 10. Unlike CIFAR10, CIFAR100 has a larger number of classes, and therefore the number of classes shared among clients when using CIFAR100 is larger, which results in fewer branches. Therefore, in the random 2 classes and 50 clients setting using CIFAR10, where the required branches is large, we see the effect of number of branches. In Fig. 3.5, the number of branches of pFedMB is changed from 2 to 20. The test accuracy increases with the number of branches, reaching a peak at 10 and then decreasing with the number of branches. Therefore, when using pFedMB, it is necessary to find the peak number of branches by fine-tuning.

3.6 Discussion

We first discuss the communication and computation costs of pFedMB. The communication and computation overheads are determined by the number of branches B . Let P be the number of model parameters used in the traditional FL methods, e.g., FedAvg. If pFedMB splits every layer into B branches, the number of model parameters increases to $B \times P$. Therefore, the communication and computation costs of pFedMB are B times those of FedAvg. Those costs can be reduced by limiting the number of layers to be branched or by stopping both training and communicating the branches that are assigned small client-specific coefficients per client (because these branches are not important to the corresponding client). As mentioned in Section 3.4.1, the computation cost of pFedMB during inference is the same as FedAvg.

Given the communication and computation bottlenecks of pFedMB, the question arises how many branches are needed. The obvious upper limit is the number of clients. This is because if the number of branches is greater than the number of clients, there is no point in collaboration among clients, and local learning is sufficient. If the clients are clustered into some groups, as in the experiment of Section 3.5.2, then B should be set to the number of groups. Also, if the data distribution of each client is a mixture of several distributions, B should be set to the number of underlying distributions. If the exact number cannot be specified in advance, B must be treated as a hyperparameter. In our experiments, the number of branches did not exceed 10 as a result of tuning in any setting.

We finally discuss the privacy issues for each client. Unlike FedFomo and FedAMP, pFedMB shares the clients' local models only with the central server, not with other clients or cloud server introduced in FedAMP. This is the benefit of pFedMB. However, as in clustering-based methods, there might be somewhat privacy concern for pFedMB when sending client-specific coefficients α^k to the server because α^k contains the information about data distribution for client k . Note that α^k does not contain the information about raw data. In real-world systems, we suggest using tools such as secure multiparty computation when conducting α -weighted aggregation if needed.

3.7 Summary

We proposed a personalized federated learning (PFL) method, pFedMB, that achieves personalization by splitting each neural network layer into multiple branches and assigning client-specific coefficients to them. We also introduced an efficient aggregation scheme that accelerates convergence and improves model performance. Numerical experiments show that pFedMB outperforms state-of-the-art PFL methods across both simple and complex data distributions, demonstrating its effectiveness and balanced performance.

3.8 Appendix

Expected KL divergence between two symmetric Dirichlet samples The Dirichlet distribution is a distribution over probability vectors on the $(M-1)$ -simplex. In this chapter, we use the symmetric Dirichlet distribution $\text{Dir}(\beta \mathbf{1}_M)$, parameterized by a single concentration parameter $\beta > 0$, where $\mathbf{1}_M \in \mathbb{R}^M$ denotes the M -dimensional all-ones vector.

Let $P = (P_1, \dots, P_M)$ and $Q = (Q_1, \dots, Q_M)$ be independent samples from $\text{Dir}(\beta \mathbf{1}_M)$:

$$P \sim \text{Dir}(\beta \mathbf{1}_M), \quad Q \sim \text{Dir}(\beta \mathbf{1}_M).$$

We derive a closed form for the expected Kullback–Leibler divergence $\mathbb{E}[D_{\text{KL}}(P||Q)]$.

Taking expectations and using the independence of P and Q , we obtain

$$\begin{aligned} \mathbb{E}[D_{\text{KL}}(P||Q)] &= \sum_{m=1}^M \mathbb{E}[P_m \log P_m] - \sum_{m=1}^M \mathbb{E}[P_m \log Q_m] \\ &= \sum_{m=1}^M \mathbb{E}[P_m \log P_m] - \sum_{m=1}^M \mathbb{E}[P_m] \mathbb{E}[\log Q_m]. \end{aligned} \quad (3.7)$$

Since P and Q are identically distributed, $\mathbb{E}[\log Q_m] = \mathbb{E}[\log P_m]$. The required moments are given in terms of the digamma function $\psi(\cdot)$:

$$\begin{aligned} \mathbb{E}[P_m] &= \frac{1}{M}, \\ \mathbb{E}[\log P_m] &= \psi(\beta) - \psi(M\beta), \\ \mathbb{E}[P_m \log P_m] &= \frac{1}{M} \left(\psi(\beta + 1) - \psi(M\beta + 1) \right). \end{aligned}$$

Substituting these expressions into Eq. (3.7) yields

$$\begin{aligned}\mathbb{E}[D_{\text{KL}}(P\|Q)] &= M \left(\frac{1}{M} (\psi(\beta + 1) - \psi(M\beta + 1)) - \frac{1}{M} (\psi(\beta) - \psi(M\beta)) \right) \\ &= (\psi(\beta + 1) - \psi(\beta)) - (\psi(M\beta + 1) - \psi(M\beta)).\end{aligned}\tag{3.8}$$

Using the digamma recurrence relation $\psi(x + 1) - \psi(x) = 1/x$ for $x > 0$, Eq. (3.8) simplifies to

$$\mathbb{E}[D_{\text{KL}}(P\|Q)] = \frac{1}{\beta} - \frac{1}{M\beta} = \frac{M - 1}{M\beta}.\tag{3.9}$$

Chapter 4

Federated Learning Enhanced by Continual Learning for Common and Uncommon Features

4.1 Introduction

Federated learning (FL) traditionally operates in a setting known as *horizontal federated learning* (HFL), where clients hold data with the same feature space but different samples. This allows clients to collaboratively train a model by sharing model updates rather than raw data. However, as discussed in Chapter 1, many real-world applications exhibit *observation heterogeneity*, and in particular, clients often share only a subset of their features. For example, hospitals, nursing homes, and physical examination centers may share common patient features (e.g., age, blood pressure) but also maintain institution-specific features such as prescription or dietary information [49]. In such cases, applying standard HFL effectively reduces the feature space to the intersection of all clients' features, discarding valuable client-specific information.

Several FL frameworks have attempted to move beyond this limitation. Vertical FL (VFL) and federated transfer learning (FTL) utilize all client features but typically rely on overlapping samples or explicit feature correspondences, which are often unavailable and may introduce privacy risks [88]. More recent approach, *heterogeneous federated transfer learning* (HFTL) [49], assumes feature co-occurrence and reconstructs missing features using feature-mapping models before applying HFL on the complete feature set. HFTL, however, make strong assumptions about

feature correlations, are limited to logistic regression, or become impractical for neural networks because exchanging feature-mapping models requires multi-party computation [89].

To address these challenges, this chapter introduces *Federated Learning Enhanced by Continual learning for common and uncommon features* (FLEC), a neural-network-based approach designed specifically for heterogeneous feature spaces without requiring overlapping samples or strong alignment assumptions. FLEC separates each client’s features into common and unique subsets, trains a global model on common features through standard HFL, and simultaneously trains client-specific networks on unique features. Using a continual learning (CL) [90] strategy inspired by progressive neural networks (PNNs) [91], FLEC integrates global and local representations in a way that facilitates knowledge transfer while avoiding interference with global training. This design enables local inference using each client’s full feature set without reconstructing missing features or sharing sensitive information.

Experiments on five real-world datasets show that FLEC outperforms four baselines: HFL using only common features, local learning, and HFL in which other clients’ unique features are imputed with zeros or mean values.

4.2 Related Work

4.2.1 Horizontal Federated Learning under Feature-Space Heterogeneity

Several works extend HFL to heterogeneous feature spaces. Gao et al. [49] assume the existence of common features and propose a method in which clients mutually complement missing features to align their feature spaces before performing standard HFL. However, their method is applicable only to logistic regression and implicitly assumes sufficient correlation between the common and unique features across clients.

Gao et al. [50] also propose a method in which each client’s data with heterogeneous feature spaces is mapped into a common feature space in the early layers of the model, after which standard HFL is applied in the later layers. Rakotomamonjy et al. [51] similarly map each client’s data into a common feature space and then apply personalized FL. However, these methods assume that clients possess sufficiently similar data such as images or EEG recordings that can be mapped into a shared feature space, and they focus on discussing the transferability in such set-

tings. Therefore, in general cases, negative transfer can occur and degrade accuracy.

In contrast, our study aims to improve performance by allowing each client to individually leverage its own unique (and not mutually similar) features.

4.2.2 Continual Learning (CL)

CL addresses the challenge of *catastrophic forgetting* [92] when a model is trained on a sequence of tasks. Methods are typically grouped into replay-based approaches, regularization-based approaches, and parameter-isolation approaches [90]. Progressive Neural Networks (PNNs) [91], a parameter-isolation method, instantiate a new neural network column for each task and use lateral connections to transfer knowledge without overwriting parameters.

CL has been combined with FL in several contexts [93–97], often to address task sequences at clients or to mitigate client-drift issues due to distribution heterogeneity. In contrast, our use of CL has a distinct purpose: integrating common-feature and unique-feature representations while avoiding interference between global and local training.

4.3 Preliminaries

Throughout this chapter, we denote model parameters of a neural network as $\theta := (W_1, \dots, W_L, b_1, \dots, b_L)$, where W_l and b_l denote the weight matrix and the bias vector of the l -th layer.

4.3.1 Progressive Neural Network (PNN)

We briefly review Progressive Neural Networks (PNN) [91], a CL method based on parameter isolation. PNN learns a sequence of tasks $s \in \{1, \dots, S\}$ by allocating a new neural network (*column*) to each task while keeping previously learned parameters fixed. All tasks are assumed to share the same input feature space.

For task s , let θ^s denote the parameters of its column and $z_l^s \in \mathbb{R}^{n_l}$ the output of layer l . When learning a new task, lateral connections transfer representations from earlier columns. For $S = 2$, the intermediate output of the second column is computed as

$$z_{l+1}^2 = a(W_{l+1}^2 z_l^2 + b_{l+1}^2 + U_{l+1} z_l^1), \quad (4.1)$$

where U_{l+1} denotes the *lateral connection matrix* and a is the activation function.

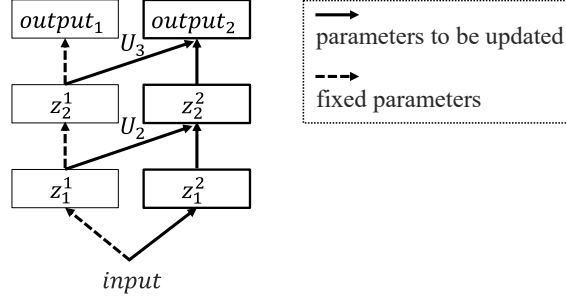


Figure 4.1: Description of a progressive neural network with three layers in the case of two tasks.

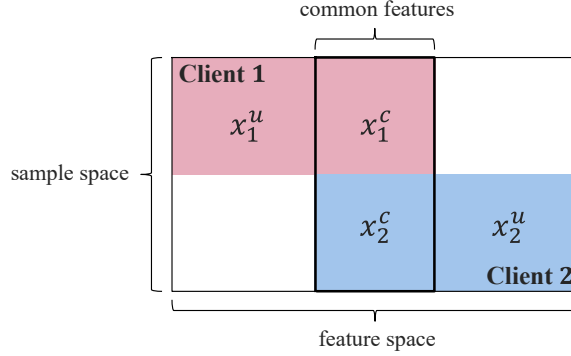


Figure 4.2: View of dataset structure in the two client scenario.

This design enables knowledge transfer from earlier tasks without catastrophic forgetting. Figure 4.1 illustrates the PNN architecture for two tasks.

4.3.2 Problem Formulation

We consider K clients in the personalized FL framework of Chapter 2. Each client k holds a private labeled dataset $D_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$. Clients have heterogeneous feature spaces with a non-empty set of common features. Following Chapter 2, we denote each input as $x_{k,i} = (x_{k,i}^c, x_{k,i}^u) \in \mathcal{X}_k = \mathcal{X}^c \times \mathcal{X}_k^u$, where $x_{k,i}^c$ represents the common features and $x_{k,i}^u$ represents the client-specific (unique) features. No assumptions are made about overlap of sample IDs or any cross-client sample alignment. Figure 4.2 illustrates this dataset structure for two clients.

Standard HFL operates only on $\mathcal{X}^c$, ignoring all $\mathcal{X}_k^u$. The goal of this chapter is to design a FL method that leverages both $\mathcal{X}^c$ and $\mathcal{X}_k^u$ while producing a personalized model for each client, capable of performing local inference using its full feature set and requiring no shared sample IDs.

4.4 Proposed Approach

We introduce the proposed approach FLEC to address the feature-space heterogeneity problem defined in Section 4.3.2.

4.4.1 Idea

The data owned by each client differ in their feature spaces; therefore, a vanilla HFL method cannot utilize all available features. To address this limitation, we split each client’s neural network into two columns: a *unique column* and a *common column*, which take unique and common features as inputs, respectively. We aim to jointly train the parameters of the common column via HFL while locally training the parameters of the unique column without interfering with the learning of the common column. This setting can be viewed as learning a unique task (inference using unique features) without forgetting the knowledge required for a common task (inference using common features).

Motivated by this perspective, we adopt a CL approach, specifically the PNN architecture [91]. The PNN architecture enables collaborative construction of the common column without being affected by client-specific unique features, while allowing the unique column to leverage federated knowledge from the common column through lateral connections. As a result, FLEC can exploit both common and unique features. In the following, we present the detailed model architecture and algorithm of FLEC.

4.4.2 Model Architecture

Fig. 4.3 shows the architecture of FLEC. FLEC has three components for each client k : common column with parameters θ^c , unique column with parameters θ_k^u , and lateral connection matrices $U_k := \{U_{k,l}\}_{l=2}^L$, where L is the number of layers in both columns. The unique column and lateral connection matrices are updated locally. We denote the full set of parameters for client k as $\Theta_k := (\theta^c, \theta_k^u, U_k)$. The outputs of the l -th layer of the common and unique columns for input (x^c, x^u) are denoted by z_l^c and z_l^u , respectively.

Let z_l^c and z_l^u denote the outputs of the l -th layer of the common and unique columns for input (x^c, x^u) , respectively. Following the PNN formulation (see Section 4.3.1), the lateral connection matrices connect intermediate outputs from the common column to those from the unique column. Accordingly, the output z_{l+1}^u of

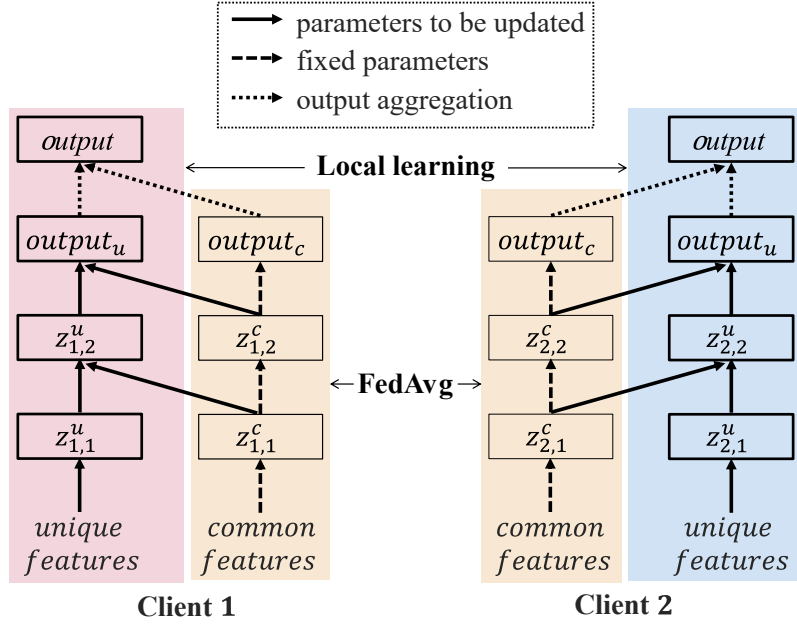


Figure 4.3: Architecture of FLEC for the case of $K = 2$ and $L = 3$. $z_{k,l}^c$ and $z_{k,l}^u$ denote the outputs of l -th layer of common and unique columns for client k 's input, respectively.

client k is computed in the same way as in Eq. (4.1):

$$z_{l+1}^u = a \left(W_{k,l+1}^u z_l^u + b_{k,l+1}^u + U_{k,l+1} z_l^c \right). \quad (4.2)$$

Unlike PNN, the two columns in FLEC take different feature inputs. Since both columns aim to solve the same prediction task, their final outputs are aggregated to improve performance. Therefore, for an input (x^c, x^u) , the final output of the full network Θ_k is given by

$$h_{\Theta_k}(x^c, x^u) = \sigma(z_L^c + z_L^u), \quad (4.3)$$

where σ denotes the softmax function.

4.4.3 Algorithm

We now describe the FLEC training algorithm. FLEC is built on FedAvg and involves both collaborative and local updates. Each client k collaboratively trains the common column θ^c using $(x_{k,i}^c, y_{k,i})$ in accordance with Algorithm 1. In parallel, each client locally trains their unique column θ_k^u and lateral connection matrices U_k using all $(x_{k,i}^c, x_{k,i}^u, y_{k,i})$. Algorithm 3 summarizes this procedure.

Specifically, at communication round r , the server first sends the current param-

Algorithm 3: FLEC: Continual horizontal federated learning

Input: K clients with local datasets $D_k = \{(x_{k,i}^c, x_{k,i}^u, y_{k,i})\}_{i=1}^{N_k}$, parameters of common column $\theta^{c,0}$, parameters of unique columns $\{\theta_k^u\}_{k=1}^K$, lateral connection matrices $\{U_k\}_{k=1}^K$, number of communication rounds R , number of local epochs E , learning rate η

Output: Trained parameters $\theta^{c,R}$ and $\{\theta_k^u\}_{k=1}^K$, trained matrices $\{U_k^R\}_{k=1}^K$

```

1 Server executes:
2 for  $r = 0$  to  $R - 1$  do
3   for  $k = 1$  to  $K$  in parallel do
4     Send  $\theta^{c,r}$  to client  $k$ 
5      $\theta_k^{c,r} \leftarrow \text{ClientLocalLearning}(k, \theta^{c,r})$ 
6    $\theta^{c,r+1} \leftarrow \frac{1}{K} \sum_{k=1}^K \theta_k^{c,r}$ 
7 return  $\theta^{c,R}$ 

8 ClientLocalLearning( $k, \theta_c^r$ ):
9    $\theta_k^{c,r} \leftarrow \theta_c^r$ 
10  for  $e = 1$  to  $E$  do
11    for each batch  $(\mathbf{x}^c, \mathbf{x}^u, \mathbf{y})$  of  $D_k$  do
12       $\theta_k^{c,r} \leftarrow \theta_k^{c,r} - \eta \nabla \widehat{\mathcal{L}}(\mathbf{x}^c, \mathbf{y}; \theta_k^{c,r})$ 
13       $\theta_k^u \leftarrow \theta_k^u - \eta \nabla_{\theta_k^u} \widehat{\mathcal{L}}(\mathbf{x}^c, \mathbf{x}^u, \mathbf{y}; \Theta_k^r)$ 
14       $U_k \leftarrow U_k - \eta \nabla_{U_k} \widehat{\mathcal{L}}(\mathbf{x}^c, \mathbf{x}^u, \mathbf{y}; \Theta_k^r)$ 
15  return  $\theta_k^{c,r}$ 

```

eters of common-column parameters $\theta^{c,r}$ to all clients. Each client k then performs local training using mini-batches $(\mathbf{x}^c, \mathbf{x}^u, \mathbf{y})$ of D_k . For each mini-batch, the client first updates the common column to obtain $\theta_k^{c,r}$ by minimizing the empirical loss $\widehat{\mathcal{L}}(\mathbf{x}^c, \mathbf{y}; \theta_k^{c,r})$ defined by Eq. (2.2). Subsequently, the client freezes $\theta_k^{c,r}$ and updates θ_k^u and U_k by minimizing the loss $\widehat{\mathcal{L}}(\mathbf{x}^c, \mathbf{x}^u, \mathbf{y}; \Theta_k^r)$, where $\Theta_k^r := (\theta_k^{c,r}, \theta_k^u, U_k)$ and the output $h_{\Theta_k^r}(x^c, x^u)$ is computed according to Eq. (4.3). The server then receives $\theta_k^{c,r}$ from all clients and aggregates them to obtain $\theta^{c,r+1}$. After R iterations, each client uses $h_{\Theta_k^R}$ for their inference model, leveraging both common and unique features.

To control the influence of lateral connections, we introduce a parameter $0 \leq \mu \leq 1$ that adjusts the strength of the interaction between the common and unique columns. When the common features and unique features are weakly correlated, strong lateral connections may hinder training of the unique column. To mitigate this effect, we replace the lateral matrices U_k with μU_k , where μ is a fixed hyperparameter rather than a learnable one. When $\mu = 0$, the common and unique columns interact only through the final-output aggregation in Eq. (4.3).

Table 4.1: Types of datasets

Dataset	Sample Num.	Feature Num.	Class Num.
coverttype	15,121	54	7
sensorless	58,509	48	11
aloi	108,000	128	1000
helena	65,196	27	100
jannis	83,733	54	4

4.5 Experiments

In this section, we evaluate the empirical performance of our FLEC in five datasets and show that it outperforms two baseline methods: a vanilla HFL using only common features and a local learning using both the common features and unique features. We first introduce experimental settings including datasets, compared methods, and implementation information.

