DATA-CENTRIC PRIVACY-PRESERVING MACHINE LEARNING

By

Junyuan Hong

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2023

ABSTRACT

With the rapid popularization of machine learning (ML), privacy emerges as a critical obstacle for extracting knowledge from sensitive data into models. Traditional machine learning methods industriously curate data from millions of clients (e.g., edge devices) and train models upon the data, which causes tremendous risks of leaking data providers' sensitive information. In this thesis, we are devoted to exploiting learning algorithms that protect data providers' privacy based on three different ways of data use: from central to distributed settings. First, we investigate private data-centralized learning (CL) in the rigorous notion of differential privacy (DP), where we search for the iterative dynamic privacy allocation in gradient descent toward higher model utility. Our theoretic work shows that optimal privacy allocation can improve the sample efficiency of DP learning. Though data can be protected in learning, CL has to make the assumption that the data management institute can be trusted, which does not have technical guarantees, though. Rather than aggregating data, federated learning (FL) coordinates clients to periodically share models trained on local data. Confronting the great data and device heterogeneity from clients in FL, we propose novel algorithms that can effectively train models from clients with heterogeneous data distributions and device capabilities. One of our methods enhances the knowledge transfer from one supervised domain to an unsupervised domain, and can reduce the performance gap between clients from different social groups. We also developed a hardware-adaptive learning algorithm that makes FL inclusive for devices with various capabilities. Not only during training, our algorithm also enables models to be customizable at test time for facilitating dynamic computation budgets. Though FL mitigates the risks by distributed data, the local training of large models could still be a significant burden for resource-limited edge devices. Last, observing the limitations of CL in the privacy of data storage and FL in computation, we propose a new computation paradigm, outsourcing training without uploading data (OT). To learn effective knowledge about private data, we sample the proximal proxy dataset from the open-source data for cloud training. Our method can efficiently and effectively spot sim-

ilar samples from privacy-free open-source data, and therefore can transfer the computation costs of training to the cloud server.

This thesis is dedicated to my parents.

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my profound gratitude towards my advisor, Dr. Jiayu Zhou, whose invaluable counsel, motivation, and unwavering support have been indispensable throughout my doctoral voyage. Dr. Zhou's role in my academic journey transcends that of a mere advisor, concurrently acting as a trusted confidante who imparts his extensive knowledge and experience in various facets of my Ph.D. odyssey. My experience in Dr. Zhou's research laboratory has been nothing short of extraordinary, a consequence of a fortunate decision taken years prior. The wealth of knowledge that I've accumulated under his guidance is vast, ranging from efficacious research methodologies to the art of influential leadership. Ranked among the world's most esteemed advisors, Dr. Zhou's proclivity towards nurturing the growth of his students, demonstrated by his constant support and prompt responses, is unparalleled. His enduring support over the years has been a linchpin in my academic accomplishments, and for that, I will remain forever appreciative.

I would like to express my deepest appreciation to my dissertation committee members, Dr. Anil K. Jain, Dr. Sijia Liu, and Dr. Zhangyang Wang for their insightful comments and helpful suggestions. Their expertise and feedback have played a pivotal role in shaping both this dissertation and my job talk. I would like to extend my profound gratitude to Dr. Zhangyang Wang, who has diligently co-supervised me alongside Dr. Zhou throughout my doctoral journey. His unflagging zeal, remarkable intellect, and unwavering professionalism have continuously served as the ideal I aspire to emulate in my future endeavors.

I was fortunate to work as an intern at Sony AI with amazing colleagues and mentors: Dr. Lingjuan Lyu, Dr. Spranger Michael, Yuyang Deng, Yi Zeng, Dr. Nidham Gazagnadou, Dr. Virat Shejwalkar, Jiahua Dong, Dr. Chen Chen, Dr. Weiming Zhuang, Dr. Jingtao Li, Yubin Hu, and Ronghang Zhu. I enjoyed the wonderful and productive months with you.

I am grateful to have had the pleasure and fortune of having supportive and encouraging colleagues during my Ph.D. I am thankful to all my colleagues from the Intelligent Data Analytics (ILLIDAN) Lab: Zhuangdi Zhu, Mengying Sun, Kaixiang Lin, Boyang Liu, Qi

Wang, Inci Baytas, Ikechukwu Uchendu, Andy Tang, Liyang Xie, Jun Chen, Shuyang Yu, Haobo Zhang, Siqi Liang, Yijiang Pang, and Hoang Cao Bao. I would love to extend my deepest appreciations to my collaborators: Dr. Haotao Wang, Dr. Zhangyang Wang, Dr. Ruoxi Jia, Dr. Hiroko Dodge, Dr. Xiaomin Ouyang, Dr. Mehrdad Mahdavi, Dr. Yang Zhou, Dr. Vishnu Boddeti, Dr. Steve Drew, Dr. Aston Zhang, Dr. Zenglin Xu, and Dun Zeng.

I would also like to extend my heartfelt appreciation to my dear friends at MSU who provided unwavering support and encouragement during the moments when I felt frustrated during my Ph.D. study. Dr. Wei Wang, Dr. Kaixiong Zhou, Meijun Gao, Tianxudong Tang, Fei Zhang, Rundong Zhao, Xu Dong, Feiran Li, Xitong Zhang, Jingwen Shi, Wentao Wang, Xinyu Lei, Wei Jin, Wentao Bao, Junwen Chen, Xinda Qi, Pengyu Chu, Xiaorui Liu, Shaohua Yang, Qiaozi Gao, Dong Chen, Dr. Nan Du, Dr. Ding Wang, Mi Gong, Guangjing Wang, Ze Zhang, Ruoze Su, Hao Wang, Ruoqiao Chen, Pai Li, Xuan Xie, Cathy Bing, Zeyu Qin, Zhiyuan Ren, Shengjie Zhu, and Xinyi Wang, your friendship and belief in me have been truly invaluable.

Finally, I would like to thank my family and my girlfrind, Zhujie Hong, Yafen Chen and Han Meng, for their unconditional love and support.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Motivation 1
1.2 Overview of Thesis Structure 2
1.3 Defitinion of Privacy-Preserving Learning 3
1.4 A Glance of Prior Arts 4

CHAPTER 2 DYNAMIC PRIVACY BUDGET ALLOCATION FOR ENHANCED CENTRALIZED PRIVATE LEARNING 7
2.1 Introduction 7
2.2 Related Work 9
2.3 Private Gradient Descent 11
2.4 Dynamic Policies by Minimizing Utility Upper Bounds 14
2.5 Experiments 27

CHAPTER 3 FEDERATED ADVERSARIAL DEBIASING FOR TRANSFERABLE AND FAIR REPRESENTATIONS 29
3.1 Introduction 29
3.2 Related Work 32
3.3 Federated Adversarial Debiasing 33
3.4 Optimality Analysis 35
3.5 Experiments on Unsupervised Domain Adaptation 40
3.6 Experiments on Fair Federated Learning 45

CHAPTER 4 EFFICIENT FEDERATED LEARNING FOR ON-DEMAND AND IN-SITU CUSTOMIZATION 49
4.1 Introduction 49
4.2 Related Work 52
4.3 Problem Setting 54
4.4 Method 55
4.5 Empirical Studies 61

CHAPTER 5 OUTSOURCING TRAINING WITHOUT UPLOADING DATA . . . 67
5.1 Introduction 67
5.2 Related Work 70
5.3 Outsourcing Model Training With Open-Source Data 71
5.4 Empirical Results 76

CHAPTER 6 OVERVIEW 84

APPENDIX A DYNAMIC PRIVACY BUDGET ALLOCATION 101

APPENDIX B FEDERATED ADVERSARIAL DEBIASING FOR FAIR AND TRANSFERABLE REPRESENTATIONS 104

APPENDIX C EFFICIENT FEDERATED LEARNING FOR ON-DEMAND AND IN-SITU CUSTOMIZATION 111

APPENDIX D OUTSOURCING TRAINING WITHOUT UPLOADING DATA . . 121

CHAPTER 1

INTRODUCTION

1.1 Motivation

When machine learning becomes the fundamental block in artificial intelligence, the recent advances in deep machine learning hinge on a large amount of data. Such great demands for massive data cast significant challenges on the broader applications of deep learning in practical scenarios, where data are collected from the amount of distributed personal devices containing private information and therefore cannot be freely collected without *privacy, communication, and computation costs*. For example, the widely-used ImageNet dataset [26] gathers millions of images from Internet, which are captured by camera sensors distributed across the world. When human objects are involved in the system, the privacy of the personal information (including identities, activities, and preferences) in the photos has to be carefully protected.

In this thesis, we study privacy protection from a data-centric perspective: *How data are used in private learning?*. We answer this question from three ways of data use. **1) Centralize data.** First, centralized learning has been a long-term topic, where all data are gathered into an institute to conduct follow-up learning. For centralized data storage, the institute responsible for data management is promised to protect the users' data, where the model delivered by centralized learning should not leak private information by any means. **2) Distributed data and training.** However, the storage institute may break the trust when the clients can not monitor the data use. For example, Facebook sells the users' data to Cambridge Analytics without acknowledging users [2]. The leakage strengthens people's doubts about the trustworthiness of big corporations. Therefore, in the last decades, many countries or organizations have carried out laws to constrain the use of private data by any institute, including General Data protection Regulation [1], and Health Insurance Portability and Accountability Act (HIPPA) [4]. **3) Distributed data but centralized training.**

Even if the data storage is trustworthy, the trained model may be inverse-engineered to leak private information on publishing [45]. For distributed learning, private information can be mined by exploring the transmitted gradients [166]. Meanwhile, the computation overhead brought by local training calls for full cloud training without accessing private data.

Observing the risks of data privacy, we aim to protect privacy in machine learning from a data-centric perspective and focus on efficient, effective, and flexible privacy-preserving learning. First, we use the notion of differential privacy (DP) [38] to quantify the privacy risks in centralized learning and design algorithms for sample-efficient DP learning. Second, we consider distributed data and training based on a federated learning framework, where data and computation tasks are distributed to clients with heterogeneous data, devices, and demands. The heterogeneity of clients makes it difficult to accommodate the federated learning to more data and clients and therefore calls for advanced learning algorithms that are flexible with hardware and data biases. Third, in our ongoing work, we consider cloud training where data are maintained locally by a client, and computation-intensive training is done on the cloud. To train a model that is effective on local data, it is essential to find proxy data that is privacy-free and consists of similar features as the private data, critically via a privacy-preserving collaboration.

1.2 Overview of Thesis Structure

This section summarizes each of the chapters in this thesis.

Chapter 2 elaborates on the theoretical analysis of differentially-private gradient descent to establish the utility-enhanced convergence bound. Chapter 3 discuss how to close the domain or group biases in federated learning towards fair and transferable representations. In Chapter 4, we study the heterogeneity of hardware and the split-mix principle for model-customizable training and inference. In Chapter 5, we present our ongoing work for out-sourcing training without uploading data by sampling a proximal dataset from open-source data.

Abbreviation	Definition
DP	Differential Privacy
GD	Gradient Descent
PGD	Private Gradient Descent
FL	Federated Learning

Table 1.1: Overview of Abbreviations.

1.3 Defitinion of Privacy-Preserving Learning

In this section, we present definitions of privacy-preserving learning (PPL) in three different ways of data use.

1.3.1 Private Gradient Descent for Centralized Learning

We consider a learning task by empirical risk minimization (ERM) $f(\theta) = \frac{1}{N}\sum_{n=1}^{N} f(\theta; x_n)$ on a private dataset $\{x_n\}_{n=1}^{N}$ and $\theta \in \mathbb{R}^D$. The gradient methods are defined as $\theta_{t+1} = \theta_t - \eta_t \nabla_t$, where $\nabla_t = \nabla f(\theta_t) = \frac{1}{N}\sum_n \nabla f(\theta_t; x_n)$ denotes the non-private gradient at iteration t, η_t is the step learning rate. $\nabla_t^{(n)} = \nabla f(\theta_t; x_n)$ denotes the gradient on a sample x_n. $\mathbb{I}_c$ denotes the indicator function that returns 1 if the condition c holds, otherwise 0.

Generally, we define the Private Gradient Descent (PGD) method as iterations for $t = 1 \dots T$:

$$\theta_{t+1} = \theta_t - \eta_t \phi_t = \theta_t - \eta_t(\nabla_t + \sigma_t G \nu_t / N), \tag{1.1}$$

where $\phi_t = g_t$ is the gradient privatized from ∇_t as shown in Algorithm 2.1, G/N is the bound of sensitivity of the gathered gradient excluding one sample gradient, and $\nu_t \sim \mathcal{N}(0, I)$ is a vector element-wisely subject to Gaussian distribution. We use σ_t to denote the noise scale at step t and use σ to collectively represents the schedule $(\sigma_1, \dots, \sigma_T)$ if not confusing. When the Lipschitz constant is unknown, we can control the upper bound by scaling the gradient if it is over some constant. The scaling operation is often called *clipping* in literature since it clips the gradient norm at a threshold. After the gradient is noised, we apply a modification, $\phi(\cdot)$, to enhance its utility.

1.3.2 Data-Distributed Learning via Federated Learning

As centralizing data into an untrustworthy center causes uncontrollable risks, learning with distributed data and computation could alleviate the issue accordingly. One extraordinary example of this distributed learning is to federated learning (FL) strategy, which yields a global model by aggregating models from many clients. Formally, FL minimizes the objective

$$\frac{1}{\sum_k |D_k|} \sum_{k=1}^{K} \sum_{(x,y)\in D_k} L(f(x;\theta), y)$$

where L is loss function (e.g., cross-entropy loss), f is a model parameterized by θ, and $\{D_k\}_{k=1}^{K}$ are the training datasets on K participants with the image-label pairs (x, y). Following the standard FL setting [96], only model parameters can be shared to protect privacy.

1.3.3 Outsourcing Training without Uploading Private Data

As FL requires intensive local training, it is less suitable for resource-constrained edge devices. To transfer the computation overhead to cloud without using private data, the goal of Outsourcing Training without Uploading Private Data (OT) is to train a model on a proxy dataset proxy such that the model can transfer to the target private data distribution. The training can be formulated as the empirical risk minimization on the cloud dataset. The loss is defined as

$$\mathcal{L}(\theta; D_{\text{proxy}}) = \frac{1}{|D_{\text{proxy}}|} \sum_{(x,y)\in D_{\text{proxy}}} L(f(x;\theta), y).$$

A good proxy dataset should make the trained model transferable to the private data, for which purpose the proxy data should follow a similar distribution as the private data. To circumvent the privacy risk, the proxy dataset should be constructed from a non-private source and any interaction with the private data should be accounted for with privacy risks.

1.4 A Glance of Prior Arts

In this section, we briefly overview the recent advances of privacy-preserving learning approaches in three settings.

Centralized Private Learning. *(1) Private Learning.* Recent years witnessed increasing attention to the privacy risk associated to learning from sensitive training data. For example, an attacker could retrieve the training data from the models generated by the widely-used empirical risk minimization (ERM) [45]. Many efforts have been devoted to privacy-preserving learning. With the introduction of differential privacy (DP) [34], we are now able to measure and defend the risk quantitatively [11, 70, 120, 128]. The main idea is to introduce stochastic perturbations to the learning process, and the perturbations can be done in any query operations [35], such as gradient computations [3] or objective evaluation [20]. When proper noise is introduced before publishing the model, such as Gaussian mechanism [39], one can no longer easily retrieve the training data by resampling [45]. *(2) Adaptive Privacy Perturbation.* The key to achieving high utility in privacy-preserving learning is perturbation control. [162] improved the performance of models in a stochastically private manner by selecting the gradient candidates. [75] proposed adaptively and privately querying the effects of the noised gradient updates. Both mechanisms rely on querying the model outputs several times via an exponential noise mechanism [34] which degrades the effectiveness. Instead, [9] showed a simple adaptive scaling based on the noised value is capable of reducing expectation error. Inspired by this idea, our gradient protector sequentially predicts optimization updates based on current and previous protected gradients which reduces both query times and privacy costs.

Federated Learning [96] is a distributed learning framework that allows users with different capabilities to collaboratively train a model without sharing their own data. A critical challenge in FL is the heterogeneity among users. Viewing the learning process of FL as knowledge transfer among different users, heterogeneity in user data leads to negative transfer between users and compromises generalization [13]. One idea to alleviate the negative effect of heterogeneity during the training is to find a consensus among users. For example, in [49, 85, 77, 43], the consensus on task knowledge is achieved by distillation. In this work, we seek an alternative and efficient approach by adversarial debiasing the users

of different groups.

Training Outsourcing. There are a series of efforts studying how to leverage the data and computation resources on the cloud to assist client model training, especially when client data cannot be shared [156, 142]. We categorize them as follows: *1) Feature sharing*: Methods like group knowledge transfer [49], split learning [136] and domain adaptation [32, 30] transfer edge knowledge by communicating features extracted by networks. To provide a theoretical guarantee of privacy protection, [105] proposed an advanced information removal to disentangle sensitive attributes from shared features. In the notion of rigorous privacy definition, Liu *et al.* leverage public data to assist private information release [89]. Earlier, data encryption was used for outsourcing which however is too computation-intensive for a client and less applicable for large-scale data and deep networks [21, 76]. Federated Learning (FL) considered the same constraint on data sharing but the private knowledge is shared via models trained locally [97]. *2) Private labeling:* PATE and its variants were proposed to generate client-approximated labels for unlabeled public data, on which a model can be trained [108, 109]. Without training multiple models by clients, Private kNN was a more efficient alternative that explored the private neighborhood of public images for labeling [168]. These approaches are based on a strong assumption of the availability of public data that is *iid* as the local data. This paper considers a more practical yet challenging setting where public data are from multiple agnostic sources with heterogeneous features.

CHAPTER 2

DYNAMIC PRIVACY BUDGET ALLOCATION FOR ENHANCED CENTRALIZED PRIVATE LEARNING

Protecting privacy in learning while maintaining the model performance has become increasingly critical in many applications that involve sensitive data. Private Gradient Descent (PGD) is a commonly used private learning framework, which noises gradients based on the Differential Privacy protocol. Recent studies show that *dynamic privacy schedules* of decreasing noise magnitudes can improve loss at the final iteration, and yet theoretical understandings of the effectiveness of such schedules and their connections to optimization algorithms remain limited. In this paper, we provide comprehensive analysis of noise influence in dynamic privacy schedules to answer these critical questions. We first present a dynamic noise schedule minimizing the utility upper bound of PGD, and show how the noise influence from each optimization step collectively impacts utility of the final model. Our study also reveals how impacts from dynamic noise influence change when momentum is used. We empirically show the connection exists for general non-convex losses, and the influence is greatly impacted by the loss curvature.