4.5.1 Experimental Settings

4.5.1.1 Datasets

We conducted experiments on five public datasets: (i) forest coverttype dataset [98], (ii) sensorless drive diagnosis dataset [99], (iii) aloi [100], (iv) helena [101], and (v) jannis [101]. We use the training dataset of coverttype from Kaggle as a whole dataset so that the number of data samples for each class is the same. Detailed information of these datasets is shown in Table 6.2.

4.5.1.2 Compared Methods

We compare FLEC with following four baseline methods:

- Common: A vanilla HFL method over common features among different clients is used. We use FedAvg as a vanilla HFL.
- Local: Each client locally trains a neural network using all of their own features, i.e., both common features and unique features.
- ZeroImp: Each client replaces the private features owned by other clients (i.e., features it does not possess) with zeros and then conducts FedAvg using the complete feature set, including the imputed values.

- MeanImp: The procedure is the same as in ZeroImp, except that the missing features are imputed with the corresponding mean values. Before training, each client computes the mean of its own private features and shares these mean statistics with all other clients.

In all methods including FLEC, we use simple fully-connected neural networks composed of three hidden layers with 512, 256, and 128 neurons, respectively. In FLEC, we use the same architecture with the above mentioned size for the common column and unique column.

4.5.1.3 Implementations

All datasets are split randomly with 60%, 20%, and 20% for training, validation, and testing, respectively. We split and distribute training and validation datasets evenly to each client. Each client tunes hyperparameters such as learning rate and μ of FLEC using their own validation dataset. We evaluate the performance of each method by average testing accuracy of all clients. In all experiments, we randomly split features into common features and others. The ratio of the common features is set to 30% unless explicitly specified. The remaining features are divided into the number of clients, and each part is used as unique features of the corresponding client. We conduct experiments three times changing the way that features are split randomly and compute their average accuracy. Moreover, for each splitting, five experiments are performed and averaged. Throughout the experiments, the number of clients, the number of local epochs, and batch size are set to 5, 5, and 64 by default, respectively. We use Adam as the optimizer. All experiments are conducted using the PyTorch 1.8 framework on a Tesla V100 GPU with 32-GB memory.

4.5.2 Experimental Results

4.5.2.1 Main results

Table 4.2 shows the average test accuracy of five clients with standard deviations on five datasets. The highest average test accuracy is marked in bold. In FLEC, we compare two cases, $\mu = 0$ and $\mu > 0$ to investigate the effectiveness of lateral connections between two columns. In the case of $\mu > 0$, we use μ to demonstrate the best performance on validation dataset. From Table 4.2, we can see that FLEC outperforms the four baselines for every dataset. From the fact that its performance substantially surpasses both Common and Local, we can see that the effects

Table 4.2: Mean and standard deviation (%) of accuracy for the baselines and FLEC across the five datasets.

Dataset	Baseline				FLEC	
	Common	Local	ZeroImp	MeanImp	$\mu = 0$	$\mu > 0$
coverttype	62.18 $\pm$ 10.86	64.07 $\pm$ 5.94	65.27 $\pm$ 7.56	65.76 $\pm$ 7.20	67.44 $\pm$ 7.31	68.43 $\pm$ 6.61
sensorless	97.62 $\pm$ 0.24	95.26 $\pm$ 0.46	96.54 $\pm$ 0.83	96.34 $\pm$ 1.01	98.55 $\pm$ 0.17	98.63 $\pm$ 0.20
aloi	73.52 $\pm$ 3.66	70.93 $\pm$ 1.65	75.79 $\pm$ 2.12	75.81 $\pm$ 2.30	79.27 $\pm$ 1.96	79.54 $\pm$ 1.97
helena	26.04 $\pm$ 1.47	27.32 $\pm$ 1.00	28.44 $\pm$ 0.96	28.61 $\pm$ 0.94	28.68 $\pm$ 0.86	29.02 $\pm$ 0.88
jannis	64.49 $\pm$ 1.84	63.41 $\pm$ 0.90	64.79 $\pm$ 1.21	64.77 $\pm$ 1.18	65.51 $\pm$ 1.28	65.72 $\pm$ 1.11

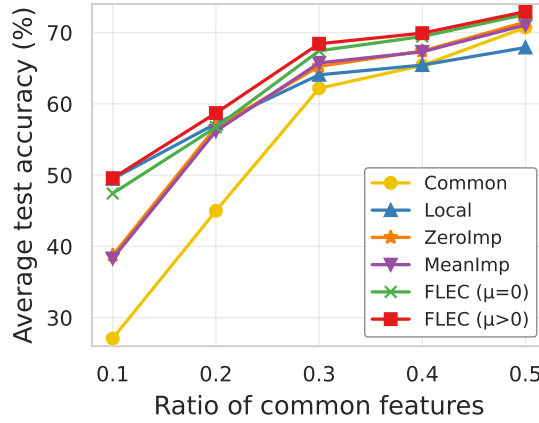


Figure 4.4: Variation of average test accuracy for coverttype with respect to the ratio of the common features.

of each client’s private features and the benefits of federated learning on the shared features are both being realized. Furthermore, because FLEC improves the performance of ZeroImp and MeanImp, it is evident that in ZeroImp and MeanImp the clients’ private features actually degrade the overall performance of federated learning. Moreover, it is shown that non-zero μ improves FLEC, which means lateral connections effectively transfer the knowledge of the common column to the unique column. However, we can also see that even the case without lateral connections (i.e., the case of $\mu = 0$) sufficiently improves baseline methods.

4.5.2.2 Ratio of common features

The effect of the ratio of common features on test accuracy for coverttype is shown in Fig. 4.4. In Fig. 4.4, the ratio is changed from 10% to 50%. We can see that FLEC in the case of $\mu > 0$ is always the best, but approaches baselines when the ratio of common features is small or large. This is because when the ratio of common

Table 4.3: Mean and standard deviation (%) of accuracy for the baselines and FLEC with different numbers of clients on the covertedype dataset.

Number of clients	Baseline				FLEC	
	Common	Local	ZeroImp	MeanImp	$\mu = 0$	$\mu > 0$
2	64.20 $\pm$ 10.98	76.49 $\pm$ 1.44	75.70 $\pm$ 1.66	75.66 $\pm$ 1.60	74.75 $\pm$ 3.56	77.48 $\pm$ 2.24
5	62.18 $\pm$ 10.86	64.07 $\pm$ 5.94	65.27 $\pm$ 7.56	65.76 $\pm$ 7.20	67.44 $\pm$ 7.31	68.43 $\pm$ 6.61
10	60.14 $\pm$ 11.12	56.28 $\pm$ 8.54	60.91 $\pm$ 9.94	60.79 $\pm$ 10.05	62.90 $\pm$ 9.21	62.91 $\pm$ 9.14

Table 4.4: Mean and standard deviation (%) of accuracy for the baselines and FLEC under different correlation strengths on the helena dataset.

Correlation strength	Baseline				FLEC	
	Common	Local	ZeroImp	MeanImp	$\mu = 0$	$\mu > 0$
Maximum	27.17 $\pm$ 0.62	28.11 $\pm$ 0.34	28.49 $\pm$ 0.18	28.47 $\pm$ 0.23	28.67 $\pm$ 0.47	29.16 $\pm$ 0.39
Median	27.97 $\pm$ 1.68	29.26 $\pm$ 0.80	31.09 $\pm$ 0.30	31.07 $\pm$ 0.19	29.99 $\pm$ 1.04	30.49 $\pm$ 0.91
Minimum	17.93 $\pm$ 1.33	24.76 $\pm$ 0.19	26.25 $\pm$ 0.32	26.36 $\pm$ 0.34	26.95 $\pm$ 0.16	27.10 $\pm$ 0.13

features is small, the effect of HFL is small, and when the ratio of common features is large, the effect of using unique features is small. In fact, the test accuracy of Common is greatly improved as the ratio of common features increases. We can also see that as the ratio of the common features increases, the difference in performance between FLEC ($\mu = 0$) and FLEC ($\mu > 0$) becomes smaller, i.e., the effect of lateral connections becomes smaller.

4.5.2.3 Number of clients

We investigated the effect of the number of clients. The results in the case of 2, 5, 10 clients for covertedype are shown in Table 4.3. The ratio of common features is fixed to 30%. The results show that the difference of FLEC and baselines is small in the case of two clients and ten clients. The reason for this is similar to the experiment in Fig. 4.4: when there are many clients (few unique features), the effect of using unique features is small, and when there are few clients, the effect of HFL is small.

4.5.2.4 Strength of correlation

As mentioned in Section 4.4.3, the strength of the correlation between the common features and unique features is considered to be related to the performance of FLEC. To confirm this intuition, we calculated the correlation coefficients between the common features and unique features using helena. First, we split features into

common features and others. Then, we calculate the correlation coefficients between any column of the common features and any column of the remaining features and sum up them. This summation of the correlation coefficients was computed in every way of feature splitting. We conducted experiments in the three ways of feature splitting where the summation of the correlation coefficients is maximum, minimum, and median, respectively. The results are shown in Table 4.4. Comparing FLEC ($\mu = 0$) and FLEC ($\mu > 0$), we can see that there is little difference between them in the case of Minimum. Therefore, when the correlation between common features and unique features is weak, a simple method FLEC without lateral connections is enough.

4.5.2.5 Comparison to another simple method

In addition to the simple method, FLEC without lateral connections, we compared FLEC with lateral connections to another simple method. We consider the following method. As with FLEC, clients collaboratively train a common neural network using common features by applying FedAvg. In parallel to that, each client trains a unique neural network that takes the concatenation of the unique features and the output of the common network as input. Thus, this method can also utilize all features. We evaluated the performance of this simple method for coverytype and jannis in the same setting as Table 4.2, which resulted in the test accuracy of 66.37 ± 7.09 and 65.28 ± 1.11 , respectively. Comparing with Table 4.2, we find that this method is less effective than even FLEC without lateral connections. Hence, we can see that our lateral connections and output aggregation are effective.

4.6 Summary

We proposed FLEC, a continual learning approach to address the heterogeneous data problem in HFL. FLEC prepares two neural networks corresponding to common features and unique features, respectively and laterally connect them while training the common network with HFL and locally training the unique network. Through the experiments on various datasets, FLEC was shown to improve vanilla HFL method that only uses common features and outperform local learning that uses all features of the corresponding client.

Chapter 5

Federated Source-free Domain Adaptation for Classification: Weighted Cluster Aggregation for Unlabeled Data

5.1 Introduction

Most federated learning (FL) methods assume that labeled data are available at each client. As discussed in Chapter 1, this may not be the case for vision tasks which often necessitate cumbersome manual data annotation requiring domain expertise. For instance, in applying FL to multi-location surveillance cameras, it is burdensome for administrators to manually annotate each image. While a server, typically hosted by organizations, is more likely to access labeled data, the included biometric information such as faces, makes direct use of this data in FL problematic due to privacy concerns. This scenario combines two types of heterogeneity introduced in Chapter 2: *label-availability heterogeneity*, since clients hold only unlabeled data, and *distribution heterogeneity* in the form of *covariate shift*, because camera environments (e.g., lighting, background) differ across locations.

Building on the preceding example, this chapter explores the practical yet less-investigated FL setting of *Federated source-Free Domain Adaptation* (FFREEDA) [1], specifically for classification tasks (Fig. 5.1). In FFREEDA, a server possesses a model pre-trained on labeled data from a source domain, but this data is not usable in later FL stages due to privacy issues or immediate deletion after pre-training.

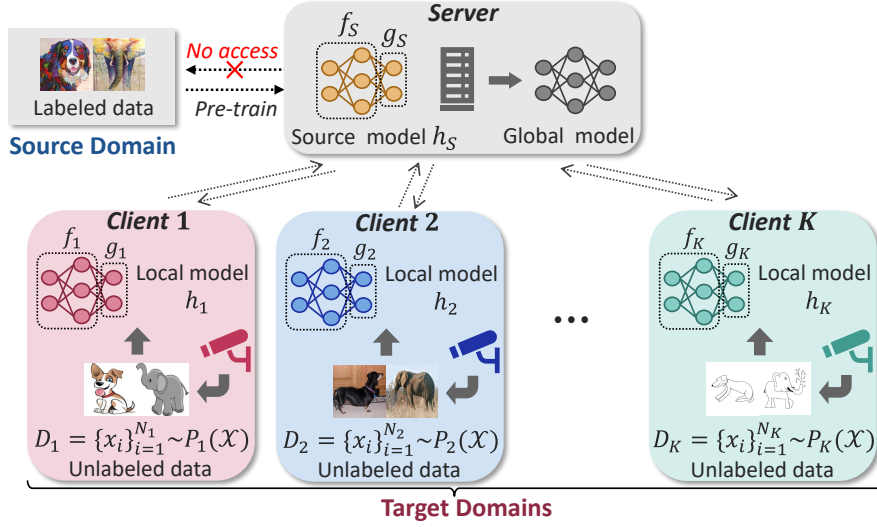


Figure 5.1: Overview of our problem setting: FFREEDA [1]. A pre-trained model on a labeled *source* domain dataset exists on the server, which becomes inaccessible following pre-training (*source-free*). Clients possess unlabeled datasets from specific *target* domains, which may or may not be identical. We aim to train personalized models tailored to each client’s domain.

In addition, each client holds limited unlabeled data in a target domain, which may greatly differ from the source domain and the domains of other clients, thus creating an unsupervised multi-target setting. FFREEDA aims for each client to train a classification model tailored to its domain, leveraging the source model and the knowledge from other clients.

We tackle two main challenges of FFREEDA: (1) fully unsupervised FL after pre-training and (2) domain shifts (covariate shifts) among clients. We propose a novel FFREEDA method for classification, *Federated Learning with Weighted Cluster Aggregation* (FedWCA). This overcomes issues in the prior FFREEDA method [1] for semantic segmentation, including privacy issues connected to image style feature sharing during client clustering and limited performance (Section 5.5) due to a cluster-restricted model aggregation and a simple pseudo-labeling. Specifically, FedWCA comprises three key components. Firstly, to avert performance degradation from domain shift (2), we cluster clients based on the initial layer parameters of each client’s locally adapted feature extractor, which contains domain information. This process relies only on model parameters, eliminating the need for additional information sharing and hyperparameters. To further boost performance, we introduce the Weighted Cluster Aggregation (WCA) module, enabling clients to combine models from each cluster (domain) using personal weights. A novel metric is de-

Table 5.1: Related problem settings. Our study focuses on classification tasks, tackling three distinct problem types: "No Labels," "Domain Shift," and "Personalization." $\triangle$ denotes specific characteristics depending on FL paradigms: while most Personalized FL research concentrates on label shifts, certain studies like [3, 4] also tackle domain shifts. Semi-Supervised FL literature typically involves unlabeled data, presuming the presence of labeled data either on the server or within a subset of clients, a scenario also observed in some FDA studies.

FL paradigm	No labels	Domain shift	Personalization	Task
Personalized FL [3, 4, 6, 41]		$\triangle$	$\checkmark$	Classification
Semi-supervised FL [52–57]	$\triangle$			Classification
Unsupervised FL [59–63]	$\checkmark$			RepL/Clustering
FDA [102–106]	$\triangle$	$\checkmark$		Classification
[107, 108]	$\triangle$	$\checkmark$	$\checkmark$	Classification
FFREEDA LADD [1]	$\checkmark$	$\checkmark$	$\checkmark$	Semantic segmentation
Ours	$\checkmark$	$\checkmark$	$\checkmark$	Classification

financed for computing cluster weights based solely on unlabeled data, considering the benefits for both individual clients and the overall group. Finally, handling unlabeled target data (1) is achieved through prototype-based pseudo-labeling [109]. We improve it to retain global knowledge and produce more reliable pseudo-labels. Empirical validation confirms the efficacy of these components, with the complete method outperforming all baselines, as detailed in Section 5.5.

This work primarily contributes as follows:

- We introduce a novel FFREEDA method for classification, named FedWCA, with three main components: (1) a private, parameter-free client clustering module, (2) a weighted cluster aggregation module for domain knowledge integration, and (3) a local adaptation module using an improved pseudo-labeling strategy.
- Empirical results show that FedWCA significantly enhances performance over all baselines, including the prior method [1], across three multi-domain benchmark datasets: Digit-Five, PACS, and Office-Home.

5.2 Related Work

5.2.1 Source-free Domain Adaptation

This chapter extends *Source-free domain adaptation* (SFDA), a type of unsupervised domain adaptation approach, to the FL setting. SFDA involves adapting a pre-trained source model to an unlabeled target domain without accessing the

source data. A fundamental SFDA method is pseudo-labeling for unlabeled target data [109–114]. A prototype-based pseudo-labelling method introduced by *Source HypOthesis Transfer* (SHOT) [109], is commonly employed to derive pseudo-labels, minimizing label noise induced by domain shifts [110–112].

5.2.2 Federated Domain Adaptation

FADA [102] is the pioneering work addressing *Federated Domain Adaptation* (FDA) for a single target domain. Subsequent studies [103–106] have also focused on a multi-source-single-target setting. DualAdapt [107] explores a single-source-multi-target scenario, similar to this chapter, where each client gets one target domain and the server holds a source domain. FRuDA [108] considers sequentially adding new target domains to the federated system. Unlike our source-free setting, these studies typically assume access to source data during FL (Table 5.1).

5.2.3 Federated Source-free Domain Adaptation

Our model is inspired by *Learning Across Domains and Devices* (LADD) [1], the sole study addressing FFREEDA, which focuses on semantic segmentation. In contrast, our model is designed for classification tasks, decreases the number of hyperparameters, alleviates privacy concerns by eliminating the need to transmit image style features from clients to servers for client clustering (Section 5.4.1), integrates all domain information with client-specific weights (Sec. 5.4.2), generates more reliable pseudo-labels (Section 5.4.3), and thus improves accuracy (Sec. 5.5).

5.3 Problem Formulation

We now formalize the FFREEDA setting in the notation of Chapter 2. A server and K clients cooperate on an M -class classification problem. Each client k holds an unlabeled dataset $D_k = \{x_i\}_{i=1}^{N_k}$ drawn from its own target-domain distribution $P_k(\mathcal{X})$ (client side in Fig. 5.1), and these distributions may differ across clients due to covariate shift. Some clients may share the same domain, but no domain labels or domain groupings are assumed.

The server has a source model $h_S: \mathcal{X} \rightarrow \mathbb{R}^M$ pre-trained on a labeled dataset from the source domain whose data distribution differs from any client’s distribution P_k (server side in Fig. 5.1). The labeled source dataset is discarded or retained

privately, preventing further access. The model h_S has two modules: $h_S = g_S \circ f_S$, where $f_S: \mathcal{X} \rightarrow \mathbb{R}^q$ is a feature extractor, $g_S: \mathbb{R}^q \rightarrow \mathbb{R}^M$ is a classifier, and $\circ$ denotes the composition operation. The feature extractor f_S converts a sample x into a q -dimensional feature vector $f_S(x)$, while the classifier g_S assigns class scores to $f_S(x)$, with $(g_S(f_S(x)))_m$ denoting the score for class m . The final linear layer of a neural network, typically viewed as the classifier g_S , has the weight matrix $W = [w_1, \dots, w_M] \in \mathbb{R}^{q \times M}$, where w_m is the classifier vector for class m .

The aim of FFREEDA is to evolve from the source model h_S to learn multiple personalized classification models $h_k = g_k \circ f_k$, each tailored to the respective client distributions $P_k(\mathcal{X})$, while encouraging collaboration among clients.

See Table 5.7 in Appendix 5.7.1 for a full list of notation.

5.4 Method

FedWCA comprises three main components: (1) private and parameter-free client clustering, (2) weighted cluster aggregation, and (3) local adaptation via pseudo-labeling. In line with prevalent SFDA practices [109, 111, 112], we fix the classifier during FL, anticipating that the source model’s classifier has developed a domain-independent capability for the task. See Appendix 5.7.2.1 for the pseudo-code of FedWCA.

5.4.1 Private and Parameter-free Client Clustering

To manage domain shifts among clients, our method clusters clients privately and without parameters as a first step, based on locally learned feature extractor parameters f_k by each client. This approach removes the need for server access to raw client data and requires only a one-time initial clustering operation.

Unlike traditional clustered FL methods that require hyperparameters such as the number of clusters, our strategy employs a parameter-free clustering: *First Integer Neighbor indices producing a Clustering Hierarchy* (FINCH) [115]. We use cosine similarity between clients’ model parameters to identify the nearest neighbor κ_k for each client k . The adjacency matrix is defined as $A(k, l) = 1$ if $l = \kappa_k$, $k = \kappa_l$, $\kappa_k = \kappa_l$, and $A(k, l) = 0$ otherwise. Clients linked by this matrix are grouped into the same cluster.