2.1 Introduction

In the era of big data, privacy protection in machine learning systems is becoming a crucial topic as increasing personal data involved in training models [37] and the presence of malicious attackers [126, 45]. In response to the growing demand, differential-private (DP) machine learning [38] provides a computational framework for privacy protection and has been widely studied in various settings, including both convex and non-convex optimization [139, 138, 64].

One widely used procedure for privacy-preserving learning is the (Differentially) Private Gradient Descent (PGD) [11, 3]. A typical gradient descent procedure updates its model by gradients of the loss evaluated on a training dataset. When the data is sensitive, the gradients should be *privatized* to prevent excess privacy leakage. The PGD privatizes a gradient by

adding controlled noise. As such, the models from PGD is expected to have a lower utility as compared to those from unprotected algorithms. In the cases where strict privacy control is exercised, or equivalently, a tight *privacy budget*, accumulating effects from highly-noised gradients may lead to unacceptable model performance. It is thus critical to design effective privatization procedures for PGD to maintain a great balance between utility and privacy.

Recent years witnessed a promising direction of privatization that *dynamically allocate a privacy budget* for each iteration to boost utility, under the constraint of a specified total privacy budget. One example is [75], which reduces the budget-bonded noise magnitude when the loss does not decrease, due to the observation that gradients become very small when approaching convergence, and a static noise scale will overwhelm these gradients. Another example is [159], which periodically decreases the magnitude following a predefined strategy, e.g., exponential decaying or step decaying. Both approaches confirmed the empirically advantages of decreasing noise magnitudes. Intuitively, the dynamic mechanism may coordinate with certain properties of the learning task, e.g., training data and loss surface. Following the work, improved allocation policies are proposed, e.g., policies transferred from auxiliary datasets [57], policies with distributed budgets [22], and a combination with adaptive learning rate [155]. Yet there is little theoretical analysis available and two important questions remain unanswered: 1) *What is the form of utility-preferred budget (or noise equivalently) schedules?* 2) *When and to what extent such an allocation policy improves utility?*

Though there are theoretical studies of static-allocation policies, e.g., [139], the data efficiency is not the focus as discussions usually assume an unlimited amount of data is available. However, we argue that the data efficiency with limited data size is critical in practice, especially when DP makes the learning more data-hungry [98]. One example is federated learning [96, 98], a distributed learning framework that aggregates many local models to form a stronger global model, where each model is privately trained on a local client, typically with very limited private data. Another example is biomedical applications, where

collecting samples involves expansive clinical trials or cohort studies, resulting the scarcity of training set. To study biomarkers of Alzheimer's, NIH has funded Alzheimer's Disease Neuroimaging Initiative for $40 million, which collected imaging and genetic biomarkers from only 800 patients after 5 years' extensive and collaborative efforts [145]. Therefore, we believe data efficiency needs to be taken into account in developing a private learning algorithm.

To answer these questions, in this paper we develop a principled approach to construct dynamic schedules and quantify their utility bounds in different learning algorithms. Our contributions are summarized as follows. 1) For the class of loss functions satisfying the Polyak-Lojasiewicz condition [114], we show that dynamic schedules, that improve the utility upper bound with high data-efficiency, are shaped by the changing influence of per-iteration noise on the final loss. As the influence is tightly connected to the loss curvature, the advantage of using dynamic schedules therefore depends on the loss function. 2) Beyond vanilla gradient descent, our results show the gradient methods with momentum implicitly introduce a dynamic schedule and result in an non-monotonous influence trend. 3) We also show that our results are generalizable to population bounds in high probability or based on uniform stability theorems. Though our major focus is on the theoretic study, we empirically validate the results on a non-convex loss function formulated by a neural network. The empirical results suggest that a preferred dynamic schedule admits the exponentially decaying form, and works better when learning with high-curvature loss functions. Moreover, dynamic schedules give higher utility under stricter privacy conditions (e.g., smaller sample size and less privacy budget).

2.2 Related Work

Differentially Private Learning. Differential privacy (DP) characterizes the chance that an algorithm output (e.g., a learned model) leaks private information of its training data when the output distribution is known. Since outputs of many learning algorithms have undetermined distributions, the probabilistic risk is hard to measure. A common approach

to tackle this issue is to inject randomness with known distribution to *privatize* the learning procedures. Classical methods include output perturbation [20], objective perturbation [20] and gradient perturbation [3, 11, 148]. Among these approaches, the Private Gradient Descent (PGD) has attracted extensive attention in recent years because it can be flexibly integrated with variants of gradient-based iteration methods, e.g., stochastic gradient descent, momentum methods [116], and Adam [71], for (non-)convex problems.

Dynamic Policies for Privacy Protection. [139] studied the empirical risk minimization using dynamic variation reduction of perturbed gradients. They showed that the utility upper bound can be achieved by gradient methods under uniform noise parameters. Instead of enhancing the gradients, [159, 75] showed the benefits of using a dynamic schedule of privacy parameters or equivalently noise scales. Following [75], a series of work [22, 61, 152, 164] adaptively allocate privacy budget towards better privacy-utility trade-off. Moreover, adaptive sensitivity control [113, 130] and dynamic batch sizes [42] are also shown to improve convergence.

Utility Upper Bounds. Utility upper bounds are a critical metric for privacy schedules, which characterizes the maximum utility that a schedule can deliver in theory. [139] is the first to prove the utility bound under the PL condition. Recently, [165] proved the utility bound by using the momentum of gradients [115, 71]. In this paper, we improve the upper bound by a more accurate estimation of the dynamic influence of step noise. We show that introducing a dynamic schedule further boosts the sample-efficiency of the upper bound. Table 2.1 summarizes the upper bounds of a selection of state-of-the-art algorithms based on private gradients (up block, see Section A.2 for the full list), and methods studied in this paper (down block), showing the benefits of dynamic influence.

Especially, a closely-related work by Feldman *et al.* achieved a convergence rate similar to ours in terms of generalization error bounds, by dynamically adjusting batch sizes [42]. However, the approach requires controllable batch sizes, which may not be feasible in many applications. In federated learning, for example, where users update models locally and then

Table 2.1: Comparison of utility upper bound using different privacy schedules. The algorithms are T-iteration $\frac{1}{2}R$-zCDP under the PL condition (unless marked with * for convexity). The O notation in this table drops other ln terms. Unless otherwise specified, all algorithms (including non-private GD) terminate at step $T = \mathcal{O}(\ln \frac{N^2 R}{D})$. Assume loss functions are 1-smooth and 1-Lipschitz continuous, and all parameters satisfy their numeric assumptions. Key notations: $\mathcal{O}_p$ – bound occurs in probability p; D – feature dimension; N – sample size; R – privacy budget where $R_{\epsilon,\delta}$ is the equivalent budget accounted by (ϵ, δ)-DP; c_i – constant.

Algorithm	Schedule (σ_t^2)	Utility Upper Bound
*GD+Adv [11]	$\mathcal{O}\left(\frac{\ln(N/\delta)}{R_{\epsilon,\delta}}\right)$	$\mathcal{O}\left(\frac{D \ln^3 N}{N R_{\epsilon,\delta}}\right)$
GD+MA [139]	$\mathcal{O}(\frac{T}{R_{\epsilon,\delta}})$	$\mathcal{O}\left(\frac{D \ln^2 N}{N^2 R_{\epsilon,\delta}}\right)$
GD+MA (adjusted utility) [157]	$\mathcal{O}(\frac{T}{R_{\epsilon,\delta}})$	$\mathcal{O}\left(\min \frac{\sqrt{D}}{N R_{\epsilon,\delta}}, \frac{D \ln N}{N^2 R_{\epsilon,\delta}^2}\right)$
*GD+Adv+BBImp [24]	$\mathcal{O}\left(\frac{n^2 \ln(n/\delta)}{R_{\epsilon,\delta}}\right)$	$\mathcal{O}_p\left(\frac{D^2 \ln^2(1/p)}{R_{\epsilon,\delta} N^{1-c}}\right)$
Adam+MA [165]	$\mathcal{O}(\frac{T}{R_{\epsilon,\delta}})$	$\mathcal{O}_p\left(\frac{\sqrt{D} \ln(ND\epsilon/(1-p))}{N R_{\epsilon,\delta}}\right)$
GD, Non-Private	0	$\mathcal{O}\left(\frac{D}{N^2 R}\right)$
GD+zCDP, Static Schedule	$\frac{T}{R}$	$\mathcal{O}\left(\frac{D \ln N}{N^2 R}\right)$
GD+zCDP, Dynamic Schedule	$\mathcal{O}\left(\frac{\gamma^{(t-T)/2}}{R}\right)$	$\mathcal{O}\left(\frac{D}{N^2 R}\right)$
Momentum+zCDP, Static Schedule	$\frac{T}{R}$	$\mathcal{O}\left(\frac{D}{N^2 R}(c + \ln N \mathbb{I}_{T>\hat{T}})\right)$
Momentum+zCDP, Dynamic Schedule	$\mathcal{O}\left(\frac{c_1 \gamma^{T+t} + c_2 \gamma^{(T-t)/2}}{R}\right)$	$\mathcal{O}\left(\frac{D}{N^2 R}(1 + \frac{cD}{N^2 R} \mathbb{I}_{T>\hat{T}})\right)$

pass the parameters to server for aggregation, the server has no control over batch sizes, and coordinating users to use varying batch sizes may not be realistic. On the other hand, our proposed method can still be applied for enhancing utility, as the server can dynamically allocate privacy budget for each round when the presence of a user in the global aggregation is privatized [98].

2.3 Private Gradient Descent

Notations. We consider a learning task by empirical risk minimization (ERM) $f(\theta) = \frac{1}{N} \sum_{n=1}^{N} f(\theta; x_n)$ on a private dataset $\{x_n\}_{n=1}^{N}$ and $\theta \in \mathbb{R}^D$. The gradient methods are defined as $\theta_{t+1} = \theta_t - \eta_t \nabla_t$, where $\nabla_t = \nabla f(\theta_t) = \frac{1}{N} \sum_n \nabla f(\theta_t; x_n)$ denotes the non-private gradient at iteration t, η_t is the step learning rate. $\nabla_t^{(n)} = \nabla f(\theta_t; x_n)$ denotes the gradient on a sample x_n. $\mathbb{I}_c$ denotes the indicator function that returns 1 if the condition c holds, otherwise 0.

Assumptions. (1) In this paper, we assume $f(\theta)$ is continuous and differentiable. Many commonly used loss functions satisfy this assumption, e.g., the logistic function. (2) For a learning task, only finite amount of privacy cost is allowed where the maximum cost is called *privacy budget* and denoted as R. (3) Generally, we assume that loss functions $f(\theta; x)$ (sample-wise loss) are G-Lipschitz continuous and $f(\theta)$ (the empirical loss) is M-smooth.

Definition 1 (G-Lipschitz continuity)**.** *A function $f(\cdot)$ is G-Lipschitz continuous if, for $G > 0$ and all x, y in the domain of $f(\cdot)$, $f(\cdot)$ satisfies $\|f(y) - f(x)\| \leq G \|y - x\|$. .*

Definition 2 (m-strongly convexity)**.** *A function $f(\cdot)$ is m-strongly convex if $f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|y - x\|^2$, for some $m > 0$ and all x, y in the domain of $f(\cdot)$.*

Definition 3 (M-smoothness)**.** *A function is M-smooth w.r.t. l_2 norm if $f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{M}{2}\|y - x\|^2$, for some constant $M > 0$ and all x, y in the domain of $f(\cdot)$.*

For a private algorithm $\mathcal{M}(d)$ which maps a dataset d to some output, the privacy cost is measured by the bound of the output difference on the adjacent datasets. *Adjacent datasets* are defined to be datasets that only differ in one sample. In this paper, we use the zero-Concentrated Differential Privacy (zCDP, see Definition 4) as the privacy measurement, because it provides the simplicity and possibility of adaptively composing privacy costs at each iteration. Various privacy metrics are discussed or reviewed in [27]. A notable example is Moment Accoutant (MA) [3], which adopts similar principle for composing privacy costs while is less tight for a smaller privacy budget. We note that alternative metrics can be adapted to our study without major impacts to the analysis.

Definition 4 (ρ-zCDP [18])**.** *Let $\rho > 0$. A randomized algorithm $\mathcal{M} : \mathcal{D}^n \to \mathbb{R}$ satisfies ρ-zCDP if, for all adjacent datasets $d, d' \in \mathcal{D}^n$, $D_\alpha(\mathcal{M}(d) \| \mathcal{M}(d')) \leq \rho\alpha,\ \forall \alpha \in (1, \infty)$ where $D_\alpha(\cdot\|\cdot)$ denotes the Rényi divergence [119] of order α.*

The zCDP provides a linear composition of privacy costs of sub-route algorithms. When the input vector is privatized by injecting Gaussian noise of $\mathcal{N}(0, \sigma_t^2 I)$ for the t-th iteration,

the composed privacy cost is proportional to $\sum_t \rho_t$ where the step cost is $\rho_t = \frac{1}{\sigma_t^2}$. For simplicity, we absorb the constant coefficient into the (residual) *privacy budget* R. The formal theorems for the privacy cost computation of composition and Gaussian noising is included in Lemmas 2 and 3.

Algorithm 2.1: Privatizing Gradients

Input: Raw gradients $[\nabla_t^{(1)}, \dots, \nabla_t^{(n)}]$ ($n = N$ by default), v_t, residual privacy budget R_t assuming the full budget is R and $R_1 = R$.

$\rho_t \leftarrow 1/\sigma_t^2$, $\nabla_t \leftarrow \frac{1}{n}\sum_{i=1}^{n} \nabla_t^{(i)}$ (Budget request)
if $\rho_t < R_t$ **then**
    $R_{t+1} \leftarrow R_t - \rho_t$
    $g_t \leftarrow \nabla_t + G\sigma_t\nu_t/N$, $\nu_t \sim \mathcal{N}(0, I)$ (Privacy noise)
    $m_{t+1} \leftarrow \phi(m_t, g_t)$ or g_1 if $t = 1$
    return $\eta_t m_{t+1}$, R_{t+1} (Utility projection)
else
    Terminate

Generally, we define the Private Gradient Descent (PGD) method as iterations for $t = 1 \dots T$:

$$\theta_{t+1} = \theta_t - \eta_t\phi_t = \theta_t - \eta_t(\nabla_t + \sigma_t G\nu_t/N), \tag{2.1}$$

where $\phi_t = g_t$ is the gradient privatized from ∇_t as shown in Algorithm 2.1, G/N is the bound of sensitivity of the gathered gradient excluding one sample gradient, and $\nu_t \sim \mathcal{N}(0, I)$ is a vector element-wisely subject to Gaussian distribution. We use σ_t to denote the noise scale at step t and use σ to collectively represents the schedule $(\sigma_1, \dots, \sigma_T)$ if not confusing. When the Lipschitz constant is unknown, we can control the upper bound by scaling the gradient if it is over some constant. The scaling operation is often called *clipping* in literatures since it clips the gradient norm at a threshold. After the gradient is noised, we apply a modification, $\phi(\cdot)$, to enhance its utility. In this paper, we consider two types of $\phi(\cdot)$:

$$\phi(m_t, g_t) = g_t \text{ (GD)}, \tag{2.2}$$

$$\phi(m_t, g_t) = [\beta(1 - \beta^{t-1})m_t + (1 - \beta)g_t]/(1 - \beta^t) \text{ (Momentum)} \tag{2.3}$$

We now show that the PGD using Algorithm 2.1 guarantees a privacy cost less than R:

Theorem 1. *Suppose $f(\theta; x)$ is G-Lipschitz continuous and the PGD algorithm with privatized gradients defined by Algorithm 2.1, stops at step T. The PGD algorithm outputs θ_T and satisfies ρ-zCDP where $\rho \leq \frac{1}{2}R$.*

Note that Theorem 1 allows σ_t to be different throughout iterations. Next we present a principled approach for deriving dynamic schedules optimized for the final loss $f(\theta_T)$.

2.4 Dynamic Policies by Minimizing Utility Upper Bounds

To characterize the utility of the PGD, we adopt the Expected Excess Risk (EER), which notion is widely used for analyzing the convergence of random algorithms, e.g., [11, 139]. Due to the presence of the noise and the limitation of learning iterations, optimization using private gradients is expected to reach a point with a higher loss (i.e., excess risk) as compared to the optimal solution without private protection. Define $\theta^* = \arg\min_\theta f(\theta)$, after Algorithm 2.1 is iterated for T times in total, the EER gives the expected utility degradation:

$$\text{EER} = \mathbb{E}_\nu[f(\theta_{T+1})] - f(\theta^*).$$

Due to the variety of loss function and complexity of recursive iterations, an exact EER with noise is intractable for most functions. Instead, we study the worst case scenario, i.e., the upper bound of the EER, and our goal is to minimize the upper bound. For consistency, we call the upper bound of EER divided by the initial error as ERUB. Since the analytical form of EER is either intractable or complicated due to the recursive iterations of noise, studying the ERUB is a convenient and tractable alternative. The upper bound often has convenient functional forms which are (1) sufficiently simple, such that we can directly minimize it, and (2) closely related to the landscape of the objective depending on both the training dataset and the loss function. As a consequence, it is also used in previous PGD literature [113, 139] for choosing proper parameters. Moreover, we let $\text{ERUB}_{\min}$ be the achievable optimal upper bound by a specific choice of parameters, e.g., the σ and T.

As the EER is iteratively determined by Eq. (2.1), we define the influence of the dynamics in noise magnitude σ_t as the derivative: $q_t^* = \frac{\partial \text{EER}}{\partial \sigma_t}$. Accordingly, we can approximate the EER shift as $q_t^* \Delta\sigma_t$ when σ_t increases by $\Delta\sigma_t$. However, because the EER is strongly data-dependent, the derived q_t^* on a given dataset may not generalize to another dataset. Instead, we consider a more general term based on ERUB, i.e., $q_t = \frac{\partial \text{ERUB}}{\partial \sigma_t}$.

In this paper, we consider the class of loss functions satisfying the Polyak-Lojasiewicz (PL) condition which bounds losses by corresponding gradient norms. It is more general than the m-strongly convexity. If f is differentiable and M-smooth, then m-strongly convexity implies the PL condition.