Given the typically large number of parameters in a feature extractor, using all parameters for distance evaluation can lead to a single cluster due to the curse of

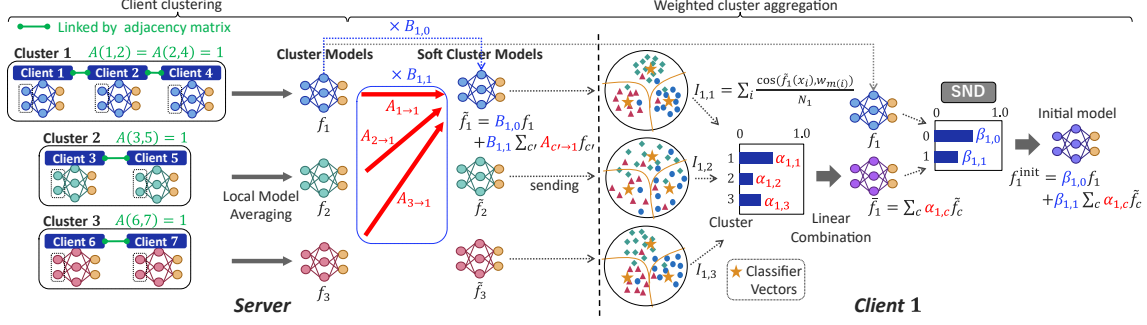


Figure 5.2: Weighted cluster aggregation (WCA). Our WCA consists of server-side (left) and client-side (right) operations. The client-side operations are represented by client 1 in cluster 1. Firstly, the server forms cluster models by averaging the local models within each cluster and develops soft cluster models by integrating models from other clusters. Next, the client merges these soft cluster models utilizing cluster weights determined by how each feature extractor aligns each sample with the respective classifier vector. Additionally, the client combines the merged model with the corresponding cluster model using Soft Neighborhood Density (SND) weights [2].

dimensionality (see Section 5.5.2.2). To mitigate this, we focus on the shallow layers where domain knowledge is stored [116]. Specifically, we only use the first-layer parameters as this layer, the furthest from the common classifier among clients, best represents domain-specific information (Section 5.5.2.2). Consequently, the server can derive C clusters (indexed by c) and generate cluster models f_c by averaging the local models in each cluster. In the next section, we construct each client’s initial model for the next round using the cluster models.

5.4.2 Weighted Cluster Aggregation

The core of our FedWCA method is the weighted cluster aggregation, depicted in Fig. 5.2. In conventional clustered FL methods, the server distributes the cluster model f_{c_k} to client k , where c_k is the cluster index assigned to client k , for use as the initial model for the next round. This approach limits clients to knowledge solely from their assigned cluster, challenging clients in clusters with few data points or whose domains are significantly different from the source domain. To overcome this, our strategy enables clients to blend all cluster models f_c using specific cluster weights $\mathbf{v}_k = (v_{k,1}, \dots, v_{k,C})$ (summing to 1), to form an initial model $f_k^{\text{init}} = \sum_c v_{k,c} f_c$ in each round r^1 aligns with the source feature extractor f_S , and g_k^{init} remains fixed to the source classifier g_S across all rounds. This blended model not only leverages cross-cluster knowledge but also corrects possible cluster assign-

¹In the initial round, f_k^{init}

ment errors. At the end of the training, each client uses its cluster’s model for the performance evaluation. The specific cluster weights calculation method is described below.

Local calculation of cluster weights. Firstly, we explore a method for clients to compute cluster weights locally, vital for model personalization. The underlying principle is to assign higher weights to those cluster models that are most beneficial for a client’s domain.

Considering that an effective feature extractor converges samples towards a corresponding class-specific classifier vector defined in Section 5.3, which effectively acts as a prototype in the feature space, we introduce new metrics $\alpha_k = (\alpha_{k,1}, \dots, \alpha_{k,C})$ to represent the cluster weights $\mathbf{v}_k$ for each client k . First, client k evaluates the cosine similarity $\cos(f_c(x_i), w_{m(i)})$ between the feature extractor output $f_c(x_i)$ and the corresponding class’ classifier vector $w_{m(i)}$, where $m(i) = \arg \max_m \cos(f_c(x_i), w_m)$ represents the class whose classifier vector is closest to $f_c(x_i)$. The similarity scores across all samples are averaged to compute $I_{k,c} = \sum_{i=1}^{N_k} \cos(f_c(x_i), w_{m(i)}) / N_k$ for each cluster c and α_k are derived using a temperature parameter T_a as follows:

$$(\alpha_{k,1}, \dots, \alpha_{k,C}) = \text{softmax}(I_{k,1}/T_a, \dots, I_{k,C}/T_a), \quad (5.1)$$

where ‘softmax’ is the softmax function.

However, this metric focuses solely on the benefits to the individual client, without considering the overall advantages across the clients. Such a client-centric approach may not necessarily contribute positively to the other clients.

Global and local calculation of cluster weights. We calculate the cluster weights, focusing on both individual client benefits and collective advantages. This involves using both cluster models f_c and ‘soft cluster models’ $\tilde{f}_c$, which integrate benefits from other clusters.

For creating soft cluster models, we define a benefit metric $A_{c' \rightarrow c}$, with $\sum_{c'} A_{c' \rightarrow c} = 1$, indicating the relative advantage of cluster c for clients in cluster c' . This metric formulates new cluster models by weighting $f_{c'}$ as $\sum_{c'} A_{c' \rightarrow c} f_{c'}$. To prevent loss of knowledge from the original cluster models, we introduce coefficients $B_{c,0}$ and $B_{c,1}$, summing to 1, for each cluster. These coefficients balance the emphasis between the original cluster model f_c and the newly combined model $\sum_{c'} A_{c' \rightarrow c} f_{c'}$. The soft

cluster models are thus given by

$$\tilde{f}_c = B_{c,0}f_c + B_{c,1} \sum_{c'} A_{c' \rightarrow c} f_{c'}. \quad (5.2)$$

$A_{c' \rightarrow c}$ and $B_{i,c}$ are estimated later in this subsection.

Upon receiving the original cluster model f_{c_k} and all soft cluster models $\tilde{f}_c$ from the server, each client k locally calculates cluster weights α_k for the soft cluster models, leading to a composite model $\bar{f}_k = \sum_c \alpha_{k,c} \tilde{f}_c$. To adequately incorporate the original cluster model f_{c_k} , client k employs pseudo-performance weights $\beta_k = (\beta_{k,0}, \beta_{k,1})$ (summing to 1) to balance between f_{c_k} and $\bar{f}_k$. These weights are derived using Soft Neighborhood Density (SND) [2], a method that assesses the overall model efficacy on unlabeled data by analyzing the entropy of similarity distributions between outputs:

$$(\beta_{k,0}, \beta_{k,1}) = \text{softmax}(S(h_{c_k})/T_b, S(\bar{h}_k)/T_b), \quad (5.3)$$

where T_b is the temperature parameter, S is the function to measure SND (detailed in Appendix 5.7.2.2), $h_{c_k} = g_S \circ f_{c_k}$, and $\bar{h}_k = g_S \circ \bar{f}_k$. The initial model for client k is therefore computed as $f_k^{\text{init}} = \beta_{k,0}f_{c_k} + \beta_{k,1} \sum_c \alpha_{k,c} \tilde{f}_c$. Expanding this equation with respect to the cluster model f_c yields the final cluster weights $\mathbf{v}_k$, shown in Appendix 5.7.2.3

On the server side, the global coefficients $A_{c \rightarrow c'}$ and $B_{c,i}$ can be estimated by averaging the local weights $\alpha_{k,c'}$ and $\beta_{k,i}$ from the previous round within each cluster c because each cluster comprises clients from similar domains. This ensures the cluster weights in f_k^{init} reflect both local preferences and global trends, aligning individual client needs.

Further discussion on privacy concerns and additional costs in distributing all soft cluster models to each client is found in Appendix 5.7.4. It also provides a proposal for reducing communication, computation, and storage costs of our approach by shifting client-side computations to the server.

5.4.3 Local Adaptation with Pseudo-Labeling

To adapt the initial model f_k^{init} constructed in the previous section to each client k from only unlabeled data, our local adaptation module adopts the pseudo-labeling method for its simplicity and effectiveness (Fig. 5.3), while any existing SFDA

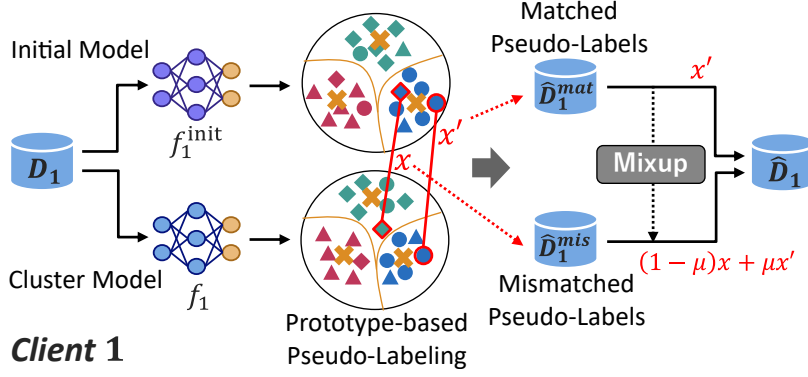


Figure 5.3: Pseudo-labeling strategy. The operation is signified by client 1 in cluster 1. Our strategy involves two phases: prototype-based pseudo-labeling with initial and cluster models, and mixup to generate samples with more reliable pseudo-labels. The first phase selects pseudo-labels closer to prototypes, while the second mixes samples with mismatched and matched pseudo-labels.

method could be applicable for local adaptation. This choice is particularly pertinent given the limited computational resources and memory capacities of individual clients, compared with other SFDA methods [117–122]. We further alter an existing feature prototype-based pseudo-labeling strategy [109–112] to lessen overfitting to sparse client data and forgetting global knowledge, and acquire more reliable pseudo-labels. Overfitting is mitigated by using round-fixed pseudo-labels while reliable pseudo-labels are obtained by using both initial and cluster models for pseudo-labeling, followed by mixup.

Pseudo-labeling strategy. First, we describe the existing prototype-based pseudo-labeling via a model $h = g \circ f$. The procedure involves calculating class-wise feature prototypes p_m for each class m and subsequently determining pseudo-labels $\hat{y}$ for data points x using the cosine similarity:

$$p_m = \frac{\sum_{x \in D_k} (h(x))_m f(x)}{\sum_{x \in D_k} (h(x))_m}, \quad \hat{y} = \arg \max_m d_m, \quad (5.4)$$

where $d_m := \cos(f(x), p_m)$ and $\cos(\cdot, \cdot)$ represents the cosine similarity. Clients then recompute the prototypes using these one-hot pseudo-labels instead of $h(x)$ in Eq. (5.4) and obtains the final pseudo-labels using these prototypes.

In our setting, each client obtains f_k^{init} and f_{c_k} in each round, allowing every data point x to have two pseudo-labels generated by these models. We select the pseudo-label with the larger similarity $d_{\hat{y}} = \cos(f(x), p_{\hat{y}})$. However, as this similarity is calculated on the feature space constructed by each feature extractor, a similarity nor-

Table 5.2: Results (Digit-Five, PACS, and Office-Home). Each row represents the selected source domain denoted by the initial two letters for each dataset, and the numbers are the mean values $\pm$ standard deviations of averaged accuracy (%) across all clients and all target domains. Thees figures are computed from 15 trials. The best numbers in each row are denoted in bold and the second best are underlined.

Datasets		Methods					
		Source Only	Local	FedAvg	LADD	FedPCL+PL	FedWCA (ours)
Digit-Five	MN	41.81 $\pm$ 0.46	50.59 $\pm$ 1.25	<u>55.88 $\pm$ 2.59</u>	49.00 $\pm$ 0.82	47.84 $\pm$ 0.79	72.74 $\pm$ 3.57
	SV	57.40 $\pm$ 1.39	83.00 $\pm$ 0.99	<u>83.76 $\pm$ 1.97</u>	78.40 $\pm$ 0.64	77.73 $\pm$ 0.56	90.13 $\pm$ 2.08
	MN-M	59.87 $\pm$ 1.71	69.44 $\pm$ 0.97	<u>80.14 $\pm$ 0.46</u>	72.93 $\pm$ 1.25	70.06 $\pm$ 1.94	82.93 $\pm$ 0.79
	US	46.93 $\pm$ 0.54	56.76 $\pm$ 1.13	53.23 $\pm$ 0.73	<u>56.93 $\pm$ 1.59</u>	56.41 $\pm$ 0.62	58.56 $\pm$ 4.26
	SY	68.69 $\pm$ 0.27	81.48 $\pm$ 0.32	80.81 $\pm$ 0.98	<u>83.79 $\pm$ 0.66</u>	81.51 $\pm$ 0.33	84.94 $\pm$ 1.15
	Avg.	54.94 $\pm$ 9.92	68.25 $\pm$ 13.07	<u>70.76 $\pm$ 13.49</u>	67.67 $\pm$ 13.25	66.71 $\pm$ 12.88	77.86 $\pm$ 11.56
PACS	Ar	65.87 $\pm$ 1.19	73.01 $\pm$ 1.40	<u>76.57 $\pm$ 1.38</u>	74.20 $\pm$ 1.48	73.83 $\pm$ 2.84	80.63 $\pm$ 2.44
	Ca	68.66 $\pm$ 1.02	77.27 $\pm$ 1.23	76.15 $\pm$ 0.72	77.37 $\pm$ 1.12	<u>79.18 $\pm$ 1.51</u>	83.18 $\pm$ 0.86
	Ph	54.20 $\pm$ 1.77	63.85 $\pm$ 1.38	60.94 $\pm$ 0.96	64.77 $\pm$ 1.80	<u>65.27 $\pm$ 1.07</u>	65.50 $\pm$ 1.24
	Sk	50.07 $\pm$ 3.21	75.56 $\pm$ 6.05	<u>80.42 $\pm$ 6.63</u>	61.82 $\pm$ 4.88	59.41 $\pm$ 6.22	84.22 $\pm$ 6.60
	Avg.	59.70 $\pm$ 8.18	72.42 $\pm$ 6.10	<u>73.52 $\pm$ 8.23</u>	69.54 $\pm$ 7.02	69.42 $\pm$ 8.42	78.38 $\pm$ 8.38
Office-Home	Ar	54.03 $\pm$ 0.30	62.50 $\pm$ 0.78	<u>63.81 $\pm$ 0.85</u>	57.33 $\pm$ 0.72	58.20 $\pm$ 0.73	66.06 $\pm$ 0.50
	Cl	58.36 $\pm$ 1.29	64.57 $\pm$ 0.70	68.73 $\pm$ 0.75	61.72 $\pm$ 0.61	62.39 $\pm$ 0.98	<u>68.32 $\pm$ 0.42</u>
	Pr	55.36 $\pm$ 0.72	59.17 $\pm$ 0.40	<u>60.15 $\pm$ 0.48</u>	57.22 $\pm$ 0.76	58.85 $\pm$ 0.46	61.46 $\pm$ 0.88
	Re	62.28 $\pm$ 0.35	65.38 $\pm$ 1.15	<u>65.93 $\pm$ 0.34</u>	64.23 $\pm$ 0.44	65.75 $\pm$ 0.80	68.06 $\pm$ 0.46
	Avg.	57.51 $\pm$ 3.33	62.90 $\pm$ 2.54	<u>64.65 $\pm$ 3.22</u>	60.25 $\pm$ 2.96	61.30 $\pm$ 3.14	65.97 $\pm$ 2.83

malized by the mean similarity between prototypes $\sum_{m \neq m'} \cos(p_m, p_{m'}) / M(M-1)$ is used to select the correct pseudo-label. This results in a pseudo-labelled dataset $\hat{D}_k = \{x_i, \hat{y}_i\}_{i=1}^{N_k}$ for each client. Contrary to common practices in SFDA, where pseudo-labels are updated in each epoch, our method keep these pseudo-labels constant throughout each round, ensuring the preservation of the global knowledge in f_k^{init} and f_{c_k} .

If f_k^{init} and f_{c_k} produce identical pseudo-labels, these are deemed more reliable; distinct pseudo-labels indicate reduced reliability. Therefore, we divide $\hat{D}_k$ into $\hat{D}_k^{\text{mat}}$ and $\hat{D}_k^{\text{mis}}$, having matched and mismatched pseudo-labels between f_k^{init} and f_{c_k} , respectively. For each instance $(x, \hat{y}) \in \hat{D}_k^{\text{mis}}$, we mix x with the randomly sampled data point $(x', \hat{y}) \in \hat{D}_k^{\text{mat}}$, resulting in $x_{\text{mix}} = (1 - \mu)x + \mu x'$ with the more reliable pseudo-label $\hat{y}$. Here, μ is a balancing parameter fixed to 0.55 in our experiments. We thus generate a pseudo-labeled dataset $\hat{D}_k = \hat{D}_k^{\text{mat}} \cup \hat{D}_k^{\text{mix}}$, where $\hat{D}_k^{\text{mix}}$ is a dataset derived from the mixup operation.

Loss function. We adopt the same loss function as SHOT [109] because of its simplicity. The total empirical loss function $\hat{\mathcal{L}}_k = \hat{\mathcal{L}}_{\text{IM}} + \lambda \hat{\mathcal{L}}_{\text{CE}}$ for client k with a balancing parameter λ comprises the cross-entropy loss for the pseudo-labeled dataset $\hat{D}_k$: $\hat{\mathcal{L}}_{\text{CE}}$ and an Information Maximization (IM) loss [109, 123] for the unlabeled dataset D_k : $\hat{\mathcal{L}}_{\text{IM}}$ (detailed in Appendix 5.7.2.4). The IM loss promotes confident model outputs and discourages trivial score distributions.

5.5 Experiment

Datasets. Our method was evaluated on three classification datasets, each with diverse domains. The **Digit-Five** dataset encompasses 5 benchmarks for handwritten-digit classification across 10 categories. It includes MNIST[124], MNIST-M[19], SVHN[125], SYNTH[19], and USPS. The **PACS** [126] dataset consists of 4 domains with 7 categories each, including Art Painting, Cartoon, Photo, and Sketch. The **Office-Home** [127] dataset is designed for object recognition, comprising 4 domains with 65 categories each and includes Art, Clipart, Product, and Real-World. For each dataset, one domain was selected as the source, and the remaining domains served as target domains. Data samples from each target domain were distributed to 8 clients for Digit-Five and 3 clients for PACS and Office-Home per domain, all in an i.i.d. manner. This resulted in a total of 32 clients for Digit-Five and 9 for PACS and Office-Home.

Models. Our model architecture reflects common structures used in SFDA research like SHOT [109], incorporating a feature extractor, a dimension-reducing bottleneck layer, and a classifier. The feature extractor differs across datasets. Following prior domain adaptation and FL studies, we use a randomly initialized LeNet [124] with two convolution layers for Digit-Five. For PACS and Office-Home, we use ResNet-18 and ResNet-50 [9] respectively, both pre-trained on ImageNet. The classifier is the last linear layer and the bottleneck layer consists of a fully connected layer equipped with batch normalization [128] and dropout [129] for all datasets.

Baselines. For comparative analysis, we benchmark our method **FedWCA** against several baseline models. **Source Only** uses only pre-trained source models. In terms of SFDA approaches, **Local** and **FedAvg** apply SFDA methods to client-side learning. While ‘Local’ represents a non-federated approach, ‘FedAvg’ incorporates model aggregation by FedAvg[31] at the server. As a foundational SFDA method, SHOT [109] is used in these approaches, with round-fixed pseudo-labels for FedAvg, akin to our method. We also assess against **LADD** [1], an existing FFREEDA method. To examine if existing FL approaches for non-i.i.d. data are applicable to FFREEDA by adding a pseudo-labeling step, we compare our method with **FedPCL+PL**. This method integrates FedPCL [4] with prototype-based pseudo-labeling. FedPCL aligns with our setting as it adapts pre-trained models to clients. See Appendix 5.7.5.3 for comparisons with other FL methods extended to FFREEDA.

Evaluation. We trained five source models using standard cross-entropy loss

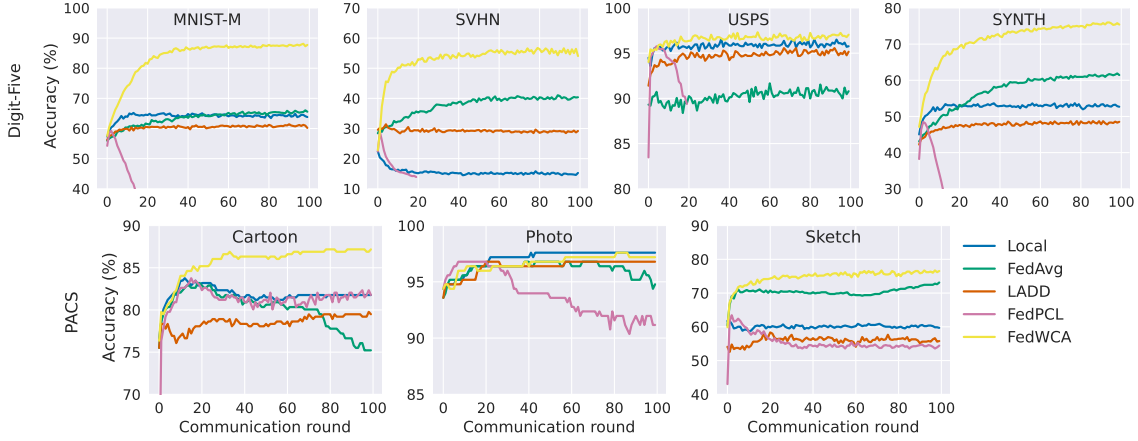


Figure 5.4: Comparison of average accuracy over clients within each target domain. The source domain is MNIST for Digit-Five (top) and Art Painting for PACS (bottom). The target domains are the others. FedWCA considerably enhances the accuracy of clients in specific target domains such as SVHN for Digit-Five and Sketch for PACS, while maintaining the performance of other domains.

without incorporating any target data information to ensure a fair comparison. We then conducted three trials for each method per source model, totaling fifteen trials. The evaluation metric is the average accuracy across all clients, assessed by the final personalized models using client-specific test data. The results provide the mean values of this average accuracy over all trials.