Definition 5 (Polyak-Lojasiewicz condition [114])**.** *For $f(\theta)$, there exists $\mu > 0$ and for every θ, $\|\nabla f(\theta)\|^2 \geq 2\mu(f(\theta) - f(\theta^*))$.*

The PL condition helps us to reveal how the influence of step noise propagates to the final excess error, i.e., EER. Though the assumption was also used previously in **(author?)** [139, 165], neither did they discuss the propagated influence of noise. In the following sections, we will show how the influence can tighten the upper bound in gradient descent and its momentum variant.

2.4.1 Gradient Descent Methods and Noise Influences

For the brevity of variables, we first define the following summarized constants:

$$\text{non-private ERUB} : \alpha \triangleq \frac{DG^2}{2RMN^2(f(\theta_1) - f(\theta^*))} \leq \mathcal{O}\left(\frac{DG^2}{RMN^2}\right), \tag{2.4}$$

$$\text{curvature} : \kappa \triangleq \frac{M}{\mu}, \tag{2.5}$$

$$\text{convergence rate} : \gamma \triangleq 1 - \frac{1}{\kappa}, \tag{2.6}$$

which satisfy $\kappa \geq 1$ and $\gamma \in [0, 1)$. Here, α is upper bounded by non-private ERUB within $T = \left\lceil \mathcal{O}(\ln \frac{N^2R(M-\mu)}{DG^2}) \right\rceil$ iterations. Therefore, α provide a simple reference of an ideal convergence bound, reaching which indicates a superior performance with privacy guarantee. κ characterizes the curvature of $f(\cdot)$ which is the condition number of $f(\cdot)$ if $f(\cdot)$ is strongly

convex, and γ is the convergence rate for non-private SGD (c.f. Theorem 2 with $\sigma_t = 0$). κ tends to be large if the function is sensitive to small differences in inputs, and $1/\alpha$ tends to be large if more samples are provided and with a less strict privacy budget. The convergence of PGD under the PL condition has been studied for private [139] and non-private [68, 101, 118] ERM. Below we extend the bound in [139] by considering dynamic influence of noise and relax σ_t to be dynamic:

Theorem 2. *Let α, κ and γ be defined in Eq. (2.6), and $\eta_t = \frac{1}{M}$. Suppose $f(\theta; x_i)$ is G-Lipschitz and $f(\theta)$ is M-smooth satisfying the Polyak-Lojasiewicz condition. For PGD, the following holds:*

$$\text{ERUB} = \gamma^T + R\sum\nolimits_{t=1}^{T} q_t \sigma_t^2, \text{ where } q_t \triangleq \gamma^{T-t}\alpha. \tag{2.7}$$

Theorem 2 degenerates to a non-private variant as no noise is applied, i.e., $\sigma_t = 0$ for all t. In Eq. (2.7), the step noise magnitude σ_t^2 has an exponential *influence*, q_t, on the EER. Note we ignore the constant factor R in the influence. The Eq. (2.7) implies that the influence of noise at step t increase quickly by an exponential rate. Importantly, the increasing rate is the same as the convergence rate, i.e., the first term in Eq. (2.7). The dynamic characteristic of the influence is the key to prove a tighter bound. Plus, on the presence of the dynamic influence, it is natural to choose a dynamic σ_t^2. When relaxing q_t to a static 1, a static σ_t^2 was studied by [139] They proved a bound which is nearly optimal except a $\ln^2 N$ factor. To get the optimal bound, in the following sections, we look for the σ and T that minimize the upper bound.

Uniform Schedule. The uniform setting of σ_t has been previously studied in [139]. Here, we show that the bound can be further tightened by considering the dynamic influence of iterations and a proper T.

Theorem 3. *Suppose conditions in Theorem 2 are satisfied. When $\sigma_t^2 = T/R$, let α, γ and κ be defined in Eq. (2.6) and let T be: $T = \left\lceil \mathcal{O}\left(\kappa \ln\left(1 + \frac{1}{\kappa\alpha}\right)\right)\right\rceil$. Meanwhile, if $\kappa \geq \frac{1}{1-c} > 1$,*

$1/\alpha > 1/\alpha_0$ *for some constant* $c \in (0,1)$ *and* $\alpha_0 > 0$*, the corresponding bound is:*

$$\mathrm{ERUB}_{\min}^{uniform} = \Theta\left(\frac{\kappa^2}{\kappa + 1/\alpha} \ln\left(1 + \frac{1}{\kappa\alpha}\right)\right). \tag{2.8}$$

Sketch of proof. The key of proof is to find a proper T to minimize

$$\begin{aligned} \mathrm{ERUB} = E &= \gamma^T + \sum\nolimits_{t=1}^{T} \gamma^{T-t} \alpha R \sigma^2 \\ &= \gamma^T + \alpha T \frac{1-\gamma^T}{1-\gamma} = \gamma^T + \alpha\kappa(1-\gamma^T)T \end{aligned} \tag{2.9}$$

where we use $\sigma_t = \sqrt{T/R}$. Vanishing its gradient is to solve $\gamma^T \ln\gamma + \alpha\kappa(1-\gamma^T) - \alpha\kappa T\gamma^T \ln\gamma = 0$, which however is intractable. In [139], T is chosen to be $\mathcal{O}(\ln(1/\alpha))$ and ERUB is relaxed as $\gamma^T + \alpha\kappa T^2$. The approximation results in a less tight bound as $\mathcal{O}(\alpha(1 + \kappa \ln^2(1/\alpha)))$ which explodes as $\kappa \to \infty$.

We observe that for a super sharp loss function, i.e., a large κ, any minor perturbation may result in tremendously fluctuating loss values. In this case, not-stepping-forward will be a good choice. Thus, we choose $T = \frac{1}{\ln(1/\gamma)} \ln\left(1 + \frac{\ln(1/\gamma)}{\alpha}\right) \leq \mathcal{O}\left(\kappa \ln\left(1 + \frac{1}{\kappa\alpha}\right)\right)$ which converges to 0 as $\kappa \to +\infty$. The full proof is deferred to the appendix. □

Dynamic Schedule. A dynamic schedule can improve the upper bound delivered by the uniform schedule. First, we observe that the excess risk in Eq. (2.7) is upper bounded by two terms: the first term characterizes the error due to the finite iterations of gradient descents; the second term, a weighted sum, comes from error propagated from noise at each iteration. Now we show for any $\{q_t | q_t > 0, t = 1, \ldots, T\}$ (not limited to the q_t defined in Eq. (2.7)), there is a unique σ_t minimizing the weighted sum:

Lemma 1 (Dynamic schedule)**.** *Suppose* σ_t *satisfy* $\sum_{t=1}^{T} \sigma_t^{-2} = R$*. Given a positive sequence* $\{q_t\}$*, the following equation holds:*

$$\min_{\sigma} R \sum\nolimits_{t=1}^{T} q_t \sigma_t^2 = \left(\sum\nolimits_{t=1}^{T} \sqrt{q_t}\right)^2, \text{ when } \sigma_t^2 = \frac{1}{R} \sum\nolimits_{i=1}^{T} \sqrt{\frac{q_i}{q_t}}. \tag{2.10}$$

Remarkably, the difference between the minimum and $T \sum_{t=1}^{T} q_t$ *(uniform* σ_t*) monotonically increases by the variance of* $\sqrt{q_t}$ *w.r.t.* t*.*

We see that the dynamics in σ_t come from the non-uniform nature of the weight q_t. Since q_t presents the impact of the σ_t on the final error, we denote it as *influence*. Given the dynamic schedule in Eq. (2.10), it is of our interest to which extent the ERUB can be improved. First, we present Theorem 4 to show the optimal T and ERUB.

Theorem 4. *Suppose conditions in Theorem 2 are satisfied. Let α, κ and γ be defined in Eq. (2.6). When $\eta_t = \frac{1}{M}$, σ_t (based on Eqs. (2.7) and (2.10)) and the T minimizing ERUB are, i.e., $\sigma_t^2 = \frac{1}{R}\frac{\sqrt{(1/\gamma)^T}-1}{1-\sqrt{\gamma}}\sqrt{\gamma^t}$, $T = \left\lceil \left(2\kappa \ln\left(1+\frac{1}{\kappa\alpha}\right)\right)\right\rceil$. Meanwhile, when $\kappa \geq 1$ and $1/\alpha \geq 1/\alpha_0$ for some positive constant α_0, the minimal bound is:*

$$\mathrm{ERUB}_{\min}^{dynamic} = \Theta\left(\frac{\kappa^2}{\kappa^2 + 1/\alpha}\right). \tag{2.11}$$

2.4.2 Discussion

In Theorems 3 and 4, we present the tightest bounds for functions satisfying the PL condition, to our best knowledge. We further analyze the advantages of our bounds from two aspects: sample efficiency and robustness to sharp losses.

Sample efficiency. Since dataset cannot be infinitely large, it is critical to know how accurate the model can be trained privately with a limited number of samples. Formally, it is of interest to study when κ is fixed and N is large enough such that $\alpha \gg 1$. Then we have the upper bound in Eq. (2.8) as

$$\mathrm{ERUB}_{\min}^{\mathrm{uniform}} \leq \mathcal{O}\left(\kappa^2 \alpha \ln\left(\frac{1}{\kappa\alpha}\right)\right) \leq \tilde{\mathcal{O}}\left(\frac{DG^2\ln(N)}{MN^2R}\right), \tag{2.12}$$

where we ignore κ and other logarithmic constants with $\tilde{\mathcal{O}}$ as done in **(author?)** [139]. As a result, we get a bound very similar to [139], except that R is replaced by $R_{MA} = \epsilon^2/\ln(1/\delta)$ using Moment Accountant. In comparison, based on Lemma 4, $R = 2\rho = 2\epsilon + 4\ln(1/\delta) + 4\sqrt{\ln(1/\delta)(\epsilon + \ln(1/\delta)}$ if θ_T satisfies ρ-zCDP. Because $\ln(1/\delta) > 1$, it is easy to see $R = R_{zCDP} > R_{MA}$ when $\epsilon \leq 2\ln(1/\delta)$. As compared to the one reported in [139], our bound saved a factor of $\ln N$ and thus require less sample to achieve the same accuracy.

Remarkably, the saving is due to the maintaining of the influence terms as shown in the proof of Theorem 3.

Using the dynamic schedule, we have $\text{ERUB}_{\min}^{\text{dynamic}} \leq \mathcal{O}(\alpha) = \mathcal{O}\left(\frac{DG^2}{MN^2R}\right)$, which saved another $\ln N$ factor in comparison to the one using the uniform schedule Eq. (2.12). As shown in Table 2.1, such advantage maintains when comparing with other baselines and reaches the ideal non-private case , recalling the meaning of α.

Stability on ill-conditioned loss. Besides sample efficiency, we are also interested in robustness of the convergence under the presence of privacy noise. Because of the privacy noise, the private gradient descent will be unable to converge to where a non-private algorithm can reach. Specifically, when the samples are noisy or have noisy labels, the loss curvature may be sharp. The sharpness also implies lower smoothness, i.e., a small M or has a very small PL parameter. Thus, gradients may change tremendously at some steps especially in the presence of privacy noise. Such changes have more critical impact when only a less number of iterations can be executed due to the privacy constraint. Assume α is some constant while $\kappa \gg 1/\alpha$, we immediately get:

$$\text{ERUB}_{\min}^{\text{uniform}} = \Theta\left(\kappa \ln\left(1 + \frac{1}{\kappa\alpha}\right)\right) = \Theta\left(\frac{1}{\alpha}\right) \leq \mathcal{O}\left(\frac{MN^2R}{DG^2}\right),$$

$$\text{ERUB}_{\min}^{\text{dynamic}} = \Theta(1).$$

Both are robust, but the dynamic schedule has a smaller factor since $1/\alpha$ could be a large number. In addition, the factor implies that when more samples are used, the dynamic schedule is more robust.

2.4.3 Gradient Descent Methods with Momentum

Section 2.4.1 shows that the step noise has an exponentially increasing influence on the final loss, and therefore a decreasing noise magnitude improves the utility upper bound by a $\ln N$ factor. However, the proper schedule can be hard to find when the curvature information, e.g., κ, is absent. A parameterized method that less depends on the curvature information is preferred. On the other hand, long-term iterations will result in forgetting of

the initial iterations, since accumulated noise overwhelmed the propagated information from the beginning. This effect will reduce the efficiency of the recursive learning frameworks.

Alternative to GD, the momentum method can mitigate the two issues. It was originally proposed to stabilize the gradient estimation [115]. In this section, we show that momentum (agnostic about the curvature) can flatten the dynamic influence and improve the utility upper bound. Previously, **(author?)** used the momentum as an estimation of gradient mean, without discussions of convergence improvements. **(author?)** gave a bound for the Adam with DP. However, the derivation is based on gradient norm, which results in a looser bound (see Table 2.1).

The momentum method stabilizes gradients by moving average history coordinate values and thus greatly reduces the variance. The $\phi(m_t, g_t)$ can be rewritten as:

$$\begin{aligned} m_{t+1} &= \phi(m_t, g_t) = \frac{v_{t+1}}{1-\beta^t}, \\ v_{t+1} &= \beta v_t + (1-\beta) g_t = (1-\beta) \sum\nolimits_{i=1}^{t} \beta^{t-i} g_t, \ v_1 = 0, \end{aligned} \tag{2.13}$$

where $\beta \in [0, 1]$. Note v_{t+1} is a biased estimation of the gradient expectation while m_{t+1} is unbiased.

Theorem 5 (Convergence under PL condition)**.** *Suppose $f(\theta; x_i)$ is G-Lipschitz, and $f(\theta)$ is M-smooth and satisfies the Polyak-Lojasiewicz condition. Assume $\beta \neq \gamma$ and $\beta \in (0, 1)$. Let $\eta_t = \frac{\eta_0}{2M}$ and $\eta_0 \leq 8\left(\sqrt{1 + 64\beta\gamma(\gamma-\beta)^{-2}(1-\beta)^{-3}} + 1\right)^{-1}$. Then the following holds:*

$$\begin{aligned} \text{EER} \leq \big(\gamma^T + 2R\eta_0\alpha \underbrace{U_3(\sigma, T)}_{\textit{noise varinace}}\big)(f(\theta_1) - f(\theta^*)) \\ -\zeta \frac{\eta_0}{2M} \underbrace{\sum\nolimits_{t=1}^{T} \gamma^{T-t} \mathbb{E} \left\| v_{t+1} \right\|^2}_{\textit{momentum effect}} \end{aligned} \tag{2.14}$$

where $\gamma = 1 - \frac{\eta_0}{\kappa}$, $\zeta = 1 - \frac{1}{\beta(1-\beta)^3}\eta_0^2 - \frac{1}{4}\eta_0 \geq 0$, and $U_3 = \sum_{t=1}^{T} \gamma^{T-t} \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum_{i=1}^{t} \beta^{2(t-i)} \sigma_i^2$.

The upper bound includes three parts that influence the bound differently: *(1) Convergence.* The convergence term is mainly determined by η_0 and κ. η_0 should be in $(0, \kappa)$ such that

the upper bound can converge. A large η_0 will be preferred to speed up convergence if it does not make the other two terms worse. *(2) Noise Variance.* The second term compressed in U_3 is the effect of the averaged noise, $\sum_{i=1}^{t} \beta^{2(t-i)}\sigma_i^2$. One difference introduced by the momentum is the factor $(1-\beta)/(1-\beta^t)$ which is less than γ^t at the beginning and converges to a non-zero constant $1-\beta$. Therefore, in U_3, $\gamma^{T-t}(1-\beta)/(1-\beta^t)$ will be constantly less than γ^T meanwhile. Furthermore, when $t > \hat{T}$, the moving average $\sum_{i=1}^{t} \beta^{2(t-i)}\sigma_i^2$ smooths the influence of each σ_t. *(3) Momentum Effect.* The momentum effect term can improve the upper bound when η_0 is small. For example, when $\beta = 0.9$ and $\gamma = 0.99$, then $\eta_0 \leq 0.98/M$ which is a rational value. Following the analysis, when M is large which means the gradient norms will significantly fluctuate, the momentum term may take the lead. Adjusting the noise scale in this case may be less useful for improving utility.

To give an insight on the effect of dynamic schedule, we provide the following utility bounds.

Theorem 6 (Uniform schedule)**.** *Suppose the assumptions in Theorem 5 are satisfied. Let* $\sigma_t^2 = T/R$*, and let:* $\hat{T} = \max t \ s.t.\ \gamma^{t-1} \geq \frac{1-\beta}{1-\beta^t}$, $T = \left\lceil \mathcal{O}\left(\frac{\kappa}{\eta_0}\ln\left(1+\frac{\eta_0}{\kappa\alpha}\right)\right)\right\rceil$*. Given some positive constant* c *and* $\alpha_0 > 0$ *with* $1/\alpha > 1/\alpha_0$*, the following inequality holds:*

$$\mathrm{ERUB}_{\min} \leq \mathcal{O}\left(\frac{\kappa^2}{\kappa+\eta_0/\alpha}\left[\mathbb{I}_{T\leq\hat{T}} + \gamma^{\hat{T}-1}\ln\left(1+\frac{\eta_0}{\kappa\alpha}\right)\mathbb{I}_{T>\hat{T}}\right]\right).$$

Theorem 7 (Dynamic schedule)**.** *Suppose the assumptions in Theorem 5 are satisfied. Let* $\alpha' = \frac{2\eta_0\alpha}{\gamma(1-\gamma\beta^2)}$, $\beta < \gamma$ *and* $\hat{T} = \max t \ s.t.\ \gamma^{t-1} \geq \frac{1-\beta}{1-\beta^t}$*. Use the following schedule:* $\sigma_t^2 = \frac{1}{R}\sum_{i=1}^{T}\sqrt{\frac{q_i}{q_t}}$, $T^{dyn} = \left\lceil \mathcal{O}\left(\frac{2\kappa}{\eta_0}\ln\left(1+\frac{\eta_0}{\kappa\alpha}\right)\right)\right\rceil$*, where* $q_t = c_1\gamma^{T+t}\mathbb{I}_{T\leq\hat{T}} + \gamma^{\hat{T}-1}c_2\gamma^{T-t}\mathbb{I}_{T>\hat{T}}$ *for some positive constants* c_1 *and* c_2*. The following inequality holds:*

$$\mathrm{ERUB} \leq \gamma^T + 2\eta_0\alpha\sum_{t=1}^{T} Rq_t\sigma_t^2,$$
$$\mathrm{ERUB}_{\min} \leq \mathcal{O}\left(\frac{\kappa\alpha}{\kappa\alpha+\eta_0}\left(\frac{\kappa\alpha}{\kappa\alpha+\eta_0}\mathbb{I}_{T\leq\hat{T}} + \mathbb{I}_{T>\hat{T}}\right)\right).$$

Discussion. Theoretically, the dynamic schedule is more influential in vanilla gradient descent methods than the momentum variant. The result is mainly attributed to the averaging

operation. The moving averaging, $(1-\beta)\sum_{i=1}^{t}\beta^{t-i}g_i/(1-\beta^t)$, increase the influence of the under-presented initial steps and decrease the one of the over-sensitive last steps. Counterintuitively, the preferred dynamic schedule should be increasing since q_t decreases when $t \leq \hat{T}$.