See Appendix 5.7.3 for more detailed experimental settings.

5.5.1 Comparison with Baselines

The comparison across the three datasets (Table 5.2) reveals varied performances. The ‘Source Only’ approach demonstrates limited effectiveness, while the ‘Local’ approach, applying SFDA to local learning, significantly enhances performance. ‘FedAvg’ further improves upon ‘Local’, showcasing the benefits of FL. However, ‘FedAvg’ experiences performance drops in some cases due to domain shift, such as a decrease from 56.76% to 53.23% on USPS for Digit-Five. ‘LADD’ and ‘FedPCL+PL’ mitigate domain shift issues somewhat and improve accuracy in some source domains, but are inferior to ‘FedAvg’ in many domains. This is because LADD’s primary focus is not classification tasks and therefore it uses a simple pseudo-labelling strategy, and furthermore it does not fully utilize knowledge from all domains. On the other hand, FedPCL originally employs true labels at every step and substituting them all with pseudo-labels could compromise performance.

Our FedWCA outperforms the best compared methods by notable margins on

Table 5.3: Ablation on cluster weights calculation. The FedWCA’s cluster weights calculation is replaced by various methods such as one-hot weights (One-hot), equal weights (Equal), a hybrid of them (One-Equal), an adaptive version of One-Equal (One-Equal-A), and a local calculation-only FedWCA variant described in Section 5.4.2 (FedWCA-L). The original FedWCA surpasses all other methods across all PACS source domains.

Methods	PACS				
	Ar	Ca	Ph	Sk	Avg.
One-hot	73.22	77.34	64.45	80.40	73.85
Equal	76.91	76.03	61.16	79.55	73.41
One-Equal	77.89	79.69	64.76	83.13	76.37
One-Equal-A	77.80	79.17	63.64	83.29	75.98
FedWCA-L	77.12	81.94	62.25	82.31	75.91
FedWCA	80.63	83.18	65.50	84.22	78.38

almost all source domains (7.10% for Digit-Five, 4.86% for PACS, 1.32% for Office-Home on average). The results also confirm that FedWCA statistically outperforms the others in a majority of source domains. The improvement on Office-Home is more modest, likely due to the smaller domain gap in this dataset, which can be judged from the fact that the improvement for ‘Source Only’ by ‘Local’ is considerably smaller (5.39%) than for the other datasets (13.31% for Digit-Five and 12.72% for PACS).

In Fig. 5.4, we analyze the changes in average accuracy of clients in each target domain, using specific source domains, such as MNIST for Digit-Five (top) and Art Painting for PACS (bottom). Unlike ‘FedAvg’, FedWCA significantly improves accuracy in almost all target domains without compromising accuracy in specific target domains such as USPS for Digit-Five and Photo for PACS.

5.5.2 Ablation Studies

5.5.2.1 Cluster Weight Calculation

To validate the effectiveness of our cluster weight calculation method, we conducted an ablation study comparing our method with simpler cluster weight computation methods including: (1) **One-hot**: Assigns a weight of 1 to each client’s own cluster and 0 to others. (2) **Equal**: Allocates an equal weight to all clusters, resembling FedAvg. (3) **One-Equal**: A hybrid approach that assigns a weight of p (set to 0.8) to each client’s cluster and distributes the remaining weight equally among other clusters. (4) **One-Equal-A**: An adaptive method that combines models created by

Table 5.4: Ablation on layers used for clustering. Art Painting of PACS is used as the source domain, and ResNet-18 is employed. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. The table below reports the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. Ideally, each client group (C, C', C''), (P, P', P''), and (S, S', S'') should be grouped together, as is achieved by the first and second layers.

Layer	Clients								
	C	C'	C''	P	P'	P''	S	S'	S''
1st	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
2nd	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
3rd	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
4th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$
17th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
All	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$

One-hot and Equal using SND, as in FedWCA. (5) **FedWCA-L**: A variant of FedWCA wherein cluster weights are computed solely locally, described in Section 5.4.2.

The performance comparison of these methods on PACS is shown in Table 5.3. The hybrid approaches, One-Equal and its adaptive variant, and FedWCA-L, increase accuracy compared to simpler methods like One-hot and Equal. FedWCA, however, outperforms them across all source domains. The superior performance illustrates the benefits of our method that integrates global and local insights in cluster weight calculation. See Appendix 5.7.5.1 for examples of specific cluster weights in our methods.

5.5.2.2 Client Clustering Method

Table 5.4 evaluates the clustering performance using different layers’ parameters in ResNet-18, showing that using only the first or second layer is the best practice (see Appendix 5.7.5.2 for full results). The results indicate that clustering performance diminishes as a deeper layer rather than the first layer (e.g., 17th) is used. Using all layers leads to a single cluster (indexed by $\bigcirc$ in Table 5.4) due to the curse of dimensionality.

Despite successful clustering in PACS and Office-Home, some errors were observed in Digit-Five (see Appendix 5.7.5.2). Nonetheless, our method improves performance even with incorrect cluster assignments by effectively combining cluster models with individualized weights (Appendix 5.7.5.2).

Table 5.5: Ablation on pseudo-labeling strategy. Key components of FedWCA are **PP**: prototype-based pseudo-labeling, **FP**: fixed pseudo-labels in each round, **TM**: two-models-based pseudo-labeling using initial and cluster models, and **MU**: mixup. All components prove effective in most PACS source domains.

PP	FP	TM	MU	PACS				
				Ar	Ca	Ph	Sk	Avg.
	✓			72.98	76.44	61.78	68.73	70.29
✓				77.63	75.66	65.48	81.31	74.39
✓	✓			78.37	81.54	65.23	83.56	77.17
✓	✓	✓		78.82	81.72	66.27	83.92	77.68
✓	✓	✓	✓	80.63	83.18	65.50	84.22	78.38

5.5.2.3 Pseudo-labeling Strategy

We examine the contribution of each key component of our pseudo-labeling strategy. The pseudo-labeling strategy of FedWCA consists of four key components (see Section 5.4.3): prototype-based pseudo-labeling (PP), fixed pseudo-labels (FP) in each round, two-models-based pseudo-labeling (TM), which utilizes both the initial and original cluster models, and mixup (MU) between the samples with the matched and mismatched pseudo-labels.

To assess the effectiveness of each component, we incrementally activated them in FedWCA and observed their impact on performance in the PACS dataset. The results, presented in Table 5.5, reveal that the full integration of all components leads to the highest performance across all source domains. This finding underscores the significance of each element, with every component contributing meaningfully to the overall efficacy of the method.

The above results demonstrate that PP and FP independently enhance accuracy and may also be incorporated into LADD. Similarly, the IM loss adopted by our method could contribute to LADD’s expansion, tailoring it more towards the classification task, akin to the local adaptation module with ‘FedAvg’ in our experiments. Table 5.6 compares the performance of the extended LADD with our FedWCA. A comparison between Table 5.2 and Table 5.6 reveals a significant performance boost in the extended LADD over the original. However, FedWCA outperforms this improved LADD, further supporting our method’s efficiency.

5.6 Summary

This chapter introduces FedWCA, a novel method for FFREEDA classification, addressing two problems: lack of labeled data post-pre-training and domain shift

Table 5.6: Comparison with extended LADD. The prior FFREEDA method LADD is extended by incorporating **PP**: prototype-based pseudo-labeling, **FP**: fixed pseudo-labels in each round, and **IM**: IM loss. Nevertheless, FedWCA outperforms it.

Datasets		Methods	
		LADD+PP+FP+IM	FedWCA
Digit-Five	MN	60.45 $\pm$ 2.14	72.74 $\pm$ 3.57
	SV	90.06 $\pm$ 2.41	90.13 $\pm$ 2.08
	MN-M	72.11 $\pm$ 3.06	82.93 $\pm$ 0.79
	US	56.96 $\pm$ 1.67	58.56 $\pm$ 4.26
	SY	87.12 $\pm$ 0.65	84.94 $\pm$ 1.15
	Avg.	73.34 $\pm$ 13.71	77.86 $\pm$ 11.56
PACS	Ar	73.19 $\pm$ 1.73	80.63 $\pm$ 2.44
	Ca	75.98 $\pm$ 2.49	83.18 $\pm$ 0.86
	Ph	65.15 $\pm$ 1.27	65.50 $\pm$ 1.24
	Sk	79.24 $\pm$ 6.76	84.22 $\pm$ 6.60
	Avg.	73.39 $\pm$ 6.41	78.38 $\pm$ 8.38
Office-Home	Ar	65.04 $\pm$ 0.82	66.06 $\pm$ 0.50
	Cl	67.88 $\pm$ 0.67	68.32 $\pm$ 0.42
	Pr	61.06 $\pm$ 0.53	61.46 $\pm$ 0.88
	Re	67.58 $\pm$ 0.80	68.06 $\pm$ 0.46
	Avg.	65.39 $\pm$ 3.14	65.97 $\pm$ 2.83

among clients. Experimental results show FedWCA’s superior performance over several baselines, validating its effectiveness in FFREEDA. We discuss the limitations in Appendix 5.7.4.

5.7 Appendix

5.7.1 Notations

The notations used in this chapter are listed in Table 5.7.

5.7.2 Details of Method

5.7.2.1 Algorithm

Algorithm 4 outlines the comprehensive training procedure for our suggested FedWCA, while Algorithm 5 details the local adaptation process.

5.7.2.2 Soft Neighborhood Density

We review Soft Neighborhood Density (SND) [2] used to assess the overall model efficacy in our FedWCA (see Eq. (5.3)). SND is originally proposed as a method for evaluating models in an unsupervised manner by analyzing how densely data points

Table 5.7: Notation.

Symbol/Notation	Definition
Problem	
$\mathcal{X}$	feature space
$\mathcal{Y}$	label space
M	number of classes indexed by m
K	number of clients indexed by k
R	number of communication rounds indexed by r
E	number of local epochs for each client
$D_k = \{x_i\}_{i=1}^{N_k}$	unlabeled dataset of client k with N_k samples
$P_k(\mathcal{X})$	data distribution of client k
$f: \mathcal{X} \rightarrow \mathbb{R}^q$	feature extractor
$g: \mathbb{R}^q \rightarrow \mathbb{R}^M$	classifier
$h = g \circ f$	classification model
$h_S = g_S \circ f_S$	source model
$W = [w_1, \dots, w_M] \in \mathbb{R}^{q \times M}$	classifier weight for g_S
$h_k = g_k \circ f_k$	locally trained model by client k
Method	
$A(k, l)$	adjacency matrix for client clustering
κ_k	nearest neighbor client for client k
C	number of clusters indexed by c
c_k	cluster index assigned to client k
f_c	cluster model for cluster c
$\tilde{f}_c$	soft cluster model for cluster c : $\tilde{f}_c = B_{c,0}f_c + B_{c,1} \sum_{c'} A_{c' \rightarrow c} f_{c'}$
$\bar{f}_k$	locally combined model by client k using soft cluster models: $\bar{f}_k = \sum_c \alpha_{k,c} \tilde{f}_c$
f_k^{init}	initial model of client k for each round: $f_k^{\text{init}} = \beta_{k,0}f_{c_k} + \beta_{k,1} \sum_c \alpha_{k,c} \tilde{f}_c$
$m(i)$	class whose classifier weight vector is closest to $f_c(x_i)$: $m(i) = \arg \max_m \cos(f_c(x_i), w_m)$
$v_k = (v_{k,1}, \dots, v_{k,C})$	cluster weights for cluster models: $f_k^{\text{init}} = \sum_c v_{k,c} \tilde{f}_c$
$\alpha_k = (\alpha_{k,1}, \dots, \alpha_{k,C})$	locally calculated cluster weights for soft cluster models
$\beta_k = (\beta_{k,0}, \beta_{k,1})$	pseudo-performance weights for f_{c_k} and $\bar{f}_k$
$A_{c' \rightarrow c}$	benefit metric indicating the relative advantage of cluster c for clients in cluster c'
$B_{c,0}, B_{c,1}$	coefficients balancing the emphasis between f_c and $\sum_{c'} A_{c' \rightarrow c} f_{c'}$
T_a, T_b	temperature parameters controlling α_k and β_k , respectively
λ, μ	balancing parameters for the loss function and mixup, respectively
$\hat{D}_k = \{x_i, \hat{y}_i\}_{i=1}^{N_k}$	pseudo-labeled dataset of client k obtained by a prototype-based pseudo-labeling and mixup

cluster together using the overall models' outputs. It defines 'soft neighborhoods' of a data point by the distribution of its similarity to other points, and measures density as the entropy of this distribution.

Let $h = g \circ f$ and $D = \{x_i\}_{i=1}^N$ be an evaluated model and an unlabeled dataset, respectively. SND first computes the similarity between samples. Let $Q_{i,j} = \cos(h(x_i), h(x_j))$ be the (i, j) element of the similarity matrix, where $\cos(\cdot, \cdot)$ is the cosine similarity. The diagonal elements of Q are ignored to compute the distance to neighbors for each sample. Q is then converted into a probabilistic distribution P using the scaling temperature parameter T and the softmax function:

$$P_{i,j} = \frac{\exp(Q_{i,j}/T)}{\sum_{j'} \exp(Q_{i,j'}/T)}. \quad (5.5)$$

The temperature is set to 0.05 in the original paper and we use it. We finally obtain the SND value of h by computing the entropy for each row of P (i.e., each sample)

and taking the average of all samples:

$$S(h) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N P_{i,j} \log P_{i,j}. \quad (5.6)$$

See the original paper for detailed explanation.

5.7.2.3 Cluster Weights

We provide the final specific cluster weights $\mathbf{v}_k = (v_{k,1}, \dots, v_{k,C})$ in the initial model $f_k^{\text{init}} = \sum_c v_{k,c} f_c$. According to Section 5.4.2, the initial model for client k is computed as follows:

$$\begin{aligned} f_k^{\text{init}} &= \beta_{k,0} f_{c_k} + \beta_{k,1} \sum_c \alpha_{k,c} \tilde{f}_c \\ &= \beta_{k,0} f_{c_k} + \beta_{k,1} \sum_c \alpha_{k,c} \{B_{c,0} f_c + B_{c,1} \sum_{c'} A_{c' \rightarrow c} f_{c'}\} \\ &= \sum_c \{\mathbb{1}(c = c_k) \beta_{k,0} + \beta_{k,1} \alpha_{k,c} B_{c,0} + \beta_{k,1} \sum_{c'} B_{c',1} \alpha_{k,c'} A_{c \rightarrow c'}\} f_c, \end{aligned} \quad (5.7)$$

where $\mathbb{1}$ is the indicator function. The third equation can be obtained by replacing the subscripts c and c' . We thus obtain the cluster weights $\mathbf{v}_k$ as:

$$v_{k,c} = \mathbb{1}(c = c_k) \beta_{k,0} + \beta_{k,1} \alpha_{k,c} B_{c,0} + \beta_{k,1} \sum_{c'} B_{c',1} \alpha_{k,c'} A_{c \rightarrow c'}. \quad (5.8)$$

5.7.2.4 Loss Function of SHOT

We review the loss function of SHOT [109], adopted as a loss function of our method because of its simplicity. The loss function comprises the cross-entropy loss for the pseudo-labeled dataset and an Information Maximization (IM) loss [123] for the unlabeled dataset. The IM loss promotes confident model outputs and counteracts a bias towards any single class by discouraging trivial score distributions. The total empirical loss function $\widehat{\mathcal{L}}(D; h)$ for the model h on D weights these two loss functions

Table 5.8: Dataset information including number of samples, classes, and clients for three datasets.

Datasets	Number of samples		Classes	Clients
	Per domain	Total		
Digit-Five	MNIST: 34,000, MNIST-M: 34,000 SVHN: 34,000, SYNTH: 34,000, USPS: 9298	145,298	10	32
PACS	Art Painting: 2,048, Cartoon: 2,344 Photo: 1,670, Sketch: 3,929	9,991	7	9
Office-Home	Art: 2,424, Clipart: 4,365 Product: 4,437, Real-World: 4,353	15,579	65	9

with a balancing parameter λ :

$$\begin{aligned}
\widehat{\mathcal{L}}(D; h) &= \widehat{\mathcal{L}}_{\text{IM}}(D; h) + \lambda \widehat{\mathcal{L}}_{\text{CE}}(\hat{D}; h); \\
\widehat{\mathcal{L}}_{\text{CE}}(\hat{D}; h) &= -\frac{1}{N} \sum_{(x, \hat{y}) \in \hat{D}} \log(h(x))_{\hat{y}}, \\
\widehat{\mathcal{L}}_{\text{IM}}(D; h) &= \sum_{m=1}^M \hat{o}_m \log \hat{o}_m - \frac{1}{N} \sum_{m=1}^M \sum_{x \in D} (h(x))_m \log(h(x))_m,
\end{aligned} \tag{5.9}$$

where N is the dataset size and $\widehat{\mathcal{L}}_{\text{CE}}(\hat{D}; h)$ represents the cross-entropy loss calculated over the pseudo-labeled dataset $\hat{D}$, which encourages the model h to align with the pseudo-labels. The IM loss $\widehat{\mathcal{L}}_{\text{IM}}(D; h)$, on the other hand, comprises two terms: (1) the negative entropy of the mean output score $\hat{o} = \sum_{x \in D} h(x)/N$ and (2) the entropy of the model’s output scores for each data point in the unlabeled dataset D .

5.7.3 Detailed Experimental Settings

5.7.3.1 Datasets

Table 5.8 summarizes the details of datasets used in our experiments. For Digit-Five, we used the entire dataset for USPS and 34,000 randomly selected samples for the other domains. For each dataset, one domain was selected as the source, and the remaining domains served as target domains. Data samples from each target domain were distributed to 8 clients for Digit-Five and 3 clients for PACS and Office-Home per domain, all in an i.i.d. manner. Each client’s data were divided into three subsets: 20% for testing, 64% for training, and 16% for validation.

Table 5.9: Hyperparameters for our FedWCA including the learning rate (lr), balancing parameter for cross-entropy loss λ , balancing parameter for mixup μ , and temperature parameters T_a , T_b . Only for SYNTH of Digit-Five and Clipart of Office-Home source domains, T_a is set to 0.001 and 0.1, respectively.

Datasets	Hyperparameters				
	lr	λ	μ	T_a	T_b
Digit-Five	0.001	0.1	0.55	0.01	0.05
PACS	0.0001	0.3	0.55	0.1	0.05
Office-Home	0.0001	0.3	0.55	0.01	0.05

5.7.3.2 Implementation Details

For the implementation, we set the local epoch to $E = 5$ and total communication rounds to $R = 100$ for Digit-Five and PACS, and $R = 50$ for Office-Home, ensuring convergence of each method’s learning. In the case of FedPCL+PL, we limited R to 20, due to overfitting observed in this method on the PACS and Office-Home datasets. The stochastic gradient descent (SGD) was used with the best learning rate of 10^{-3} for Digit-Five and 10^{-4} for PACS and Office-Home in all methods, which are selected from $\{10^{-i} | i \in \{1, 2, 3, 4, 5\}\}$. For source model training, we set the learning rate to 0.001 in all datasets. Additionally, we adopted a weight decay of 0.001 and a momentum of 0.9, in line with standard SFDA studies [109]. In LADD’s implementation, we randomly selected 100 samples from each client’s target data for computing style features, focusing on the central 5% region of the frequency spectrum. Other hyperparameters for LADD were also appropriately tuned. The temperature parameters of FedPCL were set to 70 as a result of tuning. For our methods, the temperature parameters were set to $T_b = 0.05$ for all datasets (aligned with SND [2]) and $T_a = 0.01$ for Digit-Five and Office-Home, except for SYNTH ($T_a = 0.001$) and Clipart ($T_a = 0.1$) source domains, while $T_a = 0.1$ for PACS. The balancing parameter λ was assigned the same value as in the original SHOT paper[109]. In particular, Table 5.9 summarizes the hyperparameters for FedWCA.

5.7.3.3 Hyperparameter Tuning

For LADD, we searched the regularization parameters and starting rounds from $\{10^{-i} | i \in \{-2, -1, 0, 1, 2, 3, 4, 5\}\}$ and $\{5, 10, 30, 50, 80, 100\}$, respectively. For FedPCL, the best temperature parameter was selected from $\{7 \times 10^{-i} | i \in \{-2, -1, 0, 1, 2, 3\}\}$ as par the original paper. For our FedWCA, we searched the temperature parameters T_a and T_b from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$.