2.4.4 Extension to Private Stochastic Gradient Descent

Though PGD provides a guarantee both for utility and privacy, computing gradients of the whole dataset is impractical for large-scale problems. For this sake, studying the convergence of Private Stochastic Gradient Descent (PSGD) is meaningful. The Algorithm 2.1 can be easily extended to PSGD by subsampling n gradients where the batch size $n \ll N$. According to [159], when privacy is measured by zCDP, there are two ways to account for the privacy cost of PSGD depending on the batch-sampling method: sub-sampling with or without replacement. In this paper, we focus on the random subsampling with replacement since it is widely used in deep learning in literature, e.g., [3, 42]. Accordingly, we replace N in the definition of α by n because the term is from the sensitivity of batch data (see Eq. (2.1)). For clarity, we assume that T is the number of iterations rather than epochs and that $\tilde{\nabla}_t$ is mean stochastic gradient.

When a batch of data are randomly sampled, the privacy cost of one iteration is cp^2/σ_t where c is some constant, $p = n/N$ is the sample rate, and $1/\sigma_t^2$ is the full-batch privacy cost. Details of the sub-sampling theorems are referred to the Theorem 3 of [159] and their empirical setting. Threfore, we can replace the privacy constraint $\sum_t p^2/\sigma_t^2 = R$ by $\sum_t 1/\sigma_t^2 = R'$ where $R' = R/p^2 = \frac{N^2}{n^2}R$. Remarkably, we omit the constant c because it will not affect the results regarding uniform or dynamic schedules. Notice N^2R in the α is replaced by $n^2R' = N^2R$. Thus, the form of α is not changed which provides convenience for the following derivations.

Now we study the utility bound of PSGD. To quantify the randomness of batch sampling, we define a random vector ξ_t with $\mathbb{E}[\xi_t] = 0$ and $\mathbb{E}\left\|\xi_t\right\|^2 \leq D$ such that $\tilde{\nabla}_t \leq \nabla_t + \sigma_g\xi_t/n$ for some positive constant σ_g. Because ξ_t has similar property to the privacy noise ν_t, we can

easily extend the PGD bounds to PSGD bounds by following theories.

Theorem 8 (Utility bounds of PSGD)**.** *Let α, κ and γ be defined in Eq.* (2.6)*, and $\eta_t = \frac{1}{M}$. Suppose $f(\theta; x_i)$ is G-Lipschitz and $f(\theta)$ is M-smooth satisfying the Polyak-Lojasiewicz condition. For PSGD, when batch size satisfies $n = \max\{N\sqrt{R}, 1\}$, the following holds:* $\text{ERUB} = \gamma^T + \alpha_g \sigma_g^2 + R' \sum_{t=1}^{T} q_t \sigma_t^2$, *where* $q_t \triangleq \gamma^{T-t}\alpha$, $\sum_t 1/\sigma_t^2 = R'$. *where* $\alpha_g = \frac{D}{2\mu N^2 R(f(\theta_1) - f(\theta^*))}$.

Theorem 9 (PSGD with momentum)**.** *Let $\alpha_g = \frac{D}{2\mu N^2 R(f(\theta_1) - f(\theta^*))}$. Suppose assumptions in Theorem 5 holds. When batch size satisfies $n = \max\{N\sqrt{R}, 1\}$, the $U_3(\sigma, T)$ has to be replaced by $\tilde{U}_3 = U_3^g + U_3$, with $\alpha R' U_3^g \leq \alpha_g \sigma_g^2$ when PSGD is used.*

See proof on page 104. As shown above, the utility bound of PSGD differs from the PGD merely by $\alpha_g \sigma_g^2$. Note $\alpha_g = \mathcal{O}(\frac{D}{N^2 R})$ which fits the order of dynamic-schedule bounds. In addition, α and other variables are not changed. Hence, the conclusions w.r.t. the dynamic/uniform schedules maintain the same.

2.4.5 Comaprison of generalization bounds

In addition to the empirical risk bounds in Table 2.1, in this section we study the *true risk bounds*, or generalization error bounds. True risk bounds characterize how well the learnt model can generalize to unseen samples subject to the inherent data distribution. By leveraging the generic learning-theory tools, we extend our results to the *True Excess Risk* (TER) for strongly convex functions as follows. For a model θ, its TER is defined as follows:

$$\text{TER} \triangleq \mathbb{E}_{x\sim\mathcal{X}}[\mathbb{E}[f(\theta; x)]] - \min_{\hat{\theta}} \mathbb{E}_{x\sim\mathcal{X}}[f(\hat{\theta}; x)],$$

where the second expectation is over the randomness of generating θ (e.g., the noise and stochastic batches). Assume a dataset d consist of N samples drawn i.i.d. from the distribution $\mathcal{X}$. Two approaches could be used to extend the empirical bounds to the true excess risk: One is proposed by [124] where the true excess risk of PGD can be bounded in high probability. For example, [11] achieved a $\frac{\ln^2 N}{N}$ bound with N^2 iterations. Alternatively, instead of relying on the probabilistic bound, **(author?)** [12] used the uniform stability to

Table 2.2: Comparison of true excess risk bounds. The algorithms are T-iteration $\frac{1}{2}R$-zCDP or equivalently (ϵ, δ)-DP under the μ-strongly-convex condition. The $\mathcal{O}$ notation in this table drops other ln terms. Assume loss functions are 1-smooth and 1-Lipschitz continuous, and all parameters satisfy their numeric assumptions. * marks the method with convex assumption.

Algorithm	Utility Upper Bd.	T
GD+Adv [11]	$\mathcal{O}_{1-p}\left(\frac{\sqrt{D}\ln^2 N \ln(1/p)}{p\mu N R_{\epsilon,\delta}}\right)$	$\mathcal{O}(N^2)$
SVRG+MA [139]	$\mathcal{O}\left(\frac{D\ln N}{\mu N^2 R_{\epsilon,\delta}}\right)$	$\mathcal{O}(\ln \frac{N^2 R_{\epsilon,\delta}}{D})$
SSGD+zCDP [42]	$\mathcal{O}\left(\left(\frac{1}{\sqrt{N}} + \frac{2\sqrt{D}}{\sqrt{R}N}\right)\ln N\right)$	$\mathcal{O}(\frac{N^2}{16D/R^2+4N})$
* SGD+MA [12]	$\mathcal{O}\left(\max\left\{\frac{\sqrt{D}}{N\sqrt{R_{\epsilon,\delta}}}, \frac{1}{\sqrt{N}}\right\}\right)$	$\mathcal{O}(\min\{\frac{N}{8}, \frac{N^2 R_{\epsilon,\delta}}{32D}\})$
True risk in high probability $(1-p)$		
GD+zCDP, Static Schedule	$\mathcal{O}_{1-p}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(N)\ln(1/p)}{NR}} + \frac{4}{p}\right)\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
GD+zCDP, Dynamic Schedule	$\mathcal{O}_{1-p}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(1/p)}{NR}} + \frac{4}{p}\right)\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
Momentum+zCDP, Static Sch.	$\mathcal{O}_{1-p}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(1/p)}{NR}(c + \ln N \mathbb{I}_{T>\hat{T}})} + \frac{4}{p}\right)\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
Momentum+zCDP, Dynamic Sch.	$\mathcal{O}_{1-p}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(1/p)}{NR}(1 + \frac{cD}{N^2 R}\mathbb{I}_{T>\hat{T}})} + \frac{4}{p}\right)\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
True risk by uniform stability		
GD, Non-Private	$\mathcal{O}\left(\frac{D}{N^2 R}\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
GD+zCDP, Static Schedule	$\mathcal{O}\left(\frac{D\ln N}{N^2 R}\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
GD+zCDP, Dynamic Schedule	$\mathcal{O}\left(\frac{D}{N^2 R}\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
Momentum+zCDP, Static Sch.	$\mathcal{O}\left(\frac{D}{N^2 R}(c + \ln N \mathbb{I}_{T>\hat{T}})\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$
Momentum+zCDP, Dynamic Sch.	$\mathcal{O}\left(\frac{D}{N^2 R}(1 + \frac{cD}{N^2 R}\mathbb{I}_{T>\hat{T}})\right)$	$\mathcal{O}(\ln \frac{N^2 R}{D})$

give a tighter bound. Later, **(author?)** [42] improve the efficiency of gradient computation to achieve a similar bound. Both approaches introduce an additive term to the empirical bounds. In this section, we adopt both approaches to investigate the two types of resulting true risk bounds.

(1) True Risk in High Probability. First, we consider the high-probability true risk bound. Based on Section 5.4 from [124] (restated in Theorem 10), we can relate the EER to the TER.

Theorem 10. *Let $f(\theta; x)$ be G-Lipschitz, and $f(\theta)$ be μ-strong convex loss function given any $x \in \mathcal{X}$. With probability at least $1-p$ over the randomness of sampling the data set d,*

the following inequality holds:

$$\mathrm{TER}(\theta) \leq \sqrt{\frac{2G^2}{\mu N}}\sqrt{f(\theta) - f(\theta^*)} + \frac{4G^2}{p\mu N}, \tag{2.15}$$

where $\theta^* = \arg\min_\theta f(\theta)$.

To apply the Eq. (2.15), we need to extend EER, the expectation bound, to a high-probability bound. Following [11] (Section D), we repeat the PGD with privacy budget R/k for k times. Note, the output of all repetitions is still of R budget. When $k = 1$, let the EER of the algorithm be denoted as $F(R)$. Then the EER of one execution of the k repetitions is $F(R/k)$ where privacy is accounted by zCDP. When $k = \log_2(1/p)$ for $p \in [0, 1]$, by Markov's inequality, there exists one repetition whose EER is $F(R/\log_2(1/p))$ with probability at least $1 - 1/2^k = 1 - p$. Combined with Eq. (2.15), we use the bounds of uniform schedule and dynamic schedules in Section 2.4.2 to obtain:

$$\mathrm{TER}^{\mathrm{uniform}} \leq \tilde{\mathcal{O}}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(N)\ln(1/p)}{NR}} + \frac{4}{p}\right)\right), \tag{2.16}$$

$$\mathrm{TER}^{\mathrm{dynamic}} \leq \tilde{\mathcal{O}}\left(\frac{G^2}{\mu N}\left(\sqrt{\frac{D\ln(1/p)}{NR}} + \frac{4}{p}\right)\right), \tag{2.17}$$

where we again ignore the κ and other constants. Similarly, we can extend the momentum methods.

(2) True Risk by Uniform Stability. Following **(author?)** [12], we use the uniform stability (defined in Definition 6) to extend the empirical bounds. We restate the related definition and theorems as follows.

Definition 6 (Uniform stability)**.** *Let* $s > 0$. *A randomized algorithm* $\mathcal{M} : \mathcal{D}^N \to \Theta$ *is* s*-uniformly stable w.r.t. the loss function* f *if for any neighbor datasets* d *and* d'*, we have:*

$$\sup_{x\in\mathcal{X}} \mathbb{E}[f(\mathcal{M}(d); x) - f(\mathcal{M}(d'); x)] \leq s,$$

where the expectation is over the internal randomness of $\mathcal{M}$.

Theorem 11 (See, e.g., [123]). *Suppose $\mathcal{M} : \mathcal{D}^N \to \Theta$ is a s-uniformly stable algorithm w.r.t. the loss function f. Let $\mathcal{D}$ be any distribution over data space and let $d \sim \mathcal{D}^N$. The following holds true.*

$$\mathbb{E}_{d\sim\mathcal{D}^N}[\mathbb{E}[f(\mathcal{M}(d);\mathcal{D}) - f(\mathcal{M}(d);d)]] \leq s,$$

where the second expectation is over the internal randomness of $\mathcal{M}$. $f(\mathcal{M}(d);\mathcal{D})$ and $f(\mathcal{M}(d);d)$ represent the true loss and the empirical loss, respectively.

Theorem 12 (Uniform stability of PGD from [12]). *Suppose $\eta < 2/M$ for M smooth, G-Lipschitz $f(\theta;x)$. Then PGD is s-uniformly stable with $s = G^2T\eta/N$.*

Combining Theorems 11 and 12, we obtain the following:

$$\text{TER} \leq \text{EER} + G^2\frac{\eta T}{N}.$$

Because EER in this paper compresses a γ^T or similar exponential terms, unlike [12], we cannot directly minimize the TER upper bound w.r.t. T and η in the presence of a polynomial form of γ^T and T. Therefore, we still use $T = \mathcal{O}(\ln\frac{N^2R}{D})$ and η for minimizing EER. Note that

$$G^2\frac{\eta T}{N} \leq \mathcal{O}(\frac{G^2}{MN}\ln\frac{N^2R}{D}) \leq \mathcal{O}\left(\frac{G^2}{M}\right)$$

where we assume $N \gg D$ and use $\ln N \leq N$. Because the term $\mathcal{O}\left(G^2/M\right)$ is constant and independent from dimension, we follow [12] to drop the term when comparing the bounds. After dropping the additive term, it is obvious to see that the advantage of dynamic schedules still maintains since TER $\leq$ EER. A similar extension can be derived for [139].

We summarize the results and compare them to prior works in Table 2.2 where we include an additional method: Snowball Stochastic Gradient Descent (SSGD). SSGD dynamically schedule the batch size to achieve an optimal convergence rate in linear time.

Discussion. By using uniform stability, we successfully transfer the advantage of our dynamic schedules from empirical bounds to true risk bounds. The inherent reason is that our

bounds only need $\ln N$ iterations to reach the preferred final loss. With uniform stability, the logarithmic T reduce the gap caused by transferring. Compared to the [42, 12], our method has remarkably improved efficiency in T from N or N^2 to $\ln(N)$. That implies fewer iterations are required for converging to the same generalization error.

2.5 Experiments

We empirically validate the properties of privacy schedules and their connections to learning algorithms. In this section, we briefly review the schedule behavior on quadratic losses under varying data sensitivity.

Dataset. We create a subset of the MNIST dataset [74] including 1000 handwritten images of 10 digits (MNIST). We also construct a subset of the MNIST dataset with digit 3 and 5 only, denoted as MNIST35. Compared to the original dataset ($70,000$ samples), the small set will be more vulnerable to attack and the private learning will require larger noise (see the $1/N$ factor in Eq. (2.1)). Following the preprocessing in [3], we project the vectorized images into a 60-dimensional subspace extracted by PCA.

Setup. The samples are first normalized so that $\sum_{n=1}^{N} x_n = 0$ and the standard deviation is 1. Then the sample norms are scaled such that $\max_n \|x_n\| = 10$ (i.e., *data scales*). Upon the scaled data, we train a 2-layer Deep Neural Network (DNN) with 1000 hidden units by logistic regression. We fix the learning rate to 0.1 based on the corresponding experiments of non-private training (same setting without noise). The total privacy budget is $(4, 10^{-8})$-DP, equal to 0.1963-zCDP, which implies $R = 0.3927$. To control the sensitivity of the gradients, we clip gradients by a clipping norm fixed at 4. Formally, we scale down the sample gradients to length 4 if its norm is larger than 4. Because the schedule highly depends on the iteration number T, we grid search the best T in range $[50, 150]$ for compared methods. Therefore, we ignore the privacy cost of such tuning in our experiments which protocol is also used in previous work [3, 148]. All the experiments are repeated 100 times and metrics are averaged afterwards.

We first show the estimated influence of step noise q_t (by retraining the private learn-

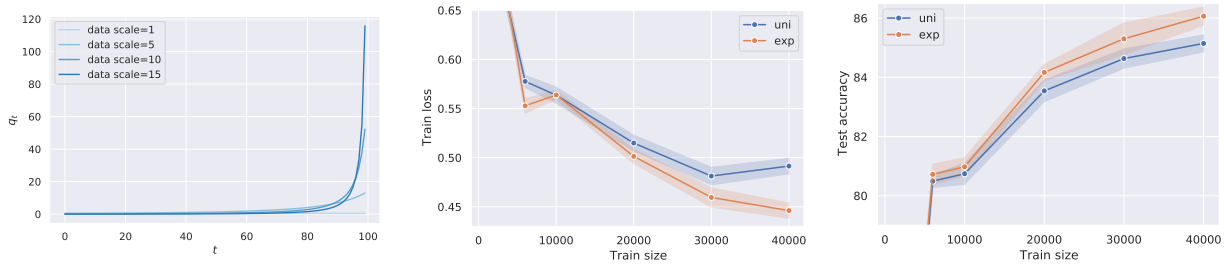


Figure 2.1: Comparison of dynamic schedule and uniform schedule on different data scale. Left pane is the influence by iteration estimated by retraining. The rest two panes are performance of DNN trained on the MNIST35 dataset with a varying total number of training samples, when the exponential influence is estimated on a randomly-generated auxiliary dataset.

ing algorithms) in Fig. 2.1 Left. We see the trends of influence are approximately in an exponential form of t. By Eq. (2.10), the resultant schedule on noise scale σ_t will be a normalized exponential decay. This observation motivates the use of exponential decay schedule in practice.

To estimate the influence without extra privacy costs, we use an auxiliary set, which is randomly sampled from Gaussian distribution, to pick the proper influence curvature parameterized by an exponential function. We use auxiliary synthesized datasets of the same size as the corresponding private datasets to tune the parameters. We vary the size of training data to examine the data efficiency of the dynamic schedule denoted as `exp`. For a fair comparison, we also choose the hyper-parameters of uniform schedule (`uni`) on the same auxiliary dataset. We show that as the training size increases, `exp` outperforms `uni` both on the training loss and the test accuracy. The result verifies our theoretic conclusion: dynamic schedule is more data efficient than the static schedule.