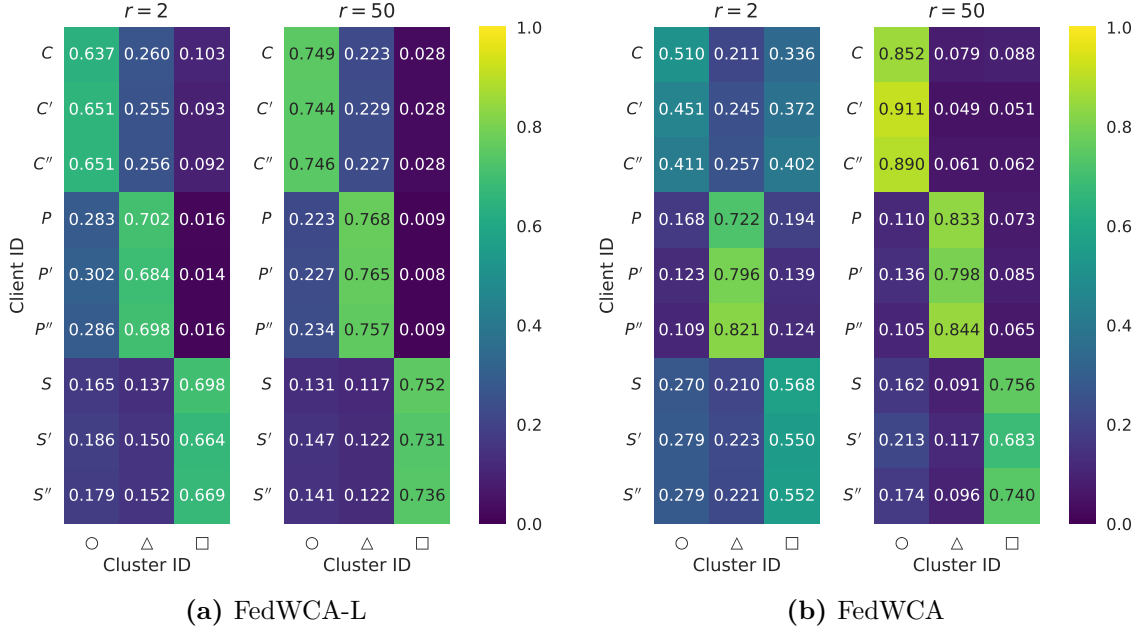


Figure 5.5: Visualization of cluster weights for FedWCA-L and FedWCA when the communication round r is 2 and 50. Art Painting in PACS dataset is used for the source dataset. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. See the first row of Table 5.4 for the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. The number represents the weight for the corresponding cluster.

5.7.4 Limitations and Discussions

Complexity of FedWCA. We mention that the derivation of cluster weights in our method, while not computationally intensive, involves a complex procedure. Although Section 5.5.2 demonstrates that simpler weight calculations do not provide sufficient performance, identifying simpler yet effective methods for calculating cluster weights that match FedWCA’s performance remains a task for future research. However, our method can be slightly modified to reduce computational costs, as discussed below.

Concerns about distributing all cluster models. Our method requires each client to receive all soft cluster models, which may raise some concerns: (1) privacy concerns and (2) additional communication and storage costs.

(1) Distributing cluster models poses minimal privacy risks due to several factors. Initially, we note that the server distributes all “soft cluster models” to each client, along with the original cluster model of the respective client. Soft cluster models, produced by merging all cluster models, roughly incorporate all clients’ local models and thus, privacy risks remain similar to typical FL methods such as FedAvg. Further, distributing the single original cluster model to clients is a standard practice

Table 5.10: Results for FedWCA modification reducing costs. The numbers are the mean values $\pm$ standard deviations of the averaged accuracy (%) across all clients and all target domains. In the latter $U - 1$ rounds of every U round, the initial model of the FedWCA can be computed server-side as opposed to client-side, thereby cutting communication, storage, and computational costs. This revised FedWCA boasts an equivalent performance to its original counterpart when $U = 5$.

Datasets		Methods	
		Revised FedWCA $U = 5$	FedWCA
Digit-Five	MN	73.37 ± 3.65	72.74 ± 3.57
	SV	89.88 ± 2.24	90.13 ± 2.08
	MN-M	83.16 ± 0.56	82.93 ± 0.79
	US	57.33 ± 3.00	58.56 ± 4.26
	SY	85.70 ± 1.05	84.06 ± 1.38
	Avg.	77.89 ± 11.93	77.69 ± 11.47
PACS	Ar	80.89 ± 2.54	80.63 ± 2.44
	Ca	83.23 ± 0.91	83.18 ± 0.86
	Ph	65.32 ± 1.06	65.50 ± 1.24
	Sk	84.28 ± 6.40	84.22 ± 6.60
	Avg.	78.43 ± 8.45	78.38 ± 8.38
Office-Home	Ar	66.13 ± 0.67	66.06 ± 0.50
	Cl	68.33 ± 0.61	68.32 ± 0.42
	Pr	61.30 ± 0.60	61.46 ± 0.88
	Re	67.63 ± 0.29	68.06 ± 0.46
	Avg.	65.85 ± 2.86	65.97 ± 2.83

in current clustered FL studies and carries minimal privacy risk. This is due to our algorithm’s design, where clients are only privy to their specific cluster IDs with no insight into other cluster details such as cluster size or client membership, even if it is composed of only two clients.

(2) Our weighted cluster aggregation (WCA) costs $C + 1$ times higher than FedAvg (C : number of clusters) due to the requirement of each client receiving and storing C soft cluster models and the original cluster model. However, we can save those costs by altering the computation of the client’s initial models in WCA as follows. In the first round of every U rounds, clients compute their initial models as the original method does. In the following $U - 1$ rounds, the initial models are computed server-side, utilizing identical cluster weights sent by clients in the first round. This modification allows the server to only distribute two models to each client: the computed initial model and the original cluster model, reducing the communication and storage costs by a factor of $2/(C + 1)$ in $(U - 1)/U$ of all rounds. This can also lessen the client’s computational cost for initial model calculations. As demonstrated in Table 5.10, further tests on Digit-Five, PACS, and Office-Home with $U = 5$ showed that the revised FedWCA maintains similar average accuracy as

the original. Notably, in some source domains like SYNTH, the revised technique improves accuracy. This enhancement is because the use of identical cluster weights reduces overfitting.

Different enhancement according to dataset. As illustrated in Section 5.5.1, the accuracy enhancement achieved by our method varies with the dataset. Specifically, for datasets exhibiting minor domain gaps, FedAvg may suffice to some extent, as the impact of domain shifts on accuracy is minimal. For instance, in the Office-Home dataset, although our method surpasses FedAvg, the superiority margin is less significant than in Digit-Five and PACS.

Extension to other tasks. While our approach is primarily tailored for classification tasks, its underlying principles could theoretically extend to other complex vision tasks, like object detection. However, direct extrapolation may encounter challenges, such as convergence of feature vectors towards classifier vectors in cluster weight calculations. Adapting and assessing our method in these diverse contexts constitutes an avenue for future research.

5.7.5 Supplementary Experimental Results

5.7.5.1 Visualization of Cluster Weights

Fig. 5.5 visualizes the cluster weights for our proposed methods FedWCA and FedWCA-L, a variant of FedWCA wherein cluster weights are computed solely locally (see Section 5.4.2). This shows that clients within the same domain obtain similar weights, indicating the efficacy of our cluster weight calculation based on Eq. (5.1). FedWCA, in particular, promotes inter-domain collaboration early in learning ($r = 2$), notably between Cartoon and Sketch domains. Specifically, Cartoon domain clients (C, C', C'') have significantly higher weights for cluster $\square$ (Sketch) in FedWCA compared to FedWCA-L. This is the result of the Cartoon domain clients considering the benefits for the clients in cluster $\square$. As learning progresses ($r = 50$), FedWCA clients increasingly concentrate on their respective clusters, highlighting the method’s balance between overall and individual advantages. As a result, this approach leads to notable accuracy improvements for both Cartoon and Sketch clients, as depicted in Fig. 5.4 (b).

Table 5.11: Full results for layer dependence of clustering. Art Painting of PACS is used as the source domain, and ResNet-18 is employed. Clients C, C', C'' belong to Cartoon, P, P', P'' to Photo, and S, S', S'' to Sketch. The table below reports the cluster IDs (denoted by $\bigcirc$, $\triangle$, and $\square$) assigned to each client. Ideally, each client group (C, C', C''), (P, P', P''), and (S, S', S'') should be grouped together, as is achieved by the first and second layers.

Layer	Clients								
	C	C'	C''	P	P'	P''	S	S'	S''
1st	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
2nd	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
3rd	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
4th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$
5th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
6th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
7th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
8th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
9th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
10th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$
11th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
12th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$
13th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
14th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$	$\square$	$\square$	$\square$
15th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
16th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\triangle$	$\triangle$	$\triangle$
17th	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
All	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$

5.7.5.2 Results of Client Clustering

Table 5.11 shows the full results for the layer dependence of client clustering in our method with the PACS dataset.

Additionally, Table 5.12 presents the clustering results for Digit-Five across various source domains using our FINCH method. Fig. 5.6 shows the specific images possessed by clients in each cluster particularly when using USPS as the source domain. The results from Table 5.12 indicate that, unlike PACS and Office-Home, the clustering for Digit-Five contains some errors, except for the SVHN source domain. To this end, we compare our standard FINCH algorithm, against two alternatives on Digit-Five: (1) LADD clustering based on the shared style features and (2) a hypothetical ‘true’ clustering based on actual domain information. It must be emphasized that LADD requires a specified number of iterations and a search range for clusters, whereas FINCH necessitates no hyperparameters. Table 5.13 shows the accuracy of FedWCA when applying each clustering algorithm. Notably, FINCH performs comparably to both the true clustering and LADD, with no need for sharing prior information on the clusters or any information other than the model parameters.

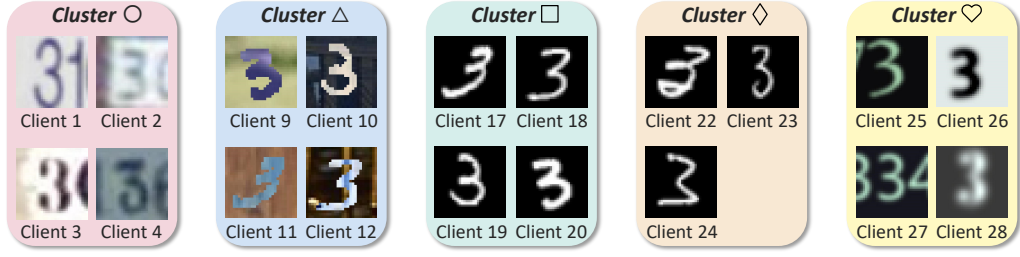


Figure 5.6: Images possessed by clients in each cluster. USPS of Digit-Five serves as the source domain. Our FedWCA clusters clients according to their local models, almost matching their data domains.

This highlights our method’s strength in enhancing client performance, even when assigned to incorrect clusters, by effectively combining all cluster models using individualized cluster weights.

5.7.5.3 Comparison with Other Federated Learning Methods

In addition to FedAvg and FedPCL, we extended other FL methods using ground-truth labels to FFREEDA by incorporating a prototype-based pseudo-labeling, fixed pseudo-labels per round, and IM loss, and compared them to our method. We adopted three additional FL methods: (1) FedProx [5], (2) FedAMP [6], and (3) IFCA [7]. FedProx modifies FedAvg by adding the proximal term to address data heterogeneity. FedAMP personalizes FL by creating a client-specific cloud model weighted on the similarity of each client’s model parameters, with clients subsequently training personalized models based on this cloud model. IFCA, a clustered FL method that requires setting the number of clusters, has each client calculate the loss for all cluster models and train the model with the lowest loss per round.

The performance comparison is shown in Table 5.14. Our FedWCA outperforms all other methods in all source domains for Digit-Five and PACS, highlighting that merely extending existing FL methods to FFREEDA is insufficient and confirming our method’s effectiveness. FedProx uses basic averaging for aggregation, FedAMP’s parameter similarity weighting is unsuitable for our unlabeled adaptation, and IFCA’s cluster model selection is unstable due to the loss calculation based on pseudo-labels.

Table 5.12: Results of client clustering in Digit-Five for each source domain. The clustering algorithm FINCH is applied to Lenet. Each row represents client IDs. Clients 1 to 8, 9 to 16, 17 to 24 and 25 to 32 each belong to the same domain. The table below reports the cluster IDs (denoted by, e.g., $\bigcirc$, $\triangle$, $\square$, and $\diamond$) assigned to each client. Ideally, each client group (1,2,3,4,5,6,7,8), (9,10,11,12,13,14,15,16), (17,18,19,20,21,22,23,24), and (25,26,27,28,29,30,31,32) should be grouped together, as is achieved by SVHN source domain.

Client ID	Source domains				
	MN	SV	MN-M	US	SY
1	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
2	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
3	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
4	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
5	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
6	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
7	$\triangle$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
8	$\triangle$	$\bigcirc$	$\bigcirc$	$\bigcirc$	$\bigcirc$
9	$\square$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
10	$\square$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
11	$\square$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
12	$\square$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
13	$\diamond$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
14	$\diamond$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
15	$\heartsuit$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
16	$\heartsuit$	$\triangle$	$\triangle$	$\triangle$	$\triangle$
17	$\star$	$\square$	$\square$	$\square$	$\square$
18	$\star$	$\square$	$\square$	$\square$	$\square$
19	$\star$	$\square$	$\square$	$\square$	$\square$
20	$\star$	$\square$	$\square$	$\square$	$\square$
21	$\star$	$\square$	$\square$	$\square$	$\diamond$
22	$\star$	$\square$	$\diamond$	$\diamond$	$\diamond$
23	$\clubsuit$	$\square$	$\diamond$	$\diamond$	$\diamond$
24	$\clubsuit$	$\square$	$\diamond$	$\diamond$	$\diamond$
25	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
26	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
27	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
28	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
29	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
30	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
31	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$
32	$\spadesuit$	$\diamond$	$\heartsuit$	$\heartsuit$	$\triangle$

Table 5.13: Ablation on clustering algorithm. Three clustering algorithms are applied to FedWCA. ‘True’ uses the hypothetical true clusters based on the domain labels. Even without specific cluster information, FedWCA offers similar performance to ‘True’.

Clustering	Digit-Five					
	MN	SV	MN-M	US	SY	Avg.
FINCH	72.74	90.13	83.93	58.56	84.94	77.86
LADD	72.33	89.89	81.99	58.72	85.58	77.70
True	72.93	90.07	83.43	56.95	85.69	77.81

Table 5.14: Performance comparison with extended FL methods (Digit-Five and PACS). The existing FL methods, FedProx [5], FedAMP [6], and IFCA [7] are extended to FFREEDA by incorporating **PP**: prototype-based pseudo-labeling, **FP**: fixed pseudo-labels in each round, and **IM**: IM loss. FedWCA outperforms them in all source domains.

Datasets		Methods (+PP+FP+IM)			
		FedProx	FedAMP	IFCA	FedWCA (ours)
Digit-Five	MN	56.01 $\pm$ 2.46	51.48 $\pm$ 0.44	58.68 $\pm$ 2.21	72.74 $\pm$ 3.57
	SV	83.27 $\pm$ 1.80	83.00 $\pm$ 0.81	83.39 $\pm$ 1.60	90.13 $\pm$ 2.08
	MN-M	81.27 $\pm$ 0.29	72.94 $\pm$ 2.09	79.82 $\pm$ 0.40	82.93 $\pm$ 0.79
	US	52.78 $\pm$ 0.74	53.81 $\pm$ 1.54	55.62 $\pm$ 2.99	58.56 $\pm$ 4.26
	SY	82.21 $\pm$ 0.54	82.22 $\pm$ 0.32	81.71 $\pm$ 0.75	84.94 $\pm$ 1.15
	Avg.	71.11 $\pm$ 13.86	68.69 $\pm$ 13.73	71.81 $\pm$ 12.30	77.86 $\pm$ 11.56
PACS	Ar	76.50 $\pm$ 1.24	73.54 $\pm$ 1.30	75.82 $\pm$ 3.11	80.63 $\pm$ 2.44
	Ca	76.16 $\pm$ 0.87	77.55 $\pm$ 1.21	79.13 $\pm$ 2.57	83.18 $\pm$ 0.86
	Ph	60.88 $\pm$ 0.74	63.70 $\pm$ 0.79	64.61 $\pm$ 1.57	65.50 $\pm$ 1.24
	Sk	80.29 $\pm$ 7.02	75.84 $\pm$ 5.87	80.93 $\pm$ 6.59	84.22 $\pm$ 6.60
	Avg.	73.46 $\pm$ 8.29	72.66 $\pm$ 6.19	75.13 $\pm$ 7.46	78.38 $\pm$ 8.38

Algorithm 4: FedWCA (Federated learning with weighted cluster aggregation)

Input: K clients, k -th client's local unlabeled data $D_k = \{x_i\}_{i=1}^{N_k}$, k -th client's initial local model $h_k^{\text{init}} = g_k^{\text{init}} \circ f_k^{\text{init}}$, source model $h_S = g_S \circ f_S$, local epochs E , total communication round R , temperature parameters T_a, T_b , balancing parameters λ, μ

Output: K personalized models $h_k = g_S \circ f_{c_k}$

```

1 for  $r = 0, \dots, R - 1$  do
2   if  $r = 0$  then
3     Client:
4       Initialize the local model:  $h_k^{\text{init}} = g_k^{\text{init}} \circ f_k^{\text{init}} \leftarrow h_S = g_S \circ f_S$ 
5        $f_k = \text{ClientLocalAdaptation}(r, D_k, f_k^{\text{init}}, g_k^{\text{init}}, E, \lambda)$ 
6       Send  $f_k$  to the server
7     Server:
8       Cluster  $K$  clients into  $C$  clusters based on the first layers'
       parameters of  $f_k$  by using FINCH:  $k \mapsto c_k$ 
9       Create cluster models  $f_c$  by averaging the local models  $f_k$  in each
       cluster  $c \in \{1, \dots, C\}$ 
10      Initialize soft cluster models  $\tilde{f}_c = f_c$ 
11   else
12     Client:
13       Calculate  $\alpha_k$  for  $\tilde{f}_c$  and  $\beta_k$  for  $f_{c_k}$  and  $\bar{f}_c = \sum_c \alpha_{k,c} \tilde{f}_c$  by Eq. (5.1)
       and Eq. (5.3)
14       Set an initial model:  $f_k^{\text{init}} = \beta_{k,0} f_{c_k} + \beta_{k,1} \sum_c \alpha_{k,c} \tilde{f}_c$ 
15        $f_k = \text{ClientLocalAdaptation}(r, D_k, f_k^{\text{init}}, f_{c_k}, g_k^{\text{init}}, E, T_a, T_b, \lambda,$ 
        $\mu)$ 
16       Send  $f_k, \alpha_k$ , and  $\beta_k$  to the server
17     Server:
18       Update cluster models  $f_c$  by averaging the local models  $f_k$  in each
       cluster  $c \in \{1, \dots, C\}$ 
19       Calculate  $A_{c \rightarrow c'}$  and  $B_{c,i}$  by averaging  $\alpha_{k,c'}$  and  $\beta_{k,i}$  across clients
       within each cluster  $c$ 
20       Update soft cluster models:  $\tilde{f}_c = B_{c,0} f_c + B_{c,1} \sum_{c' \in \mathcal{C}} A_{c' \rightarrow c} f_{c'}$ 
21   Server: Send  $f_{c_k}$  and  $\tilde{f}_c$  for every  $c$  to the client  $k$ 

```

Algorithm 5: ClientLocalAdaptation

Input: Current round r , unlabeled data $D = \{x_i\}_{i=1}^N$, initial model f^{init} , cluster model f_c , classifier g , local epochs E , temperature parameters T_a, T_b , balancing parameters λ, μ .

Output: Trained model f

```

1 if  $r = 0$  then
2   Compute class-wise prototypes  $p_m$  for each class  $m$  by  $g \circ f^{\text{init}}$  and
   assign each sample  $x \in D$  pseudo-labels  $\hat{y}$  via Eq. (5.4), and generate a
   pseudo-labeled dataset  $\hat{D} = \{x_i, \hat{y}_i\}_{i=1}^N$ 
3 else
4   Compute class-wise prototypes  $p_m$  and  $q_m$  for each class  $m$  by  $g \circ f^{\text{init}}$ 
   and  $g \circ f_c$ , and assign each sample  $x \in D$  pseudo-label  $\hat{y}_{\text{init}}$  and  $\hat{y}_c$  via
   Eq. (5.4), respectively
5   Compute normalization factors  $p, q$ :


$$p = \sum_{m \neq m'} \frac{\cos(p_m, p_{m'})}{M(M-1)}, \quad q = \sum_{m \neq m'} \frac{\cos(q_m, q_{m'})}{M(M-1)}$$


6   Assign each sample  $x \in D$  pseudo-label  $\hat{y} = \hat{y}_{\text{init}}$  if
    $\cos(f^{\text{init}}(x), p_{\hat{y}_{\text{init}}})/p \geq \cos(f_c(x), q_{\hat{y}_c})/q$ , and  $\hat{y} = \hat{y}_c$  otherwise
7   Generate pseudo-labeled datasets  $\hat{D}^{\text{mat}}$  and  $\hat{D}^{\text{mis}}$ :


$$\hat{D}^{\text{mat}} = \{(x, \hat{y}) | x \in D \text{ s.t. } \hat{y}_{\text{init}} = \hat{y}_c\}, \quad \hat{D}^{\text{mis}} = \{(x, \hat{y}) | x \in D \text{ s.t. } \hat{y}_{\text{init}} \neq \hat{y}_c\}$$


8   Generate a mixed dataset  $\hat{D}^{\text{mix}}$  and a pseudo-labeled dataset
 $\hat{D} = \hat{D}^{\text{mat}} \cup \hat{D}^{\text{mix}}$ :


$$\hat{D}^{\text{mix}} = \{((1-\mu)x + \mu x', \hat{y}) | (x, \hat{y}) \in \hat{D}^{\text{mis}}, (x', \hat{y}) \sim \hat{D}^{\text{mat}} \text{ for each } (x, \hat{y})\}$$


9 Initialize a model:  $f = f^{\text{init}}$ 
10 for  $e = 0, \dots, E-1$  do
11   Update  $f$  with SGD by minimizing the loss function

$$\hat{\mathcal{L}}(D; g \circ f) = \hat{\mathcal{L}}_{\text{IM}}(D; g \circ f) + \lambda \hat{\mathcal{L}}_{\text{CE}}(\hat{D}; g \circ f)$$

   defined in Eq. (5.9) while
   fixing  $g$ 
12 return  $f$ 

```

Chapter 6

Heterogeneous Domain Adaptation with Positive and Unlabeled Data

6.1 Introduction

Applying machine learning models trained with a domain to different domains may lead to performance degradation [130]. This motivates researchers to study *unsupervised domain adaptation* (UDA) [18, 131], which transfers knowledge from a known domain (source) where a sufficiently large number of labeled samples are available to an unknown domain (target) where only unlabeled samples are available. Conventional UDA typically assumes *covariate shift* and a shared feature space between the two domains. In practice, however, the observable feature space itself may differ across domains (e.g., different attributes are collected), which corresponds to *observation heterogeneity* as categorized in Chapter 2. *Heterogeneous unsupervised domain adaptation* (HUDA) [132] relaxes the shared-feature assumption by allowing different feature spaces across domains. Existing HUDA studies [64–66, 69, 70, 133] only consider situations where both positive and negative examples in a binary classification task are available in the source domain, as illustrated in Figure 6.1 (a). However, in some real-world applications, such as the example below, the source domain may have access to only positive examples (extreme *label shift*).