CHAPTER 3

FEDERATED ADVERSARIAL DEBIASING FOR TRANSFERABLE AND FAIR REPRESENTATIONS

Federated learning is a distributed learning framework that is communication efficient and provides protection over participating users' raw training data. One outstanding challenge of federate learning comes from the users' heterogeneity, and learning from such data may yield biased and unfair models for minority groups. While adversarial learning is commonly used in centralized learning for mitigating bias, there are significant barriers when extending it to the federated framework. In this work, we study these barriers and address them by proposing a novel approach Federated Adversarial DEbiasing (FADE). FADE does not require users' sensitive group information for debiasing and offers users the freedom to opt-out from the adversarial component when privacy or computational costs become a concern. We show that ideally, FADE can attain the same global optimality as the one by the centralized algorithm. We then analyze when its convergence may fail in practice and propose a simple yet effective method to address the problem. Finally, we demonstrate the effectiveness of the proposed framework through extensive empirical studies, including the problem settings of unsupervised domain adaptation and fair learning.

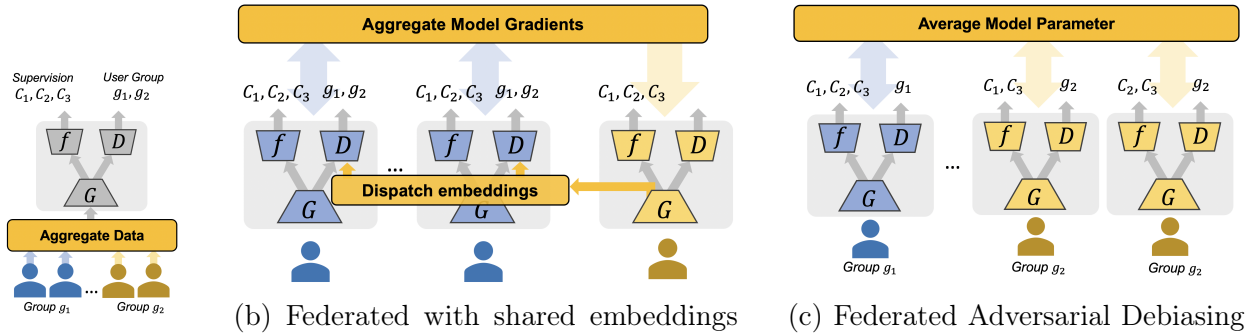
3.1 Introduction

The last decade witnessed the surging adoption of personal devices such as smartphones, smartwatches, and smart personal assistants. These devices directly interface with the users, collect personal data, conduct light-weighted computations, and use machine learning models to offer personalized services. The challenges from privacy concerns of sensitive personal data, limited computational resources, performance issues of localized learning all together lead to the federated learning (FL) paradigm [100, 16]. FedAvg [96], for example, provides an efficient and privacy-aware FL framework. Users train models locally, upload them to a central server iteratively aggregated to form a global model. FL greatly alleviated privacy concerns because the server can only access model parameters from the users instead of the

raw data used for training.

One major challenge of FL comes from the user heterogeneity where users provide statistically different data for training local models [29, 41]. Such heterogeneity may come from different sources. For example, the users may collect data under various conditions according to preferential or usages differences. Consider the learning of handwashing behavior from accelerometers of smartwatches, where patterns can drastically change when using different basins worldwide. Such domain shift [66] can lead to negative impacts during knowledge transfer among users [106]. Another common source of heterogeneity comes from the sensitive group information such as age, gender, and social groups, which are variables typically not to be identified during learning. Heterogeneity from this source is often associated with critical fairness issues [36] after deploying the models, where groups with less resource or smaller computation capability may be biased or even ignored during the learning [93], and the resulting global model may perform worse in minority groups.

Adversarial learning [47] has been a powerful approach to mitigate bias in centralized learning, in which an adversarial objective minimizes the information extracted by an *encoder* that can be maximally recovered by a parameterized model, *discriminator*. For example, it has been applied to disentangle task-specific features that may cause negative transfer [88], to perform unsupervised domain adaptation [46, 133], and recently to achieve fair learning [161]. However, there are significant barriers when applying adversarial techniques in FL: 1) Most existing approaches follow a top-down principle. In the context of FL, the adversarial objective requires the server to access the sensitive group variable (e.g., gender) to construct an adversarial loss. This requirement directly violates the privacy consideration design for FL, and users may not want to disclose their sensitive group variables. 2) adversarial learning demands extra information from users for training the adversarial component and imposes an additional computational burden on smart devices that may not be able to afford. 3) besides, it remains unknown how the introduction of an adversarial component would impact the distributed learning behavior (e.g., convergence property) of FL.



(a) Central

(b) Federated with shared embeddings [111]

(c) Federated Adversarial Debiasing (proposed)

Figure 3.1: Illustrations of different adversarial learning frameworks for debiasing. f, D and G are classifier (task model), discriminator and encoder, respectively. C_1, C_2, C_3 represents the task supervisions, for example, ground-truth classes, in the corresponding users. g_1 and g_2 represents the two groups of users. The encoders are adversarially trained such that the embeddings are informative for distinguishing C_1, C_2, C_3 but not g_1, g_2. The proposed FADE tackles a more challenging problem than other two because of isolated and non-sharing group/user data (or embeddings) and class-wise non-iid users within groups.

To address the challenges mentioned above, we propose a novel adversarial framework for debiasing federated learning following a bottom-up principle, called *Federated Adversarial DEbiasing (FADE)*. Besides the benefits from typical FL on communication efficiency and data privacy, FADE aims to achieve the following goals:

- **Privacy-Protecting**: The learning algorithm conforms to the privacy design of FL and does not require users' group variable to achieve debiasing w.r.t. the group variable.
- **Autonomous**: A user can choose to join and opt-out from the adversarial component anytime (e.g., due to computational budget or privacy budget) while still participate in the regular federated learning.
- **Satisfiable**: Under above restrictions, the distributed learning should output a debiased and accurate model, despite the user heterogeneity and unpredictable user participation.

To achieve these goals, we first propose a generic algorithm for FADE and show that ideally, it can attain the same global optimality as the one by the central algorithm. We then show how its convergence may fail in practice and propose a simple yet effective method to address

the problem. Finally, we demonstrate the effectiveness of the proposed framework through extensive empirical studies on various applications.

3.2 Related Work

Federated Learning (FL) [96] is a distributed learning framework that allows users with different capabilities to collaboratively train a model without sharing their own data. A critical challenge in FL is the heterogeneity among users. Viewing the learning process of FL as knowledge transfer among different users, heterogeneity in user data leads to negative transfer between users and compromises generalization [13]. One idea to alleviate the negative effect from the heterogeneity during the training, is to find the consensus among users. For example, in [49, 85, 77, 43], the consensus on task knowledge is achieved by distillation. In this work, we seek an alternative and efficient approach by adversarial debiasing the users of different groups.

Adversarial Learning has been widely applied in various domains, such as neural language recognition [88], image-to-image (dense) prediction [95], image generation [47], and etc. Conceptually, adversarial learning aims to solve a two-player (or multi-player) game between two adversarial objectives, which typically leads to a min-max optimization problem. Existing approaches can be briefly categorized as: *1) Sample-to-Sample (S2S)* adversarial learning, where the adversarial objective quantifies the difference between synthetic and real samples. Examples include adversarial learning against adversarial attacks [94] and generative adversarial networks [47]. *2) Group-to-Group (G2G)* adversarial learning, which aims to reduce the max discrepancy (bias) between group distributions, for example, adversarial domain adaptation [46], adversarial fairness [161] and adversarial multi-task learning [88]. All these variants assume the availability of adversarial groups in the same computation node, e.g., by aggregating data in Fig. 3.1a, and thus cannot be directly extended to federated learning to the violation of privacy design (requiring access of the sensitive group information). A recent effort is done by [111] where embeddings of different groups are shared (see Fig. 3.1b). Nevertheless, both sharing data and embeddings could induce additional privacy risk and

communication costs. The proposed FADE eliminated these requirements, leading to private and efficient distributed collaboration between users/groups.

3.3 Federated Adversarial Debiasing

In this section, we first formulate the proposed Federated Adversarial Debiasing (FADE) framework. We work on the standard federated learning problem setting which learns one model from a set of distributed participating users. Users conduct local learning based on their own data and send the parameters of learning models to a server periodically. The server aggregates the local models to form a global model. We assume the users have non-iid data and each user belongs to one of the E user groups as indicated by a group variable (e.g., age, gender, race) that is not to be shared outside of the local learning.

The model of each user consists of three components: a decoder f for the learning task (e.g., classification target), an encoder G, and a group discriminator D, as illustrated in Fig. 3.1c. In the two-group setting (a data point belongs to either group 0 and 1), D outputs a scalar in $(0, 1)$ approximating the probability of an input data point x belong to the group 0. More generally, for E groups, we use a softmax mapping in the last layer of D which outputs an E-dimensional vector. The FADE objective learns f, D, G by:

$$\min_{f,G} \mathcal{L}(f, G) = \sum\nolimits_{g=1}^{E} \sum\nolimits_{i=1}^{m_g} L_{i,g}(f, G), \tag{3.1}$$

$$L_{i,g}(f, G) = L_i^{\text{task}}(f, G) + \lambda \max_{D} L_{i,g}^{\text{adv}}(G, D), \tag{3.2}$$

where $L_i^{\text{task}}(f, G)$ is the task loss for the i-th user, $L_{i,g}^{\text{adv}}(G, D)$ is the adversarial loss, and m_g is the number of users in group g. Note that we absorb the variable model D into $L_{i,g}$ in Eq. (3.2), and the objective is still an optimization over f, D, G. For classification tasks, the task loss can be defined as $L_i^{\text{task}}(f, G) \triangleq \mathbb{E}_{(x,y)\sim p_i(x,y)}[\mathcal{E}(f(G(x)), y)]$, where $\mathcal{E}$ denotes the cross-entropy loss and p_i is the data distribution of user i. The adversarial loss is defined as $L_{i,g}^{\text{adv}}(G, D) \triangleq \mathbb{E}_{x\sim p_i(x)}[\log D_g(G(x))]$, where $D_g(G(x))$ is the g-th output of the softmax vector. The optimal solution for the min-max problem is the *adversarial balance* when D is unable to tell the difference of $G(x)$ among groups. For the two-group case, the adversarial

loss can be modified as:

$$L_{i,g}^{\text{adv}}(G, D) = \mathbb{E}_{x \sim p_i(x)} \big[\mathbb{I}(g = 0) \log D(G(x)) + \mathbb{I}(g = 1) \log(1 - D(G(x))) \big], \tag{3.3}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

One fundamental difference between traditional adversarial learning and FADE is that FADE only has one group data in the loss function. Hence, users have no sense of what an adversary (a user from other groups) looks like. Directly optimizing this objective may fail in finding the right direction towards convergence. In the worst case, the optimal solution may not be the adversarial balance. In the next section, we will provide principled analysis to the adversarial balance that is achievable under appropriate conditions.

We summarize the server and user update strategies in Algorithms 3.1 and 3.2. The server is responsible for aggregating users' models and dispatching the global models to users. Meanwhile, users train the received global model and the adversarial component using local data. Note that we use the reversal gradient strategy to implement the min-max optimization in Algorithm 3.1. Our algorithm enjoys the two nice properties:

Autonomous: Different from vanilla FL, FADE allows the users to decide whether or not to join the learning of the discriminator D at each iteration. A user can opt-in the discriminator learning at a low frequency or completely opt-out when privacy becomes a concern or learn with restrictive computational resources. For example, in the adversarial domain adaptation setting [111] where some users have supervision and some others not, some supervised user may not want to help unsupervised users. FADE will significantly reduce the communication cost and privacy risk overhead involved by cutting down the interactions form these users.

Privacy: In the proposed FADE framework, the group label g will be restricted to local learning and the group debiasing is done through the discriminator model D. Thus, users will not be able to obtain the other users' sensitive attributes including the group variable. Moreover, following [98], the privacy of FADE can be strictly protected by directly injecting *Differential-Privacy* noise during the gradient descent procedure.

Algorithm 3.1: FADE User Update

Require: f, G, D received from server, learning rate η, adversarial parameter λ, user data distribution p_i

$f_0, D_0, G_0 = f, D, G$
for $t = 1, \ldots, K$ **do**
  Sample a batch by $x \sim p_i(x)$ or $(x, y) \sim p_i(x, y)$
  $z \leftarrow G(x)$
  $\nabla_f \leftarrow \frac{\partial L_i^{\text{task}}}{\partial f}$, $\nabla_D \leftarrow \frac{\partial L_i^{\text{adv}}}{\partial D}$
  if adversarial game D is accepted by user i **then**
    $\nabla_G \leftarrow \frac{\partial z}{\partial G}\left(\frac{\partial L_i^{\text{task}}}{\partial z} + \lambda \frac{\partial L_i^{\text{adv}}}{\partial z}\right)$
    $D_{t+1} \leftarrow D_t + \eta \nabla_D$
    $G_{t+1} \leftarrow G_t - \eta \nabla_G$
  else
    $\nabla_G \leftarrow \frac{\partial z}{\partial G} \frac{\partial L_i^{\text{task}}}{\partial z}$
    $D_{t+1} \leftarrow D_t$
    $G_{t+1} \leftarrow G_t - \eta \nabla_G$
  $f_{t+1} \leftarrow f_t - \eta \nabla_f$
return f_{K+1}, G_{K+1}, D_{K+1}

Algorithm 3.2: FADE Server Aggregation

Require: Initial models f, D, G, momentum parameter β

for $t \in 1, \cdots, T_{\max}$ **do**
  Select m active users uniformly at random into $\mathcal{A}$
  Broadcast $\theta_t = (f_t, G_t, D_t)$ to m users
  for user i in $\mathcal{A}$ in parallel **do**
    User updates by Algorithm 3.1
  Aggregate $\{\theta_t^k = (f_t^i, G_t^i, D_t^i)\}_{i=1}^m$ and average

$$\theta_{t+1} \leftarrow \beta \sum\nolimits_{i=1}^{m} \frac{n_i}{N} \theta_t^i + (1 - \beta)\theta_t$$

return f_t, G_t, D_t

3.4 Optimality Analysis

Despite the fact that FADE enables autonomous and improves privacy in learning, it is critical to ask if the algorithm gives a satisfiable solution and what is the optimal solution of Eq. (3.1). Remarkably, FADE differs from traditional adversarial learning by Eq. (3.3), where only one group is used to evaluate the adversarial objective. This imposes a unique challenge in learning as it may compromise the convergence of learning. Below we give formal analysis of the optimality when Algorithm 3.2 is iterated with users from two groups

in non-zero probability. Since most of multi-group adversarial problems can be transformed into two-group problems, we focus on discussing the two-group case for the ease of analysis.

Consider the case when each group only has one user. The data distributions for the two users are p_1 and p_2, respectively. We single out the min-max optimization in Eq. (3.1) as:

$$\min_G \max_D \mathbb{E}_{p_1}[\log D(G(x))] + \mathbb{E}_{p_2}[\log(1 - D(G(x)))].$$

For simplicity, we denote $G(x)$ by z and slightly abuse $p_1(x)$ by $p_1(z)$ in our discussion. Hence, we can define:

$$\mathbf{D}_{p_1,p_2} = \max_D \mathbb{E}_{p_1}[\log D(z)] + \mathbb{E}_{p_2}[\log(1 - D(z))],$$

which is the maximal discrepancy between $p_1(z)$ and $p_2(z)$ that D can characterize. Now, we can rewrite the min-max problem as $\min_G \mathbf{D}_{p_1,p_2}(G)$ which minimizes the distribution distance over z. Alternatively, we can formulate it by $\min_{p_1,p_2} \mathbf{D}_{p_1,p_2}$ since p_1 and p_2 are parameterized by G.

Because users may participate federated learning at varying frequencies, we use an auxiliary random variable $\xi_i \in \{0, 1\}$ for $i = 0, 1$ to denote whether the user is active for training. We assume ξ_i is subject to the Bernoulli distribution, $B(1, \alpha_i)$. Plug ξ_i into $\mathbf{D}_{p_1,p_2}$ to obtain $\mathbf{D}_{p_1,p_2} = \max_D \mathbb{E}_{p_1}[\xi_1 \log D(z)] + \mathbb{E}_{p_2}[\xi_2 \log(1 - D(z))]$ and take expectation:

$$\begin{aligned}\tilde{\mathbf{D}}_{p_1,p_2} &\triangleq \mathbb{E}_{\xi_1,\xi_2}[\mathbf{D}_{p_1,p_2}] \\ &= \max_D \mathbb{E}_{p_1}[\alpha_1 \log D(z)] + \mathbb{E}_{p_2}[\alpha_2 \log(1 - D(z))]. \end{aligned} \tag{3.4}$$

Therefore, our problem is transformed as minimizing $\tilde{\mathbf{D}}_{p_1,p_2}$.

Note that with p_1 and p_2 given, the solution of the maximization in $\tilde{\mathbf{D}}_{p_1,p_2}$ is:

$$D^*_{\alpha_1,\alpha_2}(z) = \frac{\alpha_1 p_1(z)}{\alpha_1 p_1(z) + \alpha_2 p_2(z)}, \tag{3.5}$$

with which we can derive the optimality sufficiency as below.

Theorem 13. *The condition $p_1(z) = p_2(z)$ is a sufficient condition for minimizing $\tilde{\mathbf{D}}_{p_1,p_2}$ and the minimal value is $\alpha_1 \log \alpha_1 + \alpha_2 \log \alpha_2 + (\alpha_1 + \alpha_2) \log(\alpha_1 + \alpha_2)$.*

Theorem 13 shows that even if some users are inactive, the distribution matching, $p_1 = p_2$, remains a sufficient optimality condition. We remark that the above result can be generalized to multiple users when all users are iid and ξ_i represent the ratio of group i in users. In addition, we notice Theorem 13 does not guarantee a stable convergence or exclude other undesired solutions. We discuss these issues in the following.

3.4.1 The effect of imbalanced groups

Although Theorem 13 shows the optimality of the matched distribution, the optimization may still fail to converge especially when one group of users are relatively inactive, e.g., $\alpha_1 \ll \alpha_2$. When $\alpha_1 \ll \alpha_2$ or reverse, we call the situation as *imbalanced groups*. The imbalanced groups happens because the users are free to quit or joint the training. From Eq. (3.5), we observe that $D^*(x)$ will be less sensitive to changes of $p_1(x)$ if $\alpha_1 \ll \alpha_2$, and vice versa. Meanwhile, $\log D^*(x) \to -\infty$ and $\mathbf{D}_{p_1,p_2}$ approaches the minimum even if p_1 and p_2 are quite different.