Estimation of potential customers in different industries. Consider a business alliance of a bank and a jewelry store. The goal of them is to find potential customers for the jewelry store among the bank’s large customer base. This task can

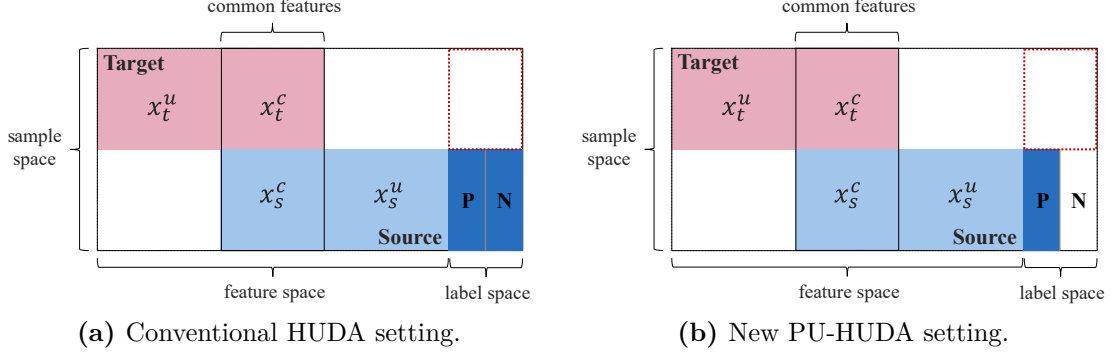


Figure 6.1: View of the data structure in two HUDA settings, (a) conventional one and (b) new one where the source domain has only positive data. Here, both of them assume that the two domains share some features.

be considered a simple classification problem of predicting whether a given customer in the bank’s database will purchase jewelry or not. To address this task, they use their own customer data represented by the following features. First, they both might possess basic demographics such as age and gender. On the other hand, each industry has their own features. The bank has detailed economic conditions represented by deposit balances, and the jewelry store has past purchase histories. Hence, this task can be considered a HUDA task with the jewelry store as the source and the bank as the target (see Figure 6.1). However, while the bank (target) do not have information on purchases of jewelry, i.e., only has unlabeled customers, unlike a conventional HUDA setting, the jewelry store (source) has only customers who have already purchased or visited the jewelry store, i.e., positive labeled customers.

Motivated by the above application, this chapter addresses a new challenging HUDA task called *positive and unlabeled heterogeneous unsupervised domain adaptation* (PU-HUDA), which consists of a source domain with positive data and a target domain with unlabeled data in the different feature spaces as described in Figure 6.1 (b). PU-HUDA can be viewed as a hybrid setup of positive unlabeled (PU) learning [71], which trains binary classifiers from positive and unlabeled data only, and HUDA. It appears that simply combining existing HUDA and PU learning methods can properly solve PU-HUDA, but it does not work well, as we demonstrate later in our experiments. This is because applying existing HUDA methods directly to PU-HUDA minimizes the divergence between the distributions of the whole source data with positive labels only and the whole target data with positive and negative labels on the transformed feature space, resulting in a less label-discriminative feature space on the target domain as shown in Figure 6.2 (top). To solve PU-HUDA

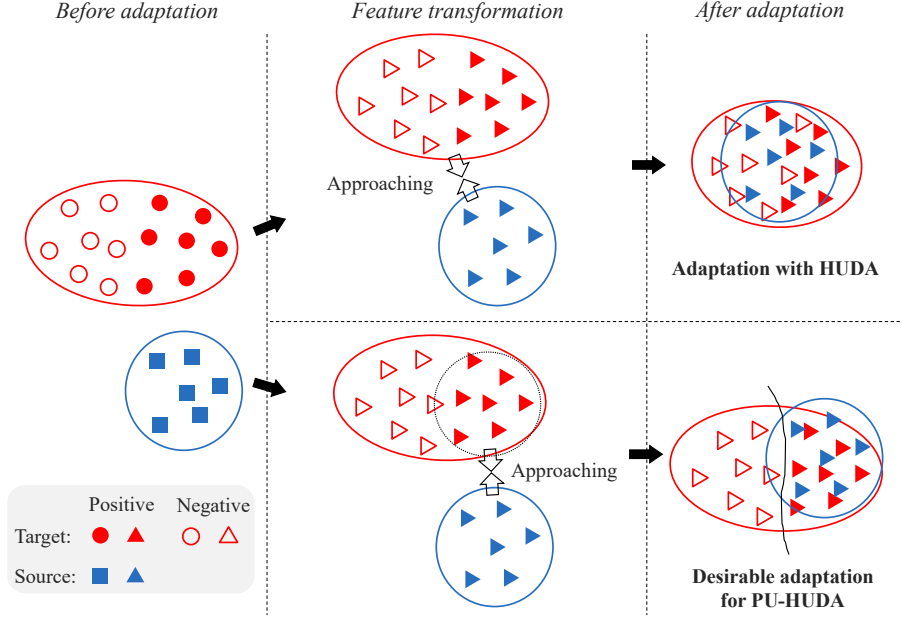


Figure 6.2: Comparison of adaptation mechanisms in existing HUDA methods (top) and our proposed PU-HUDA method (bottom).

properly, we need to minimize the distribution divergence between the whole source data and the target data with only positive labels on the transformed feature space as described in Figure 6.2 (bottom).

To address the above issue, we regard PU-HUDA as a task to identify likely positive examples from target data and reduce the distribution divergence between the whole source examples and the target examples likely to be positive. To solve this task, we propose a new method using adversarial training as in Generative Neural Network (GAN) [134]. Our proposed method mainly consists of three models, (1) a feature transformer to transform the target feature space to that of the source, (2) a classifier to predict likely positive examples from the unlabeled target data transformed by the feature transformer, and (3) a discriminator to determine whether the likely positive examples transformed by the feature transformer are from the target or source domain. The classifier is trained to identify the likely positive examples indistinguishable from the source examples by the discriminator. The feature transformer is trained to transform the target examples predicted as positive by the classifier into the indistinguishable examples from the source examples by the discriminator. To achieve these training, we adopt an objective function based on Kullback-Leibler (KL) divergence inspired by Predictive Adversarial Networks (PAN) [135], a recently proposed PU learning method. We call this approach *pre-*

Table 6.1: Setting comparison of this chapter with existing HUDA methods.

Method	Some common samples are necessary?	Some common features are necessary?	Source negative examples are necessary?
KCCA [64]	Yes	No	Yes
HHTL [65]	Yes	No	Yes
DSFT [66]	No	Yes	Yes
OT-based [133]	No	Yes	Yes
SFER [70]	No	No	Yes
GLG [69]	No	No	Yes
PADA(this chapter)	No	Yes	No

dictive adversarial domain adaptation (PADA). PADA can make the positive target and the whole source data closer while keeping the negative target and the whole source data away, as shown in Figure 6.2 (bottom). As a result, PADA obtains a classifier that provides good classification accuracy on the feature space transformed by a feature transformer.

Our contributions can be summarized as follows:

- We define a new domain adaptation task, *positive and unlabeled heterogeneous unsupervised domain adaptation* (PU-HUDA). To the best of our knowledge, this study is the first to address PU-HUDA.
- We propose a novel method using adversarial training, *positive-adversarial domain adaptation* (PADA). The PADA’s objective is based on KL divergence inspired by PAN [135].
- We experimentally show that PADA outperforms several baseline methods to solve PU-HUDA.

6.2 Related Works

We review previous studies related to the topic of this chapter and discuss how different our work is from them.

6.2.1 Heterogeneous Unsupervised Domain Adaptation

The common idea of existing HUDA techniques is to learn feature transformers to transform the source and target data into a homogeneous feature space to minimize the distribution distance of the two domains on that feature space. Due to this difficulty of dealing with two heterogeneous feature spaces, most HUDA techniques require some supplementary information to bridge the two domains. For example,

KCCA [64] and HHTL [65] require paired (common) samples between the two domains, which are often difficult to obtain in real cases. DSFT [66] and OT(optimal transport)-based method [133] require common features between the two domains as in this chapter. CL-SCL [67] and FSR [68] require semantically equivalent word pairs and meta-features, respectively. There are some studies that do not explicitly require supplementary information, such as GLG [69] and SFER [70] although they implicitly assume that the two domains have sufficiently similar features. All of the methods listed here require both positive and negative examples in the source domain. A comparison of this chapter with these methods is summarized in Table 6.1.

6.2.2 PU Learning

PU learning trains a binary classification from positive and unlabeled examples without labeled negative examples [71]. Early PU learning methods adopt the two-step technique that first identifies reliable negative examples and then conducts supervised learning [136, 137]. Some studies propose PU learning methods by considering unlabeled data as negative data having label noise [138, 139]. Another promising branch of PU learning employs the framework of cost-sensitive learning, such as uPU [140] and nnPU [141]. Recently, Hu et al. [135] proposed the state-of-the-art PU learning method, Predictive Adversarial Networks (PAN) based on the revised architecture of GAN. Besides performance, PAN has the advantage of not requiring class prior probability.

6.2.3 Domain Adaptation in PU Learning Setting

As in our study, there are several recent studies that address domain adaptation in the context of PU learning. For example, Sonntag et al. [142] considers the domain adaptation scenario where the source domain has labeled data of all classes, and the target domain has unlabeled data and a few positive labeled data in the homogeneous feature space. The setting differs from ours which is more difficult, in that the target domain has positive labeled data and the source domain has both positive and negative data, and in the homogeneity of the feature space. The scenario where both target and source domains have unlabeled and positive labeled data is also treated for a link prediction task in the homogeneous setting [143] and heterogeneous setting [144], respectively. Their settings are also different from ours where the source domain has only positive labeled data and the target domain has only unlabeled data. In addition, there are studies that address open set domain

adaptation (OSDA) by considering it as PU learning [145, 146]. OSDA performs domain adaptation while also rejects target classes that are not present in the source domain as unknown. In their settings, the source domain has a positive (known) labeled data and the target domain has unlabeled data including positive and negative (unknown) data. However, they consider the homogeneous feature space unlike our study. Thus far, no study has examined the PU learning task in the HUDA setting.

6.3 Positive and Unlabeled Heterogeneous Domain Adaptation

6.3.1 Problem Formulation

We consider a two-domain setting with a *source* domain s and a *target* domain t , following the notation in Chapter 2. The source and target domains exhibit feature-space heterogeneity with a partially overlapping space, as defined in Chapter 2. Accordingly, a sample from domain $* \in \{s, t\}$ is denoted by $x_* = (x_*^c, x_*^u) \in \mathcal{X}_* = \mathcal{X}^c \times \mathcal{X}_*^u$, where x_*^c represents the common features and x_*^u represents the domain-specific features.

We observe a positive-only labeled source dataset $D_s = \{(x_{s,i}, y_{s,i})\}_{i=1}^{N_s}$ with $y_{s,i} = 1$ for all i and a fully unlabeled target dataset $D_t = \{x_{t,i}\}_{i=1}^{N_t}$, which contains both positive and negative samples with unknown labels. This setting differs from conventional HUDA, which assumes both positive and negative labeled data in the source domain. Our goal is to predict binary labels for target data. These settings are visualized in Figure 6.1.

A simple baseline is to ignore domain-specific features and apply PU learning using only the common features $\{x_{s,i}^c\}$ and $\{x_{t,i}^c\}$. However, this cannot leverage $\{x_{s,i}^u\}$ and $\{x_{t,i}^u\}$, which often contain important discriminative information. Therefore, PU-HUDA requires a method that can explicitly utilize domain-specific features despite feature-space heterogeneity.

6.3.2 Difficulty of PU-HUDA

A naive approach is to directly combine an existing HUDA method with a PU learning method. One may first learn feature transformations that map source and target samples into a homogeneous space (e.g., DSFT, GLG, or SFER), and then

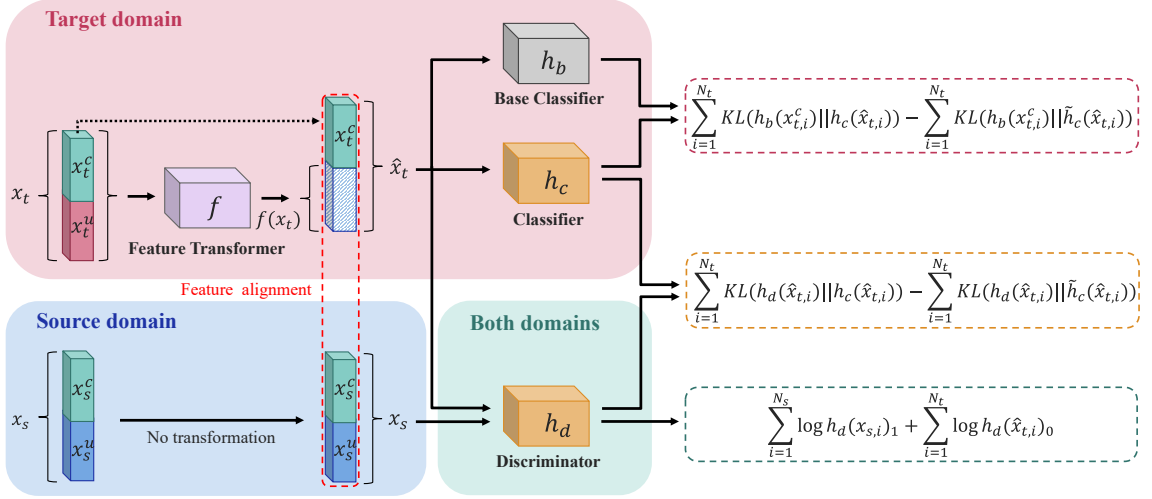


Figure 6.3: Overview of the proposed method, PADA using soft-labeling mechanism.

apply PU learning to transformed source positives and transformed unlabeled target samples.

This combination can be ineffective because HUDA methods typically learn transformations that reduce a global distribution discrepancy between transformed source and target representations (e.g., MMD). In PU-HUDA, however, the transformed source set contains only positives, while the transformed target set contains both positives and negatives. Consequently, reducing a global discrepancy between these sets does not necessarily yield a feature space that is discriminative for separating positive and negative target samples (Fig. 6.2, top). We empirically confirm in Section 6.5 that this naive combination fails, motivating a new approach.

6.4 Proposed Approach

To overcome the difficulty described in the previous section, we need to align only the positive portion of the target data with the source data, while keeping the negative target data separated from the source data in the transformed feature space.

6.4.1 Basic Idea

We view PU-HUDA as a task to identify likely positive examples from the target data and reduce the distribution divergence between the whole source examples and the likely positive target examples. To solve this task, we exploit adversarial training and thus introduce a discriminator. Hence, our proposed method mainly consists

of the following three models as described in Figure 6.3. Details of Figure 6.3 are explained in Sec. 6.4.3.

(1) Feature Transformer $f : \mathcal{X}_t \rightarrow \mathcal{X}_s^u$ transforms the target feature space to that of the source. Concretely, f transforms target sample x_t into the source domain-specific space, and we denote the transformed target sample by $\hat{x}_t = (x_t^c, f(x_t)) \in \mathcal{X}_s$. We adopt this asymmetric transformation [132] (i.e., transforming only the target) because learning separate mappings for source and target can be unstable under severe label imbalance and feature-space heterogeneity, often leading to poor local optima [132].

(2) Classifier $h_c : \mathcal{X}_s \rightarrow \mathbb{R}^2$ identifies positive examples in the transformed unlabeled target data (classifies positive and negative target examples). We represent the output confidence scores of h_c as $h_c(\cdot) = (h_c(\cdot)_0, h_c(\cdot)_1)$, where $h_c(\cdot)_1$ and $h_c(\cdot)_0$ represent the probabilities that the input samples are predicted to be positive and negative, respectively.

(3) Discriminator $h_d : \mathcal{X}_s \rightarrow \mathbb{R}^2$ determines whether the likely positive examples are from the target or source domain (negative examples are predicted as target). The same notation as h_c applies to h_d although positive and negative mean source and target for h_d , respectively.

Since h_d is trained to discriminate the source and target examples, f and h_d are trained as follows:

- f is trained so that the source data and the transformed target data predicted by h_c to be positive cannot be distinguished by h_d .
- h_c is trained to identify likely positive examples from the target data transformed by f so that they cannot be distinguished from the source data that are all positive by h_d . Therefore, h_c can classify positive and negative examples.

h_c and f provide feedback to each other. That is, if h_c can predict the correct positive examples, f can reduce the distribution divergence between the truly positive target data and the source data, and conversely if f can transform the target positive data into the data close to the source data, h_c can predict the likely positive examples more indistinguishable from the source data. Therefore, by alternately training them with h_d , we can obtain good h_c and f , which are used for binary classification inference on the target data.

We next consider the optimization ways to train h_c , h_d , and f .

6.4.2 Naive Optimization Approach

We begin with a naive idea: focus only on likely positive target samples and reduce the distribution discrepancy between these samples and the source data in the transformed space.

A standard objective function for domain adaptation methods using adversarial learning [19, 147, 148] to optimize h_d and f is the following cross-entropy loss:

$$\min_f \max_{h_d} \mathcal{L}(h_d, f) = \mathbb{E}_{x_s \sim \mathcal{P}_s(\mathcal{X}_s)} [\log h_d(x_s)_1] + \mathbb{E}_{x_t \sim \mathcal{P}_t(\mathcal{X}_t)} [\log h_d(\hat{x}_t)_0], \quad (6.1)$$

where $\hat{x}_t = (x_t^c, f(x_t))$ and $x_t = (x_t^c, x_t^u)$. Maximizing $\mathcal{L}$ trains h_d to recognize source and target domains, while minimizing $\mathcal{L}$ trains f to fool h_d , thereby reducing the divergence between the source and target data.

However, optimizing Eq. (6.1) directly suffers from the label-gap issue in Section 6.3.2. To focus on likely positive target samples and train h_c simultaneously, we incorporate $h_c(\hat{x}_t)_1$ as a weight:

$$\min_{h_c, f} \max_{h_d} \mathcal{L}_{\text{wada}}(h_d, f, h_c) = \mathbb{E}_{x_s \sim \mathcal{P}_s(\mathcal{X}_s)} [\log h_d(x_s)_1] + \frac{\mathbb{E}_{x_t \sim \mathcal{P}_t(\mathcal{X}_t)} [h_c(\hat{x}_t)_1 \log h_d(\hat{x}_t)_0]}{\mathbb{E}_{x_t \sim \mathcal{P}_t(\mathcal{X}_t)} [h_c(\hat{x}_t)_1]}. \quad (6.2)$$

The denominator of the second term prevents the trivial solution $h_c(\hat{x}_t)_1 = 0$ for all x_t . We call this naive optimization approach *weighted adversarial domain adaptation* (WADA). Although intuitive, WADA has the following two limitations:

- WADA does not penalize false negatives on h_c ; thus h_c may assign positive probability only to a small set of highly confident positives.
- Since the second term in Eq. (6.2) tends to be close to 0 for negative examples, f will overfit on positive examples, resulting in poor representations for separating negative target samples.

6.4.3 Predictive Adversarial Domain Adaptation

To address the problems in WADA, we propose *predictive adversarial domain adaptation* (PADA). PADA treats positive and negative examples equally by using an objective function based on KL divergence for confidence scores, inspired by PAN

[135]. The empirical objective function of PADA is as follows:

$$\begin{aligned} \min_{h_c, f} \max_{h_d} \widehat{\mathcal{L}}_{\text{pada}}(D_s, D_t; h_d, f, h_c) = & \sum_{i=1}^{N_s} \log h_d(x_{s,i})_1 + \sum_{i=1}^{N_t} \log h_d(\hat{x}_{t,i})_0 \\ & + \lambda \left(\sum_{i=1}^{N_t} KL(h_d(\hat{x}_{t,i}) || h_c(\hat{x}_{t,i})) - \sum_{i=1}^{N_t} KL(h_d(\hat{x}_{t,i}) || \tilde{h}_c(\hat{x}_{t,i})) \right), \end{aligned} \quad (6.3)$$

where $\hat{x}_{t,i} = (x_{t,i}^c, f(x_{t,i}))$, $x_{t,i} = (x_{t,i}^c, x_{t,i}^u)$ and $\tilde{h}_c(\cdot)$ denotes the opposite output of $h_c(\cdot)$, i.e., $\tilde{h}_c(\cdot) = (h_c(\cdot)_1, h_c(\cdot)_0)$.

Cross-entropy terms. Optimizing the first two terms (empirical version of Eq. (6.2)) has the following effects on h_d and f :

- h_d can distinguish between the source and target data to some extent. All target data are treated as negative for h_d , enabling h_d to recognize target data.
- Since f is trained adversarially against h_d , f can transform the whole target data into the data close to the source data to some extent.

KL-divergence terms. These terms mitigate the label-gap problem arising from the use of only the first two terms and address the limitations of WADA. Since the fourth term only symmetrizes the gradient by the third term and improves performance [135], we focus on the effects of the third term:

- h_c is trained to output confidence scores similar to those of h_d for transformed target data. Specifically, h_c assigns a high positivity score to target examples that h_d finds difficult to distinguish from source examples (i.e., likely positive target examples), and a low positivity score to those that are easily distinguishable (i.e., likely negative target examples). As a result, h_c acquires the ability to discriminate positive and negative target examples.
- h_d aims to discriminate between source examples and positive target examples. h_d is trained to output confidence scores opposite to those of h_c . If h_c identifies a likely positive target example, h_d attempts to classify it as target (negative). Conversely, if h_c identifies a likely negative target example, h_d attempts to classify it as source (positive). Although the latter effect is undesirable, it is counteracted by the effect of the second term.

Algorithm 6: Learning of PADA by stochastic gradient descent

Input: Positive labeled source data $D_s = \{(x_{s,i}, y_{s,i} = 1)\}_{i=1}^{N_s}$, unlabeled target data $D_t = \{x_{t,i}\}_{i=1}^{N_t}$, number of training steps R , initial parameters $\theta_c, \theta_d, \theta_f$ of classifier h_c , domain discriminator h_d , and feature extractor f

Output: Trained models h_c, h_d, f

```
1 for  $r = 1$  to  $R$  do
2   Sample mini-batch  $\mathbf{x}_s$  from  $D_s$  and  $\mathbf{x}_t$  from  $D_t$            // training  $h_d$ 
3    $\theta_d \leftarrow \theta_d + \nabla_{\theta_d} \hat{\mathcal{L}}_{\text{pada}}(\mathbf{x}_s, \mathbf{x}_t; h_d, f, h_c)$ 
4   Sample mini-batch  $\mathbf{x}_s$  from  $D_s$  and  $\mathbf{x}_t$  from  $D_t$            // training  $f$ 
5    $\theta_f \leftarrow \theta_f - \nabla_{\theta_f} \hat{\mathcal{L}}_{\text{pada}}(\mathbf{x}_s, \mathbf{x}_t; h_d, f, h_c)$ 
6   Sample mini-batch  $\mathbf{x}_t$  from  $D_t$                                // training  $h_c$ 
7    $\theta_c \leftarrow \theta_c - \nabla_{\theta_c} \hat{\mathcal{L}}_{\text{pada}}(\cdot, \mathbf{x}_t; h_d, f, h_c)$ 
8 return  $h_c, h_d, f$ 
```

- f is trained such that h_d produces confidence scores similar to those of h_c . That is, target examples predicted as likely positive (resp. negative) by h_c become indistinguishable (resp. distinguishable) from source examples by h_d . Consequently, f maps positive target data closer to the source data while keeping negative target data separated, an effect that cannot be achieved by WADA.

As a result, PADA yields a label-discriminative feature space in the target domain and a classifier that performs well on this space. Algorithm 6 summarizes the learning procedure of PADA using stochastic gradient descent.

Finally, we discuss the role of common features in PADA. Common features prevent PADA from collapsing into a trivial solution in which h_c predicts all target examples as negative. They help identify target samples that are difficult for h_d to distinguish from source samples, particularly in the early stages of training. In contrast, PADA cannot collapse into the opposite trivial solution, where h_c predicts all target samples as positive, because h_d is explicitly trained to classify target data as negative by the second term in Eq. (6.3). See the next subsection for a more explicit use of common features to facilitate the learning of h_c .

6.4.4 Soft labeling mechanism to improve PADA

In the early stage of learning PADA, f is not yet reliable, and the features $f(x_t)$ can be noisy, making it difficult for h_c to identify likely positives, which leads to

the unstable learning of h_c and f . To mitigate this issue, we guide h_c using a *base classifier* trained only on common features, which can be obtained stably without relying on f .

Specifically, we first apply a standard PU learning method to the common features of the source and target data. Let h_b denote the resulting base classifier, which takes x_t^c as input and outputs soft class probabilities. If common features are informative, h_b can achieve non-trivial accuracy and thus provide useful guidance for training h_c .

We incorporate this guidance by augmenting the PADA objective with a distillation-style soft-labeling term:

$$\begin{aligned} \min_{h_c, f} \max_{h_d} \widehat{\mathcal{L}}'_{\text{pada}}(D_s, D_t; h_d, f, h_c) &= \widehat{\mathcal{L}}_{\text{pada}}(D_s, D_t; h_d, f, h_c) \\ &+ \eta \left(\sum_{i=1}^{N_t} KL(h_b(x_{t,i}^c) || h_c(\hat{x}_{t,i})) - \sum_{i=1}^{N_t} KL(h_b(x_{t,i}^c) || \tilde{h}_c(\hat{x}_{t,i})) \right), \end{aligned} \quad (6.4)$$

where η is a balancing parameter. The added KL terms encourage h_c to follow the *soft labels*¹ of h_b as teacher labels when h_d and f are not yet learned enough, thereby stabilizing and accelerating training of h_d and f in early iterations.

Once training converges, h_c is expected to perform better than h_b . Therefore, we can repeat the soft-labeling procedure by updating the base classifier with the newly learned model h_c . In the first round, h_b uses only common features; in later rounds, the updated base classifier can also leverage the transformed representation and thus provide stronger guidance until performance saturates.

6.5 Experiments

6.5.1 Experimental Setup

6.5.1.1 Datasets

We reorganized three datasets and used them to evaluate our proposed methods since no public dataset was directly related to the PU-HUDA setting. Detailed information about the datasets is presented in Table 6.2.

¹Soft labels are usually used in the context of *distillation* [149], which transfers the knowledge of large models to small models.

Table 6.2: Detailed information about the processed datasets, Movielens-Netflix, 20-Newsgroups, and Default of credit card clients.

Dataset		Number of samples				Number of features			
		Training			Testing		Common	Domain-specific	
		Target		Source	Pos	Neg		Target	Source
		Pos	Neg	Pos					
Movielens-Netflix		6,836	9,164	8,835	55,725	74,816	4	13	20
20-Newsgroups	CR	1,168	1,197	1,768	397	390	500	5,000	5,000
	CS	1,168	1,187	1,768	378	406			
	CT	1,168	842	1,768	391	278			
	RS	1,192	1,187	1,197	396	396			
	RT	1,192	842	1,197	412	266			
	ST	1,187	842	1,186	380	296			
Credit card	0.5	3,763	3,763	2,873	753	753	4	13	13
	0.3	3,763	8,780	2,873	732	1,778			
	0.1	1,594	14,349	2,873	325	2,865			

Movielens-Netflix [150, 151]: Movielens and Netflix are two movie datasets that contain users and their ratings for movies on a scale of [0,5]. To create a classification dataset using them, we focus on the movie genre rather than the movie title. We first calculate each user’s average rating for each genre, and the differences between those and the user’s average rating for all movies are used as features. The genres common to Movielens and Netflix correspond to common features. We select one feature from common features and label each user as positive or negative according to whether the value for the feature is positive or negative, i.e., whether the user likes the corresponding genre or not. We chose Adventure as the label because the correlation of Adventure with other features is high, and the ratios of positive and negative are approximately equal. Since the default proportion of common genres is large, we randomly chose 4 genres from the 16 common genres and used them as common features. Half of the remaining common genres were used for Movielens-specific features and half for Netflix-specific features. We assigned Movielens as the target and Netflix as the source.

20-Newsgroups [152]: a collection of about 20,000 documents belonging to four top categories: computer (C), recording (R), science (S), and talk (T), each containing four more subcategories. We create a binary classification task as in DSFT [66] by selecting two top categories and assigning them positive and negative labels. Half of the subcategories are used for source and half for target. Each document is represented by tf-idf features. We use the top 500 most frequent common words in the source and target domain as common features and the top 5000 frequent specific words in each domain as domain-specific features.

Default of credit card clients [153]: a collection of credit card records, including user demographics, history of payments, bill statements, and previous payments, with users’ default payments as binary labels. We assign male users as source and female users as target. User demographics are used as common features, which may often occur in reality. We use bill statements and half of the history of payments as source-specific features, and previous payments and half of the history of payments as target-specific features. We vary the positive ratio of the target data from 0.5 to 0.1 to see the sensitivity of PADA to it.

6.5.1.2 Compared methods

We compare the performances of **PADA** and PADA with soft-labeling (**PADA_s**) to the three types of baseline methods in addition to **WADA**.

The first baseline is to apply existing PU learning methods to only common features. As existing PU learning methods, we use **NaivePU**, which treats the entire target data as negative, **nnPU** [141], and **PAN** [135].

The second type of baseline is **DIST**, which uses not only common features but also target-specific features². **DIST** only uses the soft labeling mechanism of **PADA_s**, i.e., first gives soft labels to the unlabeled target data using the model trained by the first baseline and trains a classifier using all the target features and the soft labels as in usual distillation [149]. As the first baseline, we adopt **PAN** because of its high performance.

The last baseline is a direct combination of existing **HUDA** and PU learning methods described in Sec. 6.3.2. This baseline first constructs a homogeneous feature space by existing **HUDA** methods and next applies PU learning methods to the source and target data on the constructed homogeneous feature space. As a PU learning method, we adopt **PAN** for the same reason as **DIST**. We use two **HUDA** methods as baselines, **DSFT** [66] and **SFER** [70]. The linear and non-linear **DSFT** are denoted by **DSFT_l** and **DSFT_{nl}**, respectively. **SFER** is not used for 20-Newsgroups because **SFER** can only be applied to low dimensional features due to computational cost.

²Note that **DIST** can not transfer the knowledge of the source-specific features to the target domain.

Table 6.3: Performance comparison using accuracy (%) for Movielens-Netflix dataset. Four common features were randomly selected five times, which makes 5 experimental settings.

Settings	Methods									
	NaivePU	nnPU	PAN	DIST	DSFT _{<i>l</i>}	DSFT _{<i>nl</i>}	SFER	WADA	PADA	PADA _{<i>s</i>}
A	59.01	57.89	59.93	62.64	62.34	57.33	57.84	64.40	66.29	66.13
B	58.40	57.55	59.99	62.50	57.09	58.55	60.53	61.49	61.90	63.14
C	63.72	63.10	62.64	65.07	68.35	60.94	57.82	67.90	67.82	68.93
D	62.93	63.26	65.49	66.85	69.17	58.71	62.99	65.52	70.26	69.70
E	62.65	63.10	66.61	66.59	66.34	61.54	57.70	64.60	65.44	66.72
Average	61.34	60.98	62.93	64.73	64.66	59.42	59.38	64.78	66.34	66.92

Table 6.4: Performance comparison using accuracy (%) for 20-Newsgroups dataset. We selected two top categories and assigned them as positive and negative labels.

Labels		Methods									
		NaivePU	nnPU	PAN	DIST	DSFT _{<i>l</i>}	DSFT _{<i>nl</i>}	SFER	WADA	PADA	PADA _{<i>s</i>}
C	R	63.79	73.10	80.14	84.88	84.16	83.02	-	78.37	80.43	86.11
C	S	57.82	59.91	64.71	64.75	66.07	69.81	-	57.63	65.48	66.54
C	T	57.85	73.84	80.67	79.42	80.17	80.47	-	77.18	80.67	83.76
R	S	51.43	50.80	50.93	50.13	55.72	51.35	-	53.75	55.81	56.73
R	T	57.18	58.21	66.37	66.52	65.44	67.01	-	60.28	66.96	67.21
S	T	52.76	54.19	66.47	67.06	65.78	66.17	-	56.14	66.91	71.40
Average		56.80	61.68	68.21	68.79	69.57	69.64	-	63.89	69.38	71.96

6.5.1.3 Implementation details

The binary classifier trained by each method was tested on the target domain. We split each target data into training, validation, and testing datasets. Each method was tuned by the validation dataset and tested by the testing dataset. We implemented each method three times and used average test accuracy as the test metric. For PADA and WADA, we selected the best learning rate in [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1], λ and η in [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1]. We repeated the soft-labeling of PADA until the convergence of validation accuracy. The batch size of PADA was set to 128. For a fair comparison, each method adopted a linear classifier and a linear discriminator, i.e., logistic regressions. We also used a linear transformation for a feature transformer.

6.5.2 Results and Analysis

6.5.2.1 Performance comparison

The test accuracy results of all compared methods are reported in Table 6.3, Table 6.4, and Table 6.5. We also report AUC for Default of credit card clients because the positive ratio is changed to small. The best result among compared methods is

Table 6.5: Performance comparison using accuracy (%) and AUC (%) for Default of credit card clients dataset. The positive ratio of target data was varied from 0.5 to 0.1.

Positive ratio	Test metric	Methods									
		NaivePU	nnPU	PAN	DIST	DSFT _{<i>l</i>}	DSFT _{<i>nl</i>}	SFER	WADA	PADA	PADA _{<i>S</i>}
0.5	Accuracy	54.58	55.47	54.60	57.85	44.42	54.07	49.40	59.88	60.09	63.70
	AUC	58.03	58.21	58.76	61.72	37.75	55.93	48.94	65.25	63.02	68.42
0.3	Accuracy	61.29	69.63	62.79	66.02	58.91	65.42	70.57	69.63	65.31	68.53
	AUC	58.76	60.09	61.68	66.04	49.33	45.88	58.73	67.79	65.55	68.09
0.1	Accuracy	64.03	83.78	63.32	67.64	62.87	82.59	80.61	69.20	70.73	72.20
	AUC	59.46	58.24	61.24	66.74	70.52	68.51	36.85	63.03	65.77	61.93

written in bold.

MovieLens-Netflix: Table 6.3 shows the results for MovieLens-Netflix and we can see that PADA and PADA_{*S*} outperform all the baseline methods in every setting. Specifically, they improve the first baselines such as PAN using only common features by about 4% on average and by about 6% in a particular setting, which is an effect of source-specific and target-specific features. Note that DIST also improves PAN. This is because the generalization performance is improved by using not only common features but also target-specific features when performing distillation by the soft labels given by PAN. Moreover, PADA outperforms DIST, which indicates that they successfully transfers the source-specific features to the target domain. Compared to PADA, WADA’s performance is modest, as expected. We can also observe the positive effect of the soft-labeling by PADA_{*S*} over PADA. We also note that the naive HUDA and PU learning combination methods, such as DSFT_{*l*}, DSFT_{*nl*}, and SFER, provide limited performances. Unlike DSFT, which does nothing for common features, SFER, which transforms all features, does not work at all. This fact is consistent with the difficulty of these methods described in Sec. 6.3.1. See Sec. 6.5.3 for verification of the effectiveness of PADA by observing the distribution divergence between the positive (or negative) target and source data in the transformed feature space.

20-Newsgroups: The results for 20-Newsgroups are presented in Table 6.4. In this dataset, we can observe the positive effect of DSFT. In the setting C-S, for example, DSFT_{*nl*} improves PAN by 5%. Overall, however, its improvement over PAN is modest. Although the improvement of PADA over PAN is also modest, the soft-labeling of PADA_{*S*} is considerably effective in this dataset, outperforming the baseline methods in almost all settings. We note that at most three iterations of soft labeling in PADA_{*S*} were sufficient. We finally mention that WADA does not work well at all in this dataset.

Default of credit card clients: Table 6.5 shows the results for Default of credit

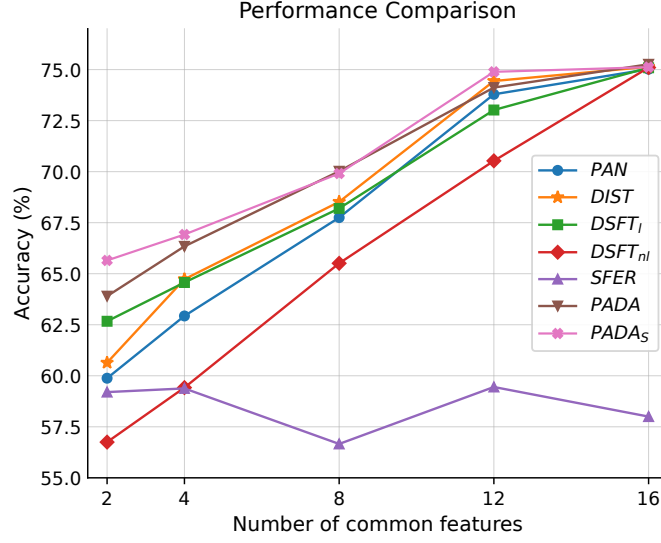


Figure 6.4: Effect of the number of selected common features for Movielens-Netflix

card clients. When the positive ratio is 0.5, we can see that $PADA_S$ is the best method in terms of both accuracy and AUC. When the positive ratio is 0.3, WADA outperforms PADA although $PADA_S$ also shows almost equivalent performance to WADA. Note that the accuracy of SFER is high, its AUC is low, which means it predicts most of the data as negative. In the case of 0.1 positive ratio, PADA and $PADA_S$ also show the improvements over most of the baselines. However, $DSFT_{nl}$ is better than them. As a result, DSFT may be effective in particular datasets and settings, but its effective range is very limited.

6.5.2.2 Analysis on the number of common features

Using the Movielens-Netflix dataset, we further investigate the effect of the number of common features on our proposed methods and several baseline methods, including PAN, DIST, DSFT, and SFER. Figure 6.4 shows the test accuracy when varying the number of selected common features from 2 to 16 (all genres common to Movielens and Netflix). We can see that the performance of each method degrades as the number of common features decreases, but $PADA_S$ shows the least degradation. Conversely, the greater the number of common features, the smaller the performance difference between all methods except SFER. This is because as the number of common features increases, the number of target-specific features decreases, i.e., the target-specific features lose their usefulness. SFER, a naive combination of HUDA and PU learning is not valid regardless of the number of common features, which means that the homogeneous feature spaces obtained by it lose or

inhibit the usefulness of common features.

6.5.3 Adaptation analysis of PADA

In this section, we experimentally confirm that PADA transforms the positive target data into data close to the source data while keeping the negative target data away from the source data using the Movielens-Netflix dataset.

We compare PADA with three baselines: (1) Common, which uses only the raw common features of the source and target data, (2) DSFT, a naive combination of HUDA and PU learning methods, and (3) WADA. To see the distribution distance between the source data and the positive (resp. negative) target data in the resulting feature space, we define a metric $Acc_d(s, t_p)$ (resp. $Acc_d(s, t_n)$), the test accuracy of discrimination by a *test discriminator* that discriminates the source data and the positive (resp. negative) target data. The test discriminator is trained on the resulting feature space after finishing the learning of each method or only on the common feature space for Common. Note that the test discriminator is different from the discriminator trained for PADA and WADA. The true labels of the target data are used to train the test discriminator. If $Acc_d(s, t_p)$ (resp. $Acc_d(s, t_n)$) is close to 0.5, the distribution divergence between the source data and the positive (resp. negative) target data is considered small. Therefore, small $Acc_d(s, t_p)$ and large $Acc_d(s, t_n)$ are desirable in PU-HUDA.

The results are presented in Table 6.6. As expected, we can see that $Acc_d(s, t_p)$ of PADA is smaller than those of the other methods, while $Acc_d(s, t_n)$ of PADA is larger than it. This means that the positive target data are transformed into data close to the source data while keeping the distance between the negative target data and the source data. $Acc_d(s, t_p)$ of Common is smaller than that of PADA, but $Acc_d(s, t_n)$ of Common is also small. As a result, the accuracy of the existing PU learning methods using only common features is low. Even for DSFT, the difference between $Acc_d(s, t_p)$ and $Acc_d(s, t_n)$ is small, which is consistent with the problem of the naive combination approaches described in Sec. 6.3.2. We also note that although the difference between $Acc_d(s, t_p)$ and $Acc_d(s, t_n)$ of WADA is large, its $Acc_d(s, t_p)$ is also large. Thus, if WADA is trained to make $Acc_d(s, t_p)$ smaller, $Acc_d(s, t_n)$ is expected to be even closer to $Acc_d(s, t_p)$, which is an undesirable result as described in Sec. 6.4.2.

Table 6.6: The average test accuracy (%) of the test discriminator for each method using Mevielens-Netflix.

Method	Accuracy of method	Accuracy of discrimination	
		$Acc_d(s, t_p)$	$Acc_d(s, t_n)$
Common	62.93	54.79	55.79
DSFT	64.73	72.59	76.58
WADA	64.78	70.73	77.12
PADA	66.34	61.08	66.02

6.6 Summary

We defined a new heterogeneous domain adaptation task where only positive examples are available in the source domain. To overcome the gap in label distribution between the source and target data, we proposed a novel method that integrates the feature alignment and PU learning in a unified adversarial training framework. Moreover, we experimentally showed that our proposed method outperformed several baseline methods.

Finally, it should be mentioned that this chapter does not take into account the divergence between the distributions of the source common features and target common features. It is a future work to reduce the divergence between the source common features and target common features (if there is any divergence) and increase the performance of our proposed method.