Theorem 14. *Let ϵ be a positive constant. Suppose $|\log p_1(x) - \log p_2(x)| \leq \epsilon$ for any x in the support of p_1 and p_2. Then we have $\tilde{\mathbf{D}}_{p_1,p_2} = \mathcal{O}(\alpha_1\epsilon/(\alpha_1 + \alpha_2))$ when $\alpha_1 \ll \alpha_2$.*

Theorem 14 reveals that the imbalance between groups could greatly reduce the sensitivity of the discrepancy ϵ between p_1 and p_2. A less sensitive discriminator will ignore the minor differences between groups. The importance of discrepancy sensitivity for the adversarial convergence was also discussed in [8]. It is easy to see the negative impact of the low sensitivity: **1)** higher communication cost incurs due to more communication rounds are required to check the discrepancy; **2)** the optimization possibly fails to converge due to vanished gradients (scaled by α_1).

3.4.2 Squared adversarial loss

In Eq. (3.4), when $\alpha_1 \to 0$ and $\alpha_2 \to 1$, we notice that $\tilde{\mathbf{D}}_{p_1,p_2}$ approaches 0 while $\mathbb{E}_{p_1}[\log D(z)] \to -\infty$. In other words, the large value of $\mathbb{E}_{p_1}[\log D(z)]$ is neglected due to its coefficient α_1. To re-emphasize the value, a heuristic method is to increase the weight when

$|\mathbb{E}_{p_1}[\log D(z)]|$ is large. Thus, we propose to replace $L_{i,g}^{\text{adv}}(G, D)$ by:

$$L_{i,g,2}^{\text{adv}}(D, G) = -\frac{1}{2}\left(L_{i,g}^{\text{adv}}(G, D)\right)^2, \tag{3.6}$$

which we call *squared adversarial loss*. We can write the corresponding discrepancy $\tilde{\mathbf{D}}_{p_1,p_2}^{(2)}$ as:

$$\min_D \alpha_1 \mathbb{E}_{p_1}^2[\log D(z)] + \alpha_2 \mathbb{E}_{p_2}^2[\log(1 - D(z))].$$

Though we derive the squared adversarial loss in a heuristic manner, the loss can be explained in the view of resource-fair federated learning [79]. Because the adversarial objective pays more attention to the frequent group, we can interpret the problem as the unfairness between groups. Following [79], we generalize our adversarial loss function as:

$$L_{i,g,2}^{\text{adv}}(D, G) \triangleq (-1)^{q-1}\frac{1}{q}\mathbb{E}_x\left[\ell_k^q(D, G; x)\right], \tag{3.7}$$

where $q \geq 1$. If $q = 1$, the loss degrades to the vanilla one.

3.4.3 The effect of non-iid users

It is well-known that typical federated learning approaches suffer from very heterogeneous users since they sample data from very different distributions. The adversarial objective captured and decreases the group heterogeneity by design. Another kind of heterogeneity is related to the users' tasks. We argue that the heterogeneity is natural and could be essential for the task discriminability but may be accidentally eliminated by adversarial learning. For example, three users are non-iid by three classes. After FADE training, the non-iid users collapse to the similar distributions due to the wrong sense of the group discrepancy.

To prove the existence of user-collapsed solution for FADE, we consider $z \sim p(z|T = t)$, or simply $z \sim p(z|t)$, where t is a discrete hidden variable related to users' tasks. For example, each user has one class of samples in classification tasks. Then t is the corresponding class. In addition, we define $\hat{p}_1(z) = \frac{1}{m}\sum_{t=1}^m p(z|t)$ which is a p.d.f. For simplicity, we assume all users always participate the learning, i.e., $\alpha_i = 1$ for all users. Hence, we can obtain $\mathbf{D}_{p_1,p_2}$

as

$$\max_D \sum_{t=1}^{m} \mathbb{E}_{p(z|t)}[\log D(z)] + \mathbb{E}_{p_2}[\log(1 - D(z))]$$
$$= \max_D m\mathbb{E}_{\hat{p}_1(z)}[\log D(z)] + \mathbb{E}_{p_2}[\log(1 - D(z))],$$

whose maximizer is given by: $D^*(z) = \frac{m\hat{p}_1(z)}{m\hat{p}_1(z)+p_2(z)}$. Use similar derivations as in Theorem 13, we can show that $\hat{p}_1(z) = p_2(z)$ is a sufficient optimality condition, which implies:

$$\sum_{t=1}^{m} p(z|t) = mp_2(z). \tag{3.8}$$

First, we can still obtain $p_1(z)\sum_{t=1}^{m} p(t|z)/p(t) = mp_2(z)$ from Eq. (3.8) where we use $p(z|t) = p_1(z)\frac{p(t|z)}{p(t)}$. If $\sum_{t=1}^{m} \frac{p(t|z)}{p(t)} = m$, then we can get the vanilla solution, $p_1(z) = p_2(z)$.

Except for the vanilla solution, a trivial solution to Eq. (3.8) is $p(z|t) = p_2(z)$. However, the solution could hurt the task utility since it may eliminate the inherent difference between tasks. For instance, if t represents the classification label, the solution will vanish the discriminability of the representation z. We call the scenario as the *user collapse*. It worth noticing that user collapse could happen even if the p_1 and p_2 are matched.

3.4.4 Mitigate user collapse

Since there are arbitrarily many solutions to $\sum_{t=1}^{m} \frac{p(t|z)}{p(t)} = m$, we need to constraint the feasible solutions such that the collapsed solution will be eliminated. Notice $\frac{p(t|z)}{p(t)} = \frac{p(t,z)}{p(z)p(t)}$ is related to the mutual information between t and z. Conceptually, we can modify the adversarial loss to:

$$\hat{L}^{\text{adv}}_{i,g,2}(D, G) = L^{\text{adv}}_{i,g,2}(D, G) + I(G(x); t|i),$$

where $I(G(x); t|i)$ is the mutual information conditioned on user i. Because mutual information is hard to estimate in practice (especially given few samples), we provide some surrogate solutions.

If the t represents the class labels and supervision is available, then $I(G(x); t|i)$ is already encouraged by L^{task}. If supervision is not available, we may maximize the entropy of the

output of classifier f such that the correlation between user's tasks and representations will not disappear during training. Useful techniques were previously exploited for unsupervised domain adaptation, e.g., [92], and we defer the technique details to Section 3.5.2.

3.4.5 Privacy risks from malicious FADE users

Our analysis suggests the feasibility of using adversarial training in the federated setting. The distribution matching is achievable under variety of cases including imbalanced groups, although the success rate may vary. But such power also implies potential privacy overhead associated with FADE. Consider a malicious user i who wants to steal data from others, FADE can match $p_i(x)$ with a victim's data $p_j(x)$. The empirical study in [53] also discussed the risk where a malicious attacker may take advantage of the discriminator to steal other users' data. Our results in Theorem 13 theoretically show that the attack is possible in general. During the learning of the adversarial discriminator, injecting predefined noise is known to be effective to defend such attacks [131]. Meanwhile, users could quit or frequently opt-out the federated communication when the privacy budget (quantified by noise and Differential Privacy metric [38]) is low. Based on Theorem 14, when more and more users opt-out the communication, the adversary's discriminator can hardly sense one victim's distribution.

3.5 Experiments on Unsupervised Domain Adaptation

In this section, we evaluate the FADE algorithms on Unsupervised Domain Adaptation (UDA) [111, 82, 43]. UDA aims to mitigate the domain shift between supervised and unsupervised data such that the trained classifiers can generalize to unlabeled data. We call the supervised user (domain) as the source user (domain). Each domain may include multiple users.

Related work. [111] is among the first to discuss the adversarial UDA under federated constraint, through sharing the embedding of samples. However, we consider a more challenging problem, a federated adversarial learning *without* sharing data. Recently, learning

without access to the source data has gained increasing attention. [82] (SHOT) considered the domain adaptation only using the source-domain model which surprisingly outperformed most traditional UDA with source supervisions. However, its success relies on the pre-matched representation distribution (but not well discriminated) by batch normalization (BN) layers. In the FADE setting, the BN layers will fail to match representations since the local estimate of their mean will be easily biased. In addition, in [43], distillation is used to avoid directly passing data. Differing from [43], FADE is more efficient since it does not need to upload all models from source domain to target domain. For example, if M_s users (M_t) in source (target) domain take part in training, sending models will involves $M_s M_t$ communication. Instead, FADE only use $M_s + M_t$ times to communicate between domains.

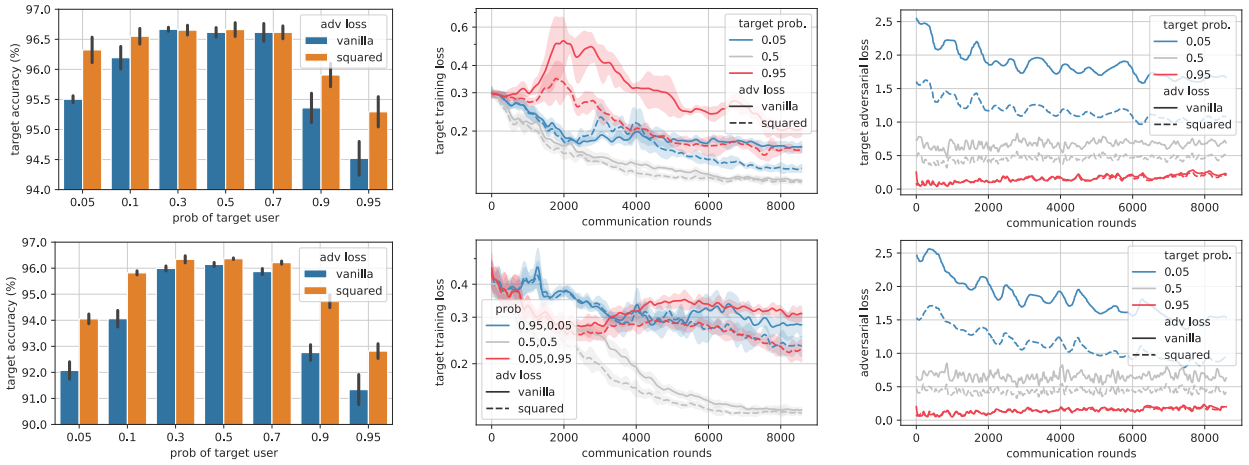


Figure 3.2: Comparison of vanilla adversarial loss versus the squared adversarial loss on MNIST-to-USPS (top) and USPS-to-MNIST (bottom) UDA. We vary the probability of target users. For both UDA experiments, the SOTA central methods [81] can achieve over 98% accuracies. From left to right, the columns are target domain accuracies, classification losses and adversarial losses of target domain users.

Network architectures. We adopt the same network architecture as the state-of-the-art of UDA [81]. As presented in Fig. B.1, we first use a backbone network to extract sample features. Specifically, we use modified LeNet [92] for digit recognition, ResNet50 [51] for Office and Office-Home datasets, and ResNet101 for the VisDA-C dataset. We use an one-layer bottleneck to reduce the feature dimension. After the bottleneck, we get a representation of 256-dimension. A single fully-connected layer is used for classification at

Table 3.1: Averaged classification UDA accuracies (%) on Office and OfficeHome dataset with 3 non-iid target users and 1 source user. Underlines indicate the occurrence of non-converged results. Standard deviations are included in brackets.

Method	A→D	A→W	D→A	D→W	W→A	W→D	Re→Ar	Re→Cl	Re→Pr	Avg.
Federated methods										
Source only	79.5	73.4	59.6	91.6	58.2	95.8	67.0	46.5	78.2	72.2
non-iid target users w/ 20 (Office) or 45 (OfficeHome) classes per user										
FADE-DANN	85.4 (1.9)	81.8(1.8)	43.1(33)	97.7(0.5)	64.8(0.5)	99.7(0.2)	46.4(37)	34.9(27)	78.8(0.1)	70.3
FADE-CDAN	**92.3(1.2)**	**91.6(0.5)**	**65.9(9.3)**	**98.9(0.2)**	**70.2(0.8)**	**99.9(0.1)**	70.3(1.6)	54.9(4.6)	**82.2(0.1)**	80.7
FedAvg-SHOT	83.6(0.5)	83.1(0.5)	64.7(1.4)	91.7(0.2)	64.7(2.2)	97.4(0.5)	**70.7(0.5)**	**55.4(0.5)**	80.1(0.3)	76.8
iid target users										
FADE-DANN	84.2(1.5)	81.3(0.4)	66.3(0.3)	97.5(1.2)	59.4(10.6)	99.9(0.2)	67.3(0.9)	51.3(0.4)	79.0(0.6)	76.2
FADE-CDAN	**93.6(0.8)**	**92.2(1.3)**	**71.2(1.0)**	**98.7(0.4)**	**71.3(0.7)**	**100(0.0)**	70.6(1.3)	55.1(1.0)	**82.3(0.2)**	81.7
FedAvg-SHOT	**96.3(0.5)**	**94.3(1.1)**	70.9(2.0)	98.4(0.4)	**72.7(0.9)**	99.8(0.0)	**74.8(0.3)**	**60.0(0.1)**	**84.9(0.2)**	83.6
Central methods										
ResNet [51]	68.9	68.4	62.5	96.7	60.7	99.3	53.9	41.2	59.9	67.9
Source [81]	80.8	76.9	60.3	95.3	63.6	98.7	65.3	45.4	78.0	73.8
DANN [46]	79.7	82.0	68.2	96.9	67.4	99.1	63.2	51.8	76.8	76.1
CDAN [92]	92.9	**94.1**	71.0	**98.6**	69.3	**100**	70.9	56.7	81.6	81.7
SHOT [81]	**94.0**	90.1	**74.7**	98.4	**74.3**	99.9	**73.3**	**58.8**	**84.3**	**83.1**

the end. The discriminators are small-scale networks to match the capability of the classifiers. The networks and algorithms are implemented using PyTorch 1.7. The ResNet backbones pre-trained on ImageNet are retrieved from the torchvision 0.8 package.

3.5.1 Digit recognition with imbalanced groups

As discussed in Section 3.4.1, group imbalance could result in the mismatch of group distributions. Here, we empirically evaluate the effect of the imbalanced groups on convergence, adversarial losses and utility performance.

Digit dataset is a standard UDA benchmark built on digit images collected from different environments. 10 digits, from 0 to 9, are included. We follow the UDA protocol of [54] and use two subsets: MNIST and USPS. MNIST dataset contains $60,000$ training images and $10,000$ testing 28×28 gray-scale images. USPS consists of 7291 training and 2007 testing 16×16 gray-scale images. We augment the USPS training set by resizing and random rotation.

Setup. We assume 2 users from source and target domain, respectively. In each round, we select one user with predefined probability. For example, the case that source and target users are of 0.05 and 0.95 probability implies severe imbalance. If a user/group has high

probability, that means the user/group will actively participate in the adversarial learning and the other will activate less. The experiment can also be generalized to multiple users in same frequency while one domain has more users. Both situations imply the imbalance between two groups. In experiments, we fix the batch size to 32 and run one user per communication round. In total, we train the users for global 8600 rounds. In each global round, the users will train locally for 10 iterations. Experiments are repeated 3 times with three fixed seeds. At the beginning, we train the models with adversarial coefficient $\lambda = 0$ when all source users are involved until the classification loss converges. Then, we follow [46, 81] to use the decaying schedule of learning rates and schedule the adversarial coefficient λ from 0 to 1.

Results are reported in Fig. 3.2. Left two figures show the negative impact of imbalanced groups. When the imbalance is severe (large or small target probability), the drop in target accuracies is more obvious. In the middle pane, the convergence curves of imbalanced groups fluctuate more significantly and fail to converge. In the last pane, the imbalanced cases have large adversarial losses which barely decrease by federated iterations. It explains why the corresponding classification tasks fail to converge. When using the squared adversarial losses, the ignored adversarial losses of low-frequent users are reduced during federated learning. Meanwhile, the convergence of utility losses are faster. Thus, the negative impact of imbalanced groups is mitigated.

3.5.2 Object recognition with non-iid users

In Section 3.4.3, we prove that the non-iid distribution of users will lead to a trivial solution which may lose the natural discrepancy between users. For federated classification learning where each user only has a partial set of classes, the loss of user discrepancy will make the representations non-discriminative to classes. Here, we conduct experiments to reveal the impact of the non-iid users.

Dataset. We adopt three object recognition datasets, Office [121] (small size), Office-Home [135] (medium size) and VisDA-C [112] (large size), including image of office products.

The former two are standard benchmarks widely used for UDA. The Office dataset contains three domains: Amazon (A), DSLR (D) and Webcam (W) with 2817, 498, 795 images, respectively. 31 object classes of images are taken under different office environments (corresponding to domains). The Office-Home datasets have 65 categories and 4 domains: Artistic images (Ar), Clip Art (Cl), Product images (Pr), and Real-World images (Re) with 2427, 4365, 4439 and 4357 images, respectively. The VisDA dataset is a challenging large-scale benchmark. The source domain comprises 12-way synthetic classification data. In total, 1.5×10^5 images are synthesized by rendering 3D models and are adapted to $55,000$ unlabeled real-world images.

Setup. In total, 4 users are generated from two domain datasets. First, we let the single source domain user with all classes. Second, we generate 3 non-iid target domain users with partial set of classes following the standard federated setting [96]. For Office dataset, each user has 20 classes and adjacent users have consecutive classes with 10-class stride. For instance, user 1 has class 0 to 20 and user 2 has class 10 to 30. For OfficeHome dataset, each user has 45 classes with 20-class stride. For VisDA-C dataset, each user has 5 classes with 4-class stride. All users in the same domain will have the same number of samples. We select 2 users per communication round when training on OfficeHome. For VisDA-C dataset, we adopt 1 user per round. In this case, the major difficulty comes from non-iid distributions of users conditioned on the subset of classes. In experiments, the parameters for SHOT follows [81]. Details of network architectures and learning setup are discussed in Section B.2.

Baselines. We compare different UDA methods extended by FADE upon the presence of non-iid users. **DANN** [46] is the first work on adversarial domain adaptation based on which many recent methods are developed. **CDAN** [92] is the first to condition the discriminator prediction on the estimated classes, which aligns with our purpose to maximize the mutual information between user (related to classes) and representation. **SHOT** [81] (extended by FedAvg [96]) is the current state-of-the-art method in domain adaptation which does not use source data, assuming approximately mitigated domain shift.

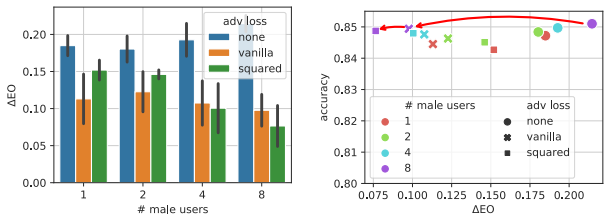
Table 3.2: Comparison of target accuracies on Visda-C dataset.

Methods	Source only	DANN	SHOT	CDAN
Central	46.6	57.6	82.9	73.9
FADE	54.3	56.4	69.2	73.1 (+SHOT)

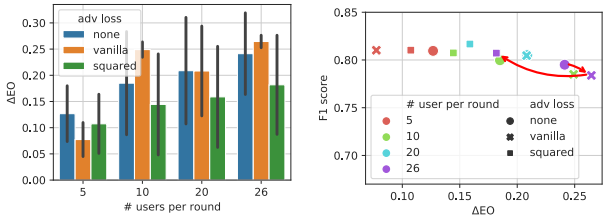
Results. We summarize the results in Tables 3.1 and 3.2. Note that the straightforward extension of DANN without constraints will suffer from the user heterogeneity. Therefore, we observe catastrophic failures by DANN, for example, D→A with only a low accuracy. This kind of failures happens when both D (498) is of less samples than A (2817). The possible reason is that the discriminators fail to sense the position of target domain batches which is a small ratio of all target-domain samples and changes frequently by iterations. In comparison, when regulated by estimated classes, SHOT and methods combined with SHOT perform better. Notably, because SHOT relies on BN states to mitigate domain shift, its accuracies are much worse than its central version. Since SHOT can provide pseudo supervisions which conditions on the estimated users' local classes, DANN+SHOT outperforms DANN. In reverse, DANN helps SHOT to mitigate the domain shift. We further explore CDAN+SHOT, which conditions group discrimination on local classifier outputs (correlated to users' classes). As a result, CDAN+SHOT outperforms other methods and is close to the central version of CDAN. Plus, CDAN+SHOT achieves the best average accuracies when the number of users per round varies from 1 to 4. Remarkably, in the hardest case where only one user is trained per round, CDAN+SHOT gains the best accuracies on 8 out of 9 tasks. In a more challenging large-scale VisDA-C dataset, CDAN+SHOT also shows its advantage against other baselines (see Table 3.2). We note that adversarial methods are more robust to the non-iid users.

3.6 Experiments on Fair Federated Learning

The fair federated learning is motivated by the imbalanced groups in training. For example, when vendor rallies people to use their software and train model with locally collected data, the global model may be biased by the majority, e.g., male users. When a user from another



(a) Results on Adult by varying number of male users.



(b) Results on MCI data by varying number of users per round.

Figure 3.3: FADE with/without adversarial losses. In each subfigure, *left* is fairness measured by ΔEO where smaller values indicates better fairness; *right* is the trade off between fairness and utility where left-top is the preferred balance.

gender uses the software, she/he may find that the model performs poorly. As a result, the minority group vanishes while majority continues to dominate. Thus, a method actively debiasing w.r.t. the groups will be essential to defend the tendency.

Related work. The fairness in federated learning was first discussed in [79] where users are thought to have different capability for computation. Fairness was enforced by increasing the weights of large loss, which was less optimized. In this experiment, we consider the unfairness brought by the difference of group distributions. With FADE, we use a discriminator locally to justify whether the user's representations are biased from the other group. Related central algorithms have been exploited [93, 137**?**]. To the best of our knowledge, we are the first to encourage such group-based fairness in federated setting. Importantly, our method preserve the privacy of group variables. The concerns of the privacy of group variables was previously discussed [48]. In [48], Hashimoto *et al.*assumes the group membership and number of groups are unknown to the central learning server, when users interact with the system and contribute data. Our FADE extends the setting to a distributed framework where the private group information is still unknown to other parties including the aggregation server.

We utilize the Equalized Odds (ΔEO) to evaluate the degree of fairness. Consider a binary classifier $f : \mathcal{Z} \to \{0, 1\}$ predicting label variable y when representations ($z \in \mathcal{Z}$) are sampled from two groups. We denote the conditional p.d.f. $p(z|g, y)$ as $z_{g,y}$ which shapes

the probability of z at group g and class y. An algorithm is said to be fair if the positive ΔEO (defined below) is close to 0.

$$\Delta\text{EO} \triangleq \left|\mathbb{E}_{z_{0,0}}[f(z)] - \mathbb{E}_{z_{1,0}}[f(z)]\right| + \left|\mathbb{E}_{z_{0,1}}[1-f(z)] - \mathbb{E}_{z_{1,1}}[1-f(z)]\right| \tag{3.9}$$

which comprises the absolute difference in false positive rates and the absolute difference in false negative rates.

3.6.1 Fair adult income prediction

Dataset. We evaluate our algorithm on the UCI Adult dataset[1] which is a standard benchmark for fair classification. The dataset consists of over 40,000 vector samples from the 1994 US Census. Each sample includes 14 attributes predicting if his/her income is over 50,000 dollars.

Setup. We adversarially disentangle the unfair representations from the gender. When keeping the total data size fixed, we construct one female user and vary the number of male users. Each synthesized user evenly split the samples in the group. We run FADE for 8,000 communication rounds. In every round, 2 users are selected to train for 1 local iteration on a batch of 64 samples. The accuracies and fairness are evaluated on the left-out 10% samples. The network structure is in Fig. B.2. We set hyper-parameters as $\beta = 0.5$ and the initial learning rate as 10^{-3}.

Results are depicted in Fig. 3.3a. Without adversarial training, the unfairness is aggravated when the imbalance between groups worsens. When more male users are involved, the squared adversarial loss is able to further improve the fairness. Instead, the vanilla adversarial learning performs better when the two groups are balanced. Both adversarial losses will maintain the utility performance close to the non-adversarial method.

3.6.2 Fair MCI detection

Dataset. Mild Cognition Impairment (MCI) is the pre-symptom of Alzheimer's Disease (AD) which typically happens on elders. Early detection of MCI is important for prevention

[1]https://archive.ics.uci.edu/ml/datasets/adult

of AD occurrence and treatment [5, 134]. Details of the dataset is comprised in Section B.2.3 where females forms the majority group (over 94%). The prediction task here is to classify the disease condition, Normal Cognition (NC) or MCI, based on the daily activities (walking speed, etc.).

Setup. In the original dataset, there are 88 users with different number of samples. We notice the imbalance between NC and MCI users will greatly degrade the model quality. To focus on our fairness task, we manually select 26 users such that 13 users was diagnosed as NC at least once and the other 13 ones are stable MCI patients. Because male users are much fewer than female ones, we prefer to select male users when balancing the two classes. After downsampling, users have 39 samples on average. Among the 26 users, there are 6 males and 20 females in total. Details of features, preprocessing and network architectures are deferred to Section B.2.3. We set hyper-parameters as $\beta = 0.5$, the initial learning rate as 10^{-2} and batch size as 16. In the 700 communication rounds, we first train without adversarial losses for 400 rounds and then schedule the λ and learning rates as the Adult experiments.

Results. We compare the convergence of the training F_1-score (utility) and ΔEO (fairness) by varying the number of users per round. As shown in Fig. 3.3b, the unfairness is obvious with ΔEO over 0.2 when no adversarial losses are used. We see that the vanilla adversarial loss failed to debias in most cases. In contrast, the squared adversarial loss stably debias the unfair performance in all cases. When the number of users per round is less than 10, even the non-adversarial loss is more fair. The natural debiasing could be attributed to the random selection of users, which breaks the domination of one group in a short span.

CHAPTER 4

EFFICIENT FEDERATED LEARNING FOR ON-DEMAND AND IN-SITU CUSTOMIZATION

Federated learning (FL) provides a distributed learning framework for multiple participants to collaborate learning without sharing raw data. In many practical FL scenarios, participants have heterogeneous resources due to disparities in hardware and inference dynamics that require quickly loading models of different sizes and levels of robustness. The heterogeneity and dynamics together impose significant challenges to existing FL approaches and thus greatly limit FL's applicability. In this paper, we propose a novel Split-Mix FL strategy for heterogeneous participants that, once training is done, provides *in-situ customization* of model sizes and robustness. Specifically, we achieve customization by learning a set of base sub-networks of different sizes and robustness levels, which are later aggregated on-demand according to inference requirements. This split-mix strategy achieves customization with high efficiency in communication, storage, and inference. Extensive experiments demonstrate that our method provides better in-situ customization than the existing heterogeneous-architecture FL methods.

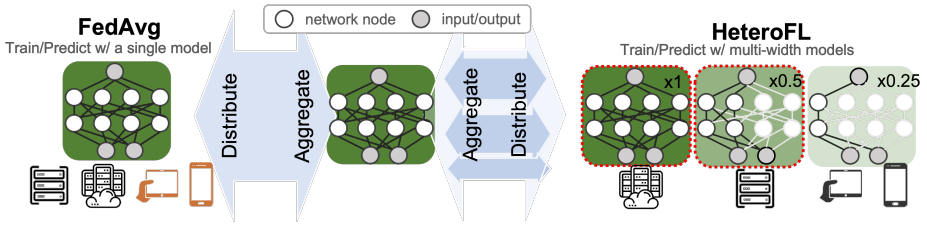
4.1 Introduction

Federated learning (FL) [72] is a distributed learning paradigm that leverages data from remote participants and aggregates their knowledge without requiring their raw data to be transferred to a central server, thereby largely reducing the concerns from data security and privacy. FedAvg [96] is among the most popular federated instantiations, which aggregates knowledge by averaging models uploaded from different participants.

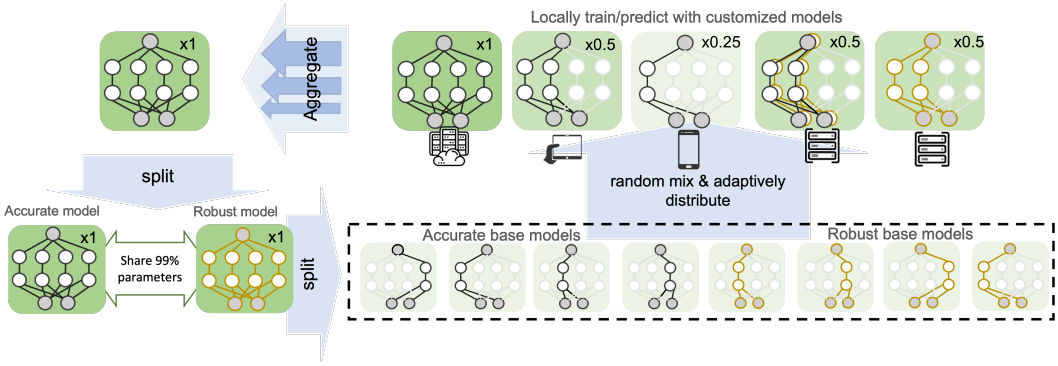
When deploying federated learning, one challenge in real-world applications is the run-time (i.e., test-time) *dynamics*: The requirements on model properties (e.g., inference efficiency, robustness, etc.) can be constantly changing during the run-time, depending on the status of the devices or the outside environment. One common and specific type of dynamics is *resource dynamics*: For each application, the allocated on-device resources (e.g.,

run-time memory, CPU bandwidth, etc.) may vary drastically during run-time, depending on how the resource allocation of the running programs are prioritized on a participant's device [153]. Another type of dynamics is the *robustness dynamics*: The constantly changing outside environment can make different requirements on the safety (or robustness) level of the model [140]. For instance, the quality of real-time videos captured by autonomous cars can suddenly degrade, e.g., on entering a poor-lighted alley or tunnel from a well-lighted avenue, on entering a section of bumpy road which leads to a sudden burst of blurring in the videos, etc. In such cases, a more robust model should be quickly switch in and replace the one used on benign conditions, in order to prevent catastrophic accidents caused by wrong recognition under poor visual conditions. Such dynamic run-time requirements demand the flexibility to customize the model. However, as illustrated in Fig. 4.1a, we show that conventional static-model FL methods, represented FedAvg, cannot provide such customization. A naive solution is to train multiple models with different desired properties and keep them all on device. However, this leads to extra training and storage costs proportional to the number of models. Moreover, since it is not practical to keep all models simultaneously in run-time memory on a resource-limited device, it also introduces inference overhead to swap the models into and out of the run-time memory [33].

To effectively and efficiently train models for on-demand an in-situ customization, new challenges will be raised by the ubiquitous *heterogeneity* of federated learning participants. Fist, the participants can have *resource heterogeneity*: Different participants have different hardware resources available, such as memory, computing power, and network bandwidth [63]. For example, in a learning task for face recognition, clients may use different types of devices (e.g., computers, tablets or smartphones) to participate in learning. To accommodate different hardware, one can turn to more resource-flexible architectures trained by distillation from ensemble [85], partial model averaging [28], or directly combining predictions [125]. Specifically, [28] is the first heterogeneous-width solution (HeteroFL) allowing in-situ model-size switching. Nevertheless, it suffers from under-training in its large models



(a) Illustration of FedAvg [96] with a device-incompatible model and a heterogenous-architecture variant (HeteroFL) [28] with under-trained wide models.



(b) The proposed Split-Mix framework provides in-situ customization of widths and adversarial robustness to address heterogeneity and dynamics, enabling efficient training and inference. In this example, we use a subnet with 1/4 channels (or widths) per layer as a base model for model-width customization. For simplicity, we denote it $\times 0.25$ net, and a $\times 1$ net can be split into 4 $\times 0.25$ base models.

Figure 4.1: Comparison of traditional and proposed methods.

due to local budget constraints as shown in Fig. 4.1a. The degradation could be worsened as facing *data heterogeneity*: The training datasets from participants are not independent and identically distributed (non-*i.i.d.*) [80, 41, 59, 169]. When one device with a unique data distribution cannot afford training a large model, the global large model may not transfer to the unseen distribution [106]. Thus, HeteroFL may not provide effective customization such that more parameters brings in higher accuracy and how to train an effectively customizable model still remains unknown.

To address the aforementioned challenges from heterogeneity and dynamics, we study a novel *Split-Mix* approach to enable FL on heterogeneous devices and achieve *in-situ model customization* for resource efficiency and robustness: The size and robustness of the resultant model can be efficiently customized at run-time. Specifically, we first **split** the complete knowledge in a large model into several small base sub-networks (shards) according to model

widths and robustness levels. To complete the knowledge, we let the base models be fully trained on all clients. To provide customized models, we **mix** selected base models to construct the desired model size and robustness. We illustrate the main idea in Fig. 4.1b. Overall, our contributions can be summarized in three folds:

- Within the domain of heterogeneous federated learning, we are the first to study training a model with the capability of *in-situ customization* with heterogeneous local computation budgets, which cannot be resolved by existing methods yet as shown in Fig. 4.1a.
- To address the challenge, we propose a novel Split-Mix framework that aggregates knowledge from heterogeneous clients into a width- and robustness-adjustable model structure. Remarkably, due to fewer parameters and modular nature, our framework is not only efficient in federated communication and flexibly adaptable to various client budgets *during training*, but also efficient and flexible in storage, model loading and execution *during inference*.
- Empirically, we demonstrate that the performance of the proposed method is better than other FL baselines under heterogeneous budget constraints. Moreover, we show its effectiveness when facing the challenge of data heterogeneity.

4.2 Related Work

Heterogeneous Federated Learning. As increasing concerns have been gained on data privacy leakage in machine learning [35, 147, 149, 57], federated learning (FL) protects data privacy by training the model locally on users' own devices without sharing data. In real-world applications, FL with budget-insufficient devices (e.g., mobile devices) has attracted a great amount of attention. For example, *FedDistill* [85] used logit averages to distill a network of the same size or several prototypes, which will be communicated with users. FedDistill made an assumption that the central server has access to a public dataset of the same learning task, which is impractical for many applications. [49] introduced a distillation-based method after aggregating private representations from all participants. The method closely resembles centralized learning because all encoded samples are gathered, and however it is less efficient

when local clients have large data dimensions or sample sizes. Importantly, the method may not transfer adversarial robustness knowledge through the intermediate representations due to the decoupling of the input and prediction. On the other hand, *HeteroFL* [28] avoids distillation, allowing participants to train different prototypes and sharing parameters among prototypes to reduce redundant parameters. However, HeteroFL also reduces the samples available for training each prototype, which leads to degraded performance. Considering the high cost of training robust models, [56] proposed an efficient way to transfer model robustness from budget-sufficient devices to insufficient ones. *FedEnsemble* [125] is technically related to the proposed approach, which uses ensemble of diverse models to accommodate non-*i.i.d.* heterogeneity. The authors showed that combining multiple base models trained simultaneously in FL can outperform a single base model in testing. A critical difference between the proposed approach and FedEnsemble comes from the challenging problem setting of constrained heterogeneous computation budgets. For the first time, we show that base models can be trained adaptively under budget constraints and used to customize efficient inference networks that can outperform a single model of the same width but trained in a heterogeneous-architecture way.

Customizable Models. To our best knowledge, this is the first paper discussing in-situ customization in federated learning and here we review similar concepts in central learning. First, customization of robustness and accuracy was discussed by [140], where an adjustable-conditional layer together with decoupled batch-normalization were used. The conditional layer enables continuous trade-off but also brings in more parameters. In comparison, a simple weighted combination without additional parameters is used in our method and is very efficient in both communication and inference. In terms of model complexity, a series of work on dynamic neural networks were proposed to provide instant, adaptive and efficient trade-off between accuracy and latency (mainly related to model complexity) of neural networks at the inference stage. Typically, sub-path networks of different complexity [87] or sub-depth networks [60, 150, 143, 163] are trained together. However, due to the large memory

footprint brought by a constant number of channels per layer, the memory footprint at inference is barely reduced. To address the challenge, slimmable neural network (SNN) [158] was proposed to train networks with adaptive layer widths. Distinct from SNN, we consider a more challenging scenario with distributed and non-sharable data and heterogeneous client capabilities.

4.3 Problem Setting

The goal of this work is to develop a heterogeneous federated learning (FL) strategy, that yields a global model, which can be efficiently customized for dynamic needs. Formally, FL minimizes the objective $\frac{1}{\sum_k |D_k|} \sum_{k=1}^{K} \sum_{(x,y)\in D_k} L(f(x;W),y)$, where L is loss function (e.g., cross-entropy loss), f is a model parameterized by W, and $\{D_k\}_{k=1}^{K}$ are the training datasets on K participants with the image-label pairs (x, y). Following the standard FL setting [96], only model parameters can be shared to protect privacy. We require efficient run-time customization on model f for resource dynamics (Section 4.4.2) and robustness dynamics (Section 4.4.3). [1]

When training customizable/adjustable models, significant challenges arise from heterogeneity among clients. In this paper, we consider the following: **1) Heterogeneous computational budgets during training** of clients constrain the maximal complexity of local models. The complexity of deep neural networks can be defined in multiple dimensions, like depths or widths [160, 129]. In this paper, we consider the width of hidden channels, because it can significantly reduce not only the number of model parameters but also the layer output cache. Thus, we assume clients have confined width capabilities $\{R_k \in (0, 1]\}_{k=1}^{K}$, defined by width ratios w.r.t. the original model, i.e., $\times R_k$ net as presented in Fig. 4.1b. Similar to FedAvg and HeteroFL, the same architecture is used by clients and therefore the model width can be tailored according to local budgets. In many applications, there are usually a significant number of devices with insufficient computational budgets [40]. For ex-

[1] Customization of other properties can also be applied within our framework. We consider model size and robustness given their practical importance.

ample, we may assume exponentially distributed budgets in uniformly divided client groups: $R_k = (1/2)^{\lceil 4k/K \rceil}$, i.e., the first group with 1/4 clients is capable for a 1-width and the rest are for $0.5, 0.25, 0.125$-widths respectively. The budget distribution simulate a real scenario where most federated mobile phones prefer a low-energy solution with smaller models for training probably in the background and maintain more resources for major tasks. Other budget distributions, e.g., log-normal distributions, are discussed in Section C.1.5. **2) Heterogeneous data distributions**, e.g., non-*i.i.d.* features, $D_k \sim \mathcal{D}_i$ for ordered domain i, induces additional challenges with a skewed budget distribution, since training a single model on multiple domains [80] or models in a single domain (due to budget constraints) [59, 31] are known to be suboptimal on transfer performance.

In summary, training various model sizes has challenges from 1) resource heterogeneity, where disparity in computation budgets will significantly affect the training of large models because they can only be trained on scarce budget-sufficient clients; 2) data heterogeneity, where the under-trained large models may perform poorly on unseen in-domain or out-of-domain samples.

4.4 Method

To provide efficient *in-situ* customization, we introduce a simple yet powerful principle, SplitMix: *shatter complete knowledge into smaller pieces, and customize by flexible formations of pieces.* Based on the principle, we propose customizations of model size and robustness in the following.

4.4.1 Case Study: Customizable Networks from Budget-Constrained FL

For motivation, we set up a standard non-*i.i.d.* FL benchmark [80] using DomainNet dataset (refer to experimental details in Section 5.4) and study how the budget constraint impedes effective training of customizable models.

Case 1: FL without budget constraint. First, we individually train networks of different widths by FedAvg. We see that the slimmer networks converge slower and are less

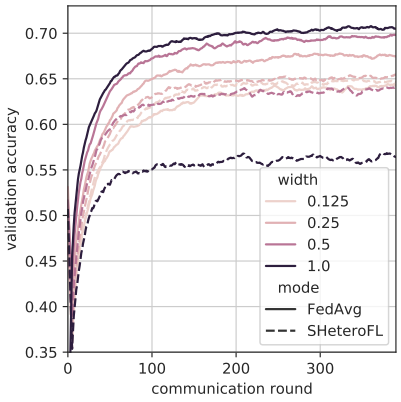


Figure 4.2: Convergence of different-width models on DomainNet.

generalizable, even though they are trained on all clients (solid lines in Fig. 4.2). The results justified the motivation of training wider networks than slimmer ones.

Case 2: FL with budget constraint. Following the above budget constraint, i.e., $R_k = (1/2)^{\lceil 4k/K \rceil}$, we deploy HeteroFL to train budget-compatible prototype models locally. Since HeteroFL was not designed for model customization, data are not fully used: each model prototype is only trained by 1/4 of clients, when clients of the $R_k = 1$ budget can actually afford all slimmer models. Therefore, we extend HeteroFL to a *slimmable* version (SHeteroFL) by training all affordable prototypes locally. As shown in Fig. 4.2, wider models (e.g., $\times 1$ net) converge to validation accuracy lower than not only the FedAvg counterparts, but also the slimmer $\times 0.25$ net, showing that the widest model may not be a good candidate model. Therefore, switching models to wider configurations lowers efficiency but does not improve accuracy, which is *not* a valid customization.

From the perspective of data allocation, it is not surprising that SHeteroFL exhibits a non-monotonous relation between the model width and accuracy. For $\times 1$ nets, only 1/4 clients and data are accessible for training. In comparison, 3/4 data are accessible by $\times 0.25$ nets and the more data empowers $\times 0.25$ nets better generalization ability than $\times 1$ nets.

4.4.2 Customize Model Size

Motivated by the above observations, we propose to increase accessible training data by splitting wide networks into universally-budget-compatible sub-networks and re-mix afterward. The overall FL algorithm is summarized in Algorithm 4.1.

More accessible data by splitting wide networks. Since a wide network cannot fit into budget-insufficient clients, we split it into budget-compatible sub-networks by channels (width) while maintaining the the total width. In terms of memory limitations, each sub-network can be painlessly and individually trained in all clients. However, sequentially training multiple slim base networks could be much slower than training a single integrated one and increases blocking time on communication. Noticing that all base models can be evaluated independently, we instead efficiently train $\times r$ base models in parallel (see Algorithm 4.3). Despite the benefit, a client's budget constraint (R) will limit the number of paralleled base models within $\lfloor R/r \rfloor$, as wider channels result in larger intermediate layer cache (activations), and excludes the rest base models from the client. Fortunately, we can select different sets of base models for a budget-limited client per round, which is inspired by FedEnsemble [125] and is presented in Algorithm 4.2. Hence, all base models can be ever trained on the client for multiple communication rounds, though not continuously every round. Note that since the federated training processes of base models are independent without interference, the training could be as stable as FedAvg with partial participants. In addition, the combination of slimmer base models is flexible and can conform a variety of client budgets.

Boost accuracy by mixture of subnet experts. To craft a wide model, we combine the outputs of multiple $\times r$ base models until the size of the ensemble reaches the same as the number of channels, e.g., $\lfloor R/r \rfloor$ bases for an $\times R$ net. We randomly initialize base models independently such that the diverse bases could extract different features from the same image [6]. Therefore, the ensemble can predict based on a variety of features, resembling an integrated wide network. We follow the common practice, Kaiming's method [50], for

initializing base networks with ReLU layers. As Kaiming's method is width-dependent, we parameterize the initialization based on the width of $\times 1$ net instead of the $\times r$ one, which leads to smaller initial convolutional weights. In Section C.1.3, intensive ablation studies show that the rescaled initialization is critical for improving test accuracy of wider networks.

Algorithm 4.1: Federated Split-Mix Learning

Input: Client datasets $\{D_k\}_{k=1}^K$, the number of total communication rounds T, $M = \lfloor 1/r \rfloor$ randomly initialized base $\times r$ nets parameterized by $\{w_{i,r}\}_{i=1}^M$, client budgets $\{R_k\}_{k=1}^K$, the number of local epochs E, learning rates $\{\eta_t\}_{t=1}^T$

Initialize $\{w_{i,r}^0\}_{i=1}^M$, model indexes $P = \text{Shuffle}([1, \cdots, M])$ and current index $p = 1$
for round $t \in \{1, \cdots, T\}$ **do**
    Initialize $W^t \leftarrow \{w_{i,r}^t \leftarrow 0\}_{i=1}^M$ and aggregation weights $c_i = 0$ for $i \in \{1, \ldots, M\}$
    for $k \in \{1, \cdots, K\}$ **do**
        Sample base models $W_k^{t-1}, p \leftarrow \text{SampleBaseModels}(P, p, \lfloor R_k/r \rfloor, W^{t-1})$
        Send W_k^{t-1} to client k and train $\hat{W}_k^t \leftarrow \text{LocalTrain}(W_k^{t-1}, D_k, E, \eta_t)$
        Aggregate $\hat{w}_{i,r}^{t,k} \in \hat{W}_k^t$ to the server: $w_{i,r}^t \leftarrow w_{i,r}^t + \hat{w}_{i,r}^{t,k}|D_k|$, $c_i \leftarrow c_i + |D_k|$
    Server update $W^t \leftarrow \{w_{i,r}^t \leftarrow w_{i,r}^t / c_i\}_{i=1}^M$
(Optional) Sort $W^T = \{w_{i,r}^T\}_{i=1}^M$ by the descending order of the validation accuracy of $w_{i,r}^T$
Output the customizable model $W^T = \{w_{i,r}^T\}_{i=1}^M$
Customize an R-width model by $F(x; W^T) = \frac{1}{K_R} \sum_{i=1}^{K_R} f(x; w_{i,r}^T)$ where $K_R = \lfloor R/r \rfloor$

Algorithm 4.2: SampleBaseModels(P, p, n, W)

if $p > |P|$ **then** Shuffle P and $p \leftarrow 1$
Initialize $W = \{w_{P[p],r}\}$
if n>1 **then**
    Uniformly sample $n-1$ values into S from $P \backslash \{P[p]\}$ without replacement
    $W \leftarrow W \cup \{w_{i,r}, \forall i \in S\}$
Return $W, p+1$

Algorithm 4.3: LocalTrain(W_k, D_k, E, η)

Initialize models $\hat{W}_k$ by W_k
for $e \in 1, \cdots, E$ **do**
    for batch data (x, y) in D_k **do**
        for $\hat{w}_{i,r} \in \hat{W}_k$ in parallel **do**
            $\hat{w}_{i,r} \leftarrow \hat{w}_{i,r} - \eta \frac{\partial L(f(x; \hat{w}_{i,r}), y)}{\partial \hat{w}_{i,r}}$
Return $\hat{W}_k$

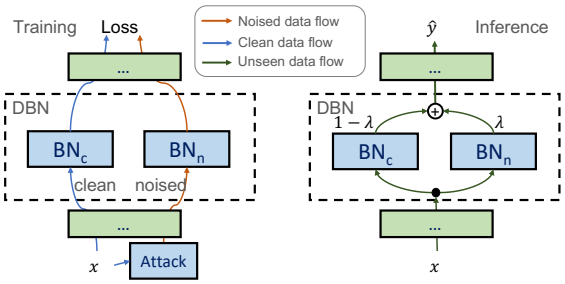


Figure 4.3: Illustration of dual batch-normalization (DBN) in training and inference. The BN_c and BN_n are for clean and noised samples.

4.4.3 Extension to Adversarial Robustness Customization

In this section, we extend the customization from one dimension to two dimensions, by jointly customizing model size and model robustness under adversarial attacks [**?**]. Model robustness has gained increasing interest [**?** 141], especially in high-stakes federated learning applications [**?**]. Adversarial training (AT) [94] is arguably the most popular and effective defense strategy against adversarial attacks. Specifically, it uses on-the-fly adversarial samples as augmentation to improve robustness. Formally, AT minimizes the following augmented loss:

$$L(f) = (1-\lambda_n)L_{CE}(f(x), y) + \lambda_n \max_{\|\delta\|_\infty \leq \epsilon} L_{CE}(f(x+\delta), y), \tag{4.1}$$

where δ is a subtle ϵ-constrained adversarial noise and transfers a clean sample x into an *adversarial sample* $x+\delta$. In Eq. (4.1), L_{CE} is the cross-entropy loss the hyper-parameter and λ_n trades off accuracy (the 1st term) and robustness (the 2nd term). When $\lambda_n = 0$ or 1, the optimization yields a *standard-training* (ST) model or an AT model, respectively. Since an AT model is commonly less accurate in predicting standard images [132], there is usually no such a sweet point of λ_n simultaneously maximizing robustness and accuracy, and one typically needs to carefully gauge the trade-off according to the demand of robustness in specific application context.

Splitting and sharing parameters. Since standard performance and adversarial robustness are irreconcilable, we can directly use two separated ST and AT models to maximally capture the each property. But do we really need two totally separated models?

Intuitively, the two models share some common knowledge, given that an adversarial image share a large part of common features with its original version. As introduced by [151], sharing all parameters except the batch-normalization (BN) layers can maximize robustness and accuracy by expertised BNs, respectively. Accordingly, we propose to split BN layers (instead of the whole model) into two sub-components: one for standard performance and the other for robustness. At training time, the first loss of Eq. (4.1) is computed with clean BN (BN_c) merely and the second adversarial loss is computed with noised BN (BN_n). The FL local training is elaborated in Algorithm C.1. As the two BNs are decoupled, there is no more trade-off in loss Eq. (4.1) and thereby we choose $\lambda_n = 0.5$ to balance their effects on parameter updates.

Customizable layer-wise mixing. After training, the problem is how to mix the two models (with different BNs) for prediction. A straightforward solution is averaging their outputs. However, forwarding memory footprint will be doubled in this way, as the two models have to be both executed separately. To avoid the doubled intermediate outputs, a gate function can be used to adaptively choose BNs like [86]. Inspired by the method, we further propose a simple parameter-free method by weighted-averaging outputs of each BN layer (see Fig. 4.3):

$$\text{DBN}(x) = (1 - \lambda)\text{BN}_c(x) + \lambda\text{BN}_n(x), \tag{4.2}$$

given a BN-layer input x. We note that the averaging strategy is entirely training-free and does not use extra parameters, and the customization weight λ is intuitive for trade-offs between SA and RA.

Lastly, it is remarkable that the DBN structure is rather lightweight in terms of model complexity. As investigated in [158] (Table 2), the parameters of BN is no more than 1% in popular deep architectures, e.g., ResNet or MobileNet. Therefore, we can plug DBN into base models in place of BN, replace Algorithm 4.3 with Algorithm C.1, and jointly customize robustness and model widths.

4.5 Empirical Studies

We design experiments to compare the proposed method against FL classification benchmarks. For **class non-*i.i.d*** configuration, we use CIFAR10 dataset [73] with preactivated ResNet (PreResNet18) [51]. CIFAR10 contains over $50,000$ 32×32 images of 10 classes. The CIFAR10 data are uniformly split into 100 clients and distribute 3 classes per client. For **(feature) non-*i.i.d.*** configuration, we use Digits with a CNN defined and DomainNet datasets [80] with AlexNet extended with BN layers after each convolutional or linear layer [80]. The first dataset is a subset (30%) of Digits, a benchmark for domain adaption [111]. Digits has 28×28 images and serves as a commonly used benchmark for FL [19, 96, 78]. The dataset includes 5 different domains: MNIST [74], SVHN [102], USPS [62], SynthDigits [46], and MNIST-M [46]. The second dataset is DomainNet [110] processed by [80], which contains 6 distinct domains of large-size 256×256 real-world images: Clipart, Infograph, Painting, Quickdraw, Real, Sketch. Each domain of Digits (or DomainNet) are split into 10 (or 5) clients, and therefore 50 (or 30) clients in total. We defer other details such as hyper-parameters to Section D.2, and focus on discussing the results.

4.5.1 Customize model sizes

In this section, we evaluate the proposed Split-Mix on tasks of customizing model sizes through adjusting model widths. Recall that in Section 4.3, we assume a specific heterogeneous training budget to facilitate our discussion, such that one client can only train models within a maximal width, and the resource distribution is imbalance among clients: $R_k = (1/2)^{\lceil 4k/K \rceil}$. Following the common FL setting, we do not consider algorithms that use public data or representation sharing in our baselines.

Baselines. As an ideal upper bound but a memory-incompatible baseline, we (re-)train networks from scratch by FedAvg to obtain individual models with different widths. The state-of-the-art heterogeneous-architecture FL method is *HeteroFL* [28], which trains different slim models in different clients. For a fair comparison, we extend HeteroFL with bounded-slimmable training in clients who can afford the larger models, named *SHeteroFL*.

For example, if a client in HeteroFL can afford $\times 0.5$ net, then the client meanwhile trains $\times 0.25$ net and other smaller subnets in slimmable training manner [158] by SHeteroFL.

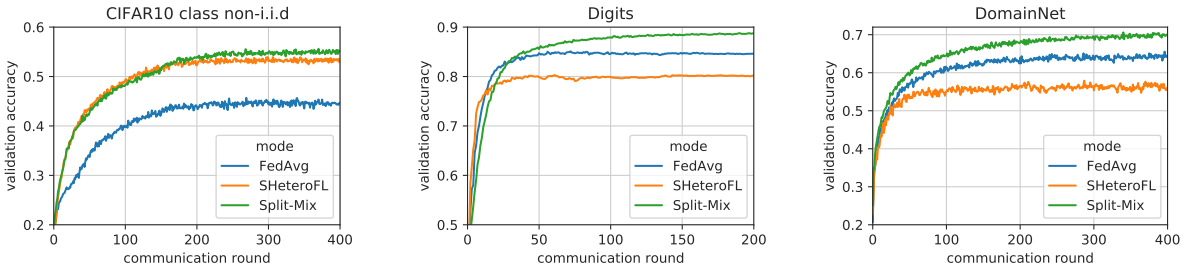


Figure 4.4: Validation accuracy of budget-compatible full-wdith nets by iterations.

Convergence. In Fig. 4.4, we compare the convergence curves of full-width models on three datasets. For FedAvg, the budget-compatible width is $\times 0.125$. Both for SHeteroFL and Split-Mix, the full-width is $\times 1$. Compared to baselines, the proposed Split-Mix converges faster, largely due to the splitting strategy. Specifically, because all base models are independently trained, the convergence of each base model mainly depends on how frequently they are trained. When enough clients participate FL, the training frequency of the $\times 1$ net is more than one, as all bases will be selected at least once in a communication round.

Performance. In Table 4.1, we compare the test accuracy of different model widths. We measure the latency in terms of MACs (number of multiplication-and-addition operations) and model size in parameter numbers. With the same width, our method outperforms SHeteroFL, using much fewer parameters and thus conducts inference in much lower latency. Remarkably, compared to the best model by SHeteroFL, e.g., 81.8% $\times 1$ net in CIFAR10, Split-Mix uses only 1.8% parameters and 1.6% MACs ($\times 0.125$ net) to reach a similar level of test accuracy. We notice that SHeteroFL has a much lower test accuracy with $\times 0.125$ net on the CIFAR10 dataset. By investigating the loss curves, we find that the inference between parameter-shared different prototypes results in unstable convergence of the $\times 0.125$ net. Attributed to the independent splitting, our method is more stable