Chapter 7

Conclusion and Future Direction

7.1 Conclusion

This thesis provided a systematic investigation of data heterogeneity in collaborative learning, a setting in which multiple parties jointly train machine learning models through either direct data aggregation or privacy-preserving distributed training. While collaborative learning aims to exploit the collective knowledge of multiple data holders, its effectiveness is fundamentally limited by heterogeneity in the underlying data distributions and observable information across parties. The research presented in this thesis addressed these challenges by designing practical algorithms that enable robust collaboration under realistic and diverse constraints.

We categorized heterogeneity into two primary types, distribution heterogeneity and observation heterogeneity, and explored their implications across various collaborative learning frameworks. Through four complementary studies, this thesis demonstrated how these forms of heterogeneity arise in practice and how algorithmic design can mitigate their negative impact. Specifically, we developed a personalized federated learning method for label-shift scenarios through a multi-branch architecture (Chapter 3), a federated learning approach that integrates common and party-specific networks to handle feature-space heterogeneity (Chapter 4), a federated adaptation framework combining pseudo-label refinement and privacy-preserving domain-aware aggregation for unlabeled heterogeneous domains (Chapter 5), and a domain adaptation method that integrates heterogeneous feature spaces and positive-unlabeled learning in a centrally aggregated yet highly heterogeneous environment (Chapter 6).

An important implication of the methods in Chapters 3 to 5 is that collaboration

among parties with similar data distributions can be interpreted as approximately performing FedAvg within subsets of such parties. As a result, the scaling behavior with respect to total data size, discussed in Section 2.2, can still hold approximately within these subsets. From this perspective, increasing the number of participating parties with similar distributions improves performance and alleviates data scarcity at individual parties. On the other hand, the method proposed in Chapter 6 addresses a substantially more severe and composite form of heterogeneity, enabling effective model learning in scenarios where training a reliable model would be infeasible for a single party in isolation.

Collectively, these contributions form a cohesive understanding of heterogeneous collaborative learning and demonstrate that effective collaboration is achievable even when parties differ in data distributions, feature observability, and label availability. By advancing algorithms tailored to practical constraints, this thesis opens new opportunities for data collaboration across organizations, industries, and domains where heterogeneity has traditionally been a barrier to joint learning.

7.2 Future Direction

While this thesis provides a systematic examination of heterogeneous collaborative learning and proposes practical algorithms for diverse scenarios, several promising directions remain open for future research.

1. Theoretical analysis of heterogeneous collaborative learning.

A deeper theoretical understanding of how heterogeneity affects learning performance is still lacking. In particular, the relationship between the degree of heterogeneity and generalization error remains unclear under observation heterogeneity, where parties differ in feature spaces or label availability. In federated learning, heterogeneity also impacts optimization behavior, making convergence analysis under realistic settings an important open problem. Establishing such theoretical foundations would clarify when collaboration is beneficial and guide the design of more principled algorithms.

2. Privacy guarantees under data heterogeneity.

Practical collaborative learning often requires strong privacy guarantees. Since the methods proposed in this thesis follow the standard FL workflow, they can in principle be combined with existing privacy-preserving techniques, such as

differential privacy and cryptographic methods, to further mitigate information leakage through model parameters. However, the optimal integration of such privacy mechanisms under data heterogeneity remains largely unexplored. For example, the interaction between the amount of noise required for differential privacy, data heterogeneity, and learning performance is unclear. Ensuring satisfactory performance under heterogeneity while satisfying privacy constraints is critical for real-world adoption.

3. Robustness to dynamic and evolving distributions.

Real-world collaborative environments often experience temporal drift in data distributions, with parties joining, leaving, or updating their data collection pipelines. Extending collaborative learning methods to adapt to dynamically changing heterogeneity, while maintaining stability and efficiency, remains an essential direction for practical deployment.

4. Practical deployment and cross-industry applications.

Building on the realistic scenarios studied in this thesis, further efforts may examine full end-to-end deployment pipelines, including automated data preprocessing, communication-efficient protocols, model selection under heterogeneity, and real-world evaluation. Such work may accelerate practical adoption in domains such as finance, healthcare, and manufacturing, where heterogeneous collaboration is both valuable and challenging.

Pursuing these directions will deepen both the theoretical and practical foundations of heterogeneous collaborative learning, ultimately broadening the range of applications where multi-party collaboration becomes feasible and beneficial.

Bibliography

- [1] Donald Shenaj, Eros Fani, Marco Toldo, Debora Caldarola, Antonio Tavera, Umberto Michieli, Marco Ciccone, Pietro Zanuttigh, and Barbara Caputo. Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 444–454, 2023.
- [2] Kuniaki Saito, Donghyun Kim, Piotr Teterwak, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9164–9173, 2021.
- [3] Benyuan Sun, Hongxing Huo, YI YANG, and Bo Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23309–23320. Curran Associates, Inc., 2021.
- [4] Yue Tan, Guodong Long, Jie Ma, LU LIU, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 19332–19344. Curran Associates, Inc., 2022.
- [5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.
- [6] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on

- non-iid data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7865–7873, 2021.
- [7] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 19586–19597. Curran Associates, Inc., 2020.
 - [8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.
 - [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
 - [10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
 - [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
 - [12] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.
 - [13] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.

- [14] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021.
- [15] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.
- [16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.
- [17] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically, 2017.
- [18] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [19] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. *Domain-Adversarial Training of Neural Networks*, pages 189–209. Springer International Publishing, Cham, 2017.
- [20] Hao Guan and Mingxia Liu. Domain adaptation for medical image analysis: A survey. *IEEE Transactions on Biomedical Engineering*, 69(3):1173–1185, 2022.
- [21] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23, 2017.
- [22] Bertrand Leblot, Yann-Aël Le Borgne, Liyun He-Guelton, Frédéric Oblé, and Gianluca Bontempi. Deep-learning domain adaptation techniques for credit cards fraud detection. In Luca Oneto, Nicolò Navarin, Alessandro Sperduti, and Davide Anguita, editors, *Recent Advances in Big Data and Deep Learning*, pages 78–88, Cham, 2020. Springer International Publishing.

- [23] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019.
- [24] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction, 2019.
- [25] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):12598, 2020.
- [26] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.
- [27] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In Keke Chen, Sangeetha Seshadri, and Liang-Jie Zhang, editors, *Big Data – BigData 2019*, pages 18–32, Cham, 2019. Springer International Publishing.
- [28] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.*, 8:985 – 1005, December 2007.
- [29] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3122–3130. PMLR, 10–15 Jul 2018.
- [30] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.
- [31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of*

- the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.
- [32] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
 - [33] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.
 - [34] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722, 2021.
 - [35] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.
 - [36] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3557–3568. Curran Associates, Inc., 2020.
 - [37] Canh T. Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21394–21405. Curran Associates, Inc., 2020.
 - [38] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.
 - [39] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *CoRR*, abs/1705.10467, 2018.

- [40] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3710–3722, 2021.
- [41] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2021.
- [42] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptol. ePrint Arch.*, 2016:892, 2016.
- [43] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.
- [44] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.
- [45] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’19, page 2232–2240, New York, NY, USA, 2019. Association for Computing Machinery.
- [46] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient collaborative learning framework for distributed features. *CoRR*, abs/1912.11187, 2019.
- [47] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, 2020.
- [48] Yuan Gao, Maoguo Gong, Yu Xie, A. K. Qin, Ke Pan, and Yew-Soon Ong. Multiparty dual learning. *IEEE Transactions on Cybernetics*, pages 1–14, 2022.

- [49] Dashan Gao, Yang Liu, Anbu Huang, Ce Ju, Han Yu, and Qiang Yang. Privacy-preserving heterogeneous federated transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2552–2559, 2019.
- [50] Dashan Gao, Ce Ju, Xiguang Wei, Yang Liu, Tianjian Chen, and Qiang Yang. Hhhfl: Hierarchical heterogeneous horizontal federated learning for electroencephalography. *arXiv preprint arXiv:1909.05784*, 2019.
- [51] Alain Rakotomamonjy, Maxime Vono, Hamlet Jesse Medina Ruiz, and Liva Ralaivola. Personalised federated learning on heterogeneous feature spaces. *arXiv preprint arXiv:2301.11447*, 2023.
- [52] Enmao Diao, Jie Ding, and Vahid Tarokh. Semifl: Semi-supervised federated learning for unlabeled clients with alternate training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17871–17884. Curran Associates, Inc., 2022.
- [53] Ming Li, Qingli Li, and Yan Wang. Class balanced adaptive pseudo labeling for federated semi-supervised learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16292–16301, 2023.
- [54] Zhengming Zhang, Yaoqing Yang, Zhewei Yao, Yujun Yan, Joseph E. Gonzalez, Kannan Ramchandran, and Michael W. Mahoney. Improving semi-supervised federated learning by reducing the gradient diversity of models. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1214–1225, 2021.
- [55] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Federated semi-supervised learning with inter-client consistency & disjoint learning. In *International Conference on Learning Representations*, 2021.
- [56] Chenyou Fan, Junjie Hu, and Jianwei Huang. Private semi-supervised federated learning. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2009–2015. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.
- [57] Xiaoxiao Liang, Yiqun Lin, Huazhu Fu, Lei Zhu, and Xiaomeng Li. Rscfed: Random sampling consensus federated semi-supervised learning. In *2022*

- IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10144–10153, 2022.
- [58] Huisi Wu, Baiming Zhang, Cheng Chen, and Jing Qin. Federated semi-supervised medical image segmentation via prototype-based pseudo-labeling and contrastive learning. *IEEE Transactions on Medical Imaging*, 43(2):649–661, 2024.
 - [59] Mykola Servetnyk, Carrson C. Fung, and Zhu Han. Unsupervised federated learning for unbalanced data. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.
 - [60] Jichan Chung, Kangwook Lee, and Kannan Ramchandran. Federated unsupervised clustering with generative models. In *AAAI 2022 International Workshop on Trustable, Verifiable and Auditable Federated Learning*, 2022.
 - [61] Fengda Zhang, Kun Kuang, Long Chen, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Fei Wu, Yueting Zhuang, et al. Federated unsupervised representation learning. *Frontiers of Information Technology & Electronic Engineering*, 24(8):1181–1193, 2023.
 - [62] Ekdeep Lubana, Chi Ian Tang, Fahim Kawsar, Robert Dick, and Akhil Mathur. Orchestra: Unsupervised federated learning via globally consistent clustering. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14461–14484. PMLR, 17–23 Jul 2022.
 - [63] Hansol Kim, Youngjun Kwak, Minyoung Jung, Jinho Shin, Youngsung Kim, and Changick Kim. Protofl: Unsupervised federated learning via prototypical distillation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6447–6456, 2023.
 - [64] Yi-Ren Yeh, Chun-Hao Huang, and Yu-Chiang Frank Wang. Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *IEEE Transactions on Image Processing*, 23(5):2009–2018, 2014.
 - [65] Joey Zhou, Sinno Pan, Ivor Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), 2014.

- [66] Pengfei Wei, Yiping Ke, and Chi Keong Goh. A general domain specific feature transfer framework for hybrid domain adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1440–1451, 2019.
- [67] Peter Prettenhofer and Benno Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127, Uppsala, Sweden, 2010. Association for Computational Linguistics.
- [68] Kyle D. Feuz and Diane J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Trans. Intell. Syst. Technol.*, 6(1), 2015.
- [69] Feng Liu, Guangquan Zhang, and Jie Lu. Heterogeneous domain adaptation: An unsupervised approach. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12):5588–5602, 2020.
- [70] Feng Liu, Jie Lu, and Guangquan Zhang. Unsupervised heterogeneous domain adaptation via shared fuzzy equivalence relations. *IEEE Transactions on Fuzzy Systems*, 26(6):3555–3568, 2018.
- [71] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109:719–760, 2020.
- [72] Zhenyu Sun, Xiaochun Niu, and Ermin Wei. Understanding generalization of federated learning via stability: Heterogeneity matters. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 676–684. PMLR, 02–04 May 2024.
- [73] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.
- [74] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020.

- [75] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [76] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.
- [77] Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. In *ICLR*, 2020.
- [78] Jialuo Cui, Qiong Wu, Zhi Zhou, and Xu Chen. Fedbranch: Heterogeneous federated learning via multi-branch neural network. In *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1101–1106, 2022.
- [79] Ching-Hao Wang, Kang-Yang Huang, Jun-Cheng Chen, Hong-Han Shuai, and Wen-Huang Cheng. Heterogeneous federated learning through multi-branch network. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2021.
- [80] Zhengyi Zhong, Ji Wang, Weidong Bao, Jingxuan Zhou, Xiaomin Zhu, and Zhang Xiongtao. Semi-hfl: semi-supervised federated learning for heterogeneous devices. *Complex & Intelligent Systems*, pages 1–23, 11 2022.
- [81] Qizhao Wang, Qing Li, Kai Wang, Hong Wang, and Peng Zeng. Efficient federated learning for fault diagnosis in industrial cloud-edge computing. *Computing*, 103(10):2319 – 2337, oct 2021.
- [82] Chen Zhu, Zheng Xu, Mingqing Chen, Jakub Konečný, Andrew Hard, and Tom Goldstein. Diurnal or nocturnal? federated learning of multi-branch networks from periodically shifting distributions. In *International Conference on Learning Representations*, 2022.
- [83] Manoj Ghuhana Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

- [84] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.
- [85] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *International Conference on Learning Representations*, 2020.
- [86] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 2019.
- [87] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9489–9502. PMLR, 18–24 Jul 2021.
- [88] Jiankai Sun, Xin Yang, Yuanshun Yao, Aonan Zhang, Weihao Gao, Junyuan Xie, and Chong Wang. Vertical federated learning without revealing intersection membership. *CoRR*, abs/2106.05508, 2021.
- [89] Andrew C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, 1982.
- [90] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.
- [91] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.
- [92] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

- [93] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12073–12086. PMLR, 18–24 Jul 2021.
- [94] Anastasiia Usmanova, François Portet, Philippe Lalanda, and Germán Vega. A distillation-based approach integrating continual learning and federated learning for pervasive services. *CoRR*, abs/2109.04197, 2021.
- [95] Sean M. Hendryx, Dharma Raj KC, Bradley Walls, and Clayton T. Morrison. Federated reconnaissance: Efficient, distributed, class-incremental learning. *CoRR*, abs/2109.00150, 2021.
- [96] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1736–1740, 2020.
- [97] Gihun Lee, Yongjin Shin, Minchan Jeong, and Se-Young Yun. Preservation of the global knowledge by not-true self knowledge distillation in federated learning. *CoRR*, abs/2106.03097, 2021.
- [98] Jock A. Blackard and Denis J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999.
- [99] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [100] J.M. Geusebroek, G.J. Burghouts, and A.W. Smeulder. The amsterdam library of object images. *International Journal of Computer Vision*, 61:103–112, 2005.
- [101] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, Alexander Statnikov, Wei-Wei Tu, and Evelyne Viegas. *Analysis of the AutoML Challenge Series 2015–2018*, pages 177–219. Springer International Publishing, Cham, 2019.

- [102] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *International Conference on Learning Representations*, 2020.
- [103] Junyuan Hong, Zhuangdi Zhu, Shuyang Yu, Zhangyang Wang, Hiroko H. Dodge, and Jiayu Zhou. Federated adversarial debiasing for fair and transferable representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 617–627, New York, NY, USA, 2021. Association for Computing Machinery.
- [104] Hua Kang, Zhiyang Li, and Qian Zhang. Communicational and computational efficient federated domain adaptation. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):3678–3689, 2022.
- [105] Yuwei Sun, Ng Chong, and Hideya Ochiai. Feature distribution matching for federated domain generalization. In Emtiyaz Khan and Mehmet Gonen, editors, *Proceedings of The 14th Asian Conference on Machine Learning*, volume 189 of *Proceedings of Machine Learning Research*, pages 942–957. PMLR, 12–14 Dec 2023.
- [106] Haozhe Feng, Zhaoyang You, Minghao Chen, Tianye Zhang, Minfeng Zhu, Fei Wu, Chao Wu, and Wei Chen. Kd3a: Unsupervised multi-source decentralized domain adaptation via knowledge distillation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3274–3283. PMLR, 18–24 Jul 2021.
- [107] Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. Federated multi-target domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1424–1433, January 2022.
- [108] Shaoduo Gan, Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D. Lane. Fruda: Framework for distributed adversarial domain adaptation. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):3153–3164, 2022.
- [109] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation.

- In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6028–6039. PMLR, 13–18 Jul 2020.
- [110] Fan Wang, Zhongyi Han, Yongshun Gong, and Yilong Yin. Exploring domain-invariant parameters for source free domain adaptation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7141–7150, 2022.
 - [111] Ning Ding, Yixing Xu, Yehui Tang, Chao Xu, Yunhe Wang, and Dacheng Tao. Source-free domain adaptation via distribution estimation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7202–7212, 2022.
 - [112] Sanqing Qu, Guang Chen, Jing Zhang, Zhijun Li, Wei He, and Dacheng Tao. BMD: A general class-balanced multicentric dynamic prototype strategy for source-free domain adaptation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 165–182, Cham, 2022. Springer Nature Switzerland.
 - [113] Jonghyun Lee, Dahuin Jung, Junho Yim, and Sungroh Yoon. Confidence score for source-free unsupervised domain adaptation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12365–12377. PMLR, 17–23 Jul 2022.
 - [114] Mattia Litrico, Alessio Del Bue, and Pietro Morerio. Guiding pseudo-labels with uncertainty estimation for source-free unsupervised domain adaptation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7640–7650, 2023.
 - [115] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8926–8935, 2019.
 - [116] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2024.

- [117] Shiqi Yang, yaxing wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29393–29405. Curran Associates, Inc., 2021.
- [118] Shiqi Yang, Yaxing Wang, Kai Wang, SHANGLING JUI, and Joost van de weijer. Attracting and dispersing: A simple approach for source-free domain adaptation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [119] Jiayi Tian, Jing Zhang, Wen Li, and Dong Xu. Vdm-da: Virtual domain modeling for source data-free domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(6):3749–3760, 2022.
- [120] Zhen Qiu, Yifan Zhang, Hongbin Lin, Shuaicheng Niu, Yanxia Liu, Qing Du, and Mingkui Tan. Source-free domain adaptation via avatar prototype generation and adaptation. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2921–2927. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.
- [121] Yixin Zhang, Zilei Wang, and Weinan He. Class relationship embedded learning for source-free unsupervised domain adaptation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7619–7629, 2023.
- [122] Ziyi Zhang, Weikai Chen, Hui Cheng, Zhen Li, Siyuan Li, Liang Lin, and Guanbin Li. Divide and contrast: Source-free domain adaptation via adaptive contrastive learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5137–5149. Curran Associates, Inc., 2022.
- [123] Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative clustering by regularized information maximization. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1, NIPS’10*, page 775 – 783, Red Hook, NY, USA, 2010. Curran Associates Inc.

- [124] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [125] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [126] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5543–5551, 2017.
- [127] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, 2017.
- [128] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.
- [129] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
- [130] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, 2011.
- [131] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.
- [132] Oscar Day and Taghi M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4, 2017.
- [133] Toshimitsu Aritake and Hideitsu Hino. Domain adaptation with optimal transport for extended variable space. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2022.

- [134] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [135] Wenpeng Hu, Ran Le, Bing Liu, Feng Ji, Jinwen Ma, Dongyan Zhao, and Rui Yan. Predictive adversarial learning from positive and unlabeled data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7806–7814, 2021.
- [136] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning, ICML '02*, page 387 – 394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.
- [137] Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. Pebl: Positive example based learning for web page classification using svm. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, page 239 – 248, New York, NY, USA, 2002. Association for Computing Machinery.
- [138] B. Liu, Y. Dai, X. Li, W.S. Lee, and P.S. Yu. Building text classifiers using positive and unlabeled examples. In *Third IEEE International Conference on Data Mining*, pages 179–186, 2003.
- [139] Chen Gong, Hong Shi, Tongliang Liu, Chuang Zhang, Jian Yang, and Dacheng Tao. Loss decomposition and centroid estimation for positive and unlabeled learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):918–932, 2021.
- [140] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1386–1394, Lille, France, 2015. PMLR.
- [141] Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In I. Guyon,

- U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [142] Jonas Sonntag, Gunnar Behrens, and Lars Schmidt-Thieme. Positive-unlabeled domain adaptation. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2022.
- [143] Paolo Mignone and Gianvito Pio. Positive unlabeled link prediction via transfer learning for gene network reconstruction. In Michelangelo Ceci, Nathalie Japkowicz, Jiming Liu, George A. Papadopoulos, and Zbigniew W. Raś, editors, *Foundations of Intelligent Systems*, pages 13–23, Cham, 2018. Springer International Publishing.
- [144] Paolo Mignone, Gianvito Pio, and Michelangelo Ceci. Distributed heterogeneous transfer learning for link prediction in the positive unlabeled setting. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5536–5541, 2022.
- [145] Mohammad Reza Loghmani, Markus Vincze, and Tatiana Tommasi. Positive-unlabeled learning for open set domain adaptation. *Pattern Recognition Letters*, 136:198–204, 2020.
- [146] Saurabh Garg, Sivaraman Balakrishnan, and Zachary Chase Lipton. Domain adaptation under open set label shift. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [147] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [148] Hui Tang and Kui Jia. Discriminative adversarial domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5940–5947, 2020.
- [149] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.
- [150] Movielens. 2019.

- [151] Netflix prize data. 2009.
- [152] Ken Lang. Newsweeder: Learning to filter netnews. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 331–339. Morgan Kaufmann, San Francisco (CA), 1995.
- [153] I-Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480, 2009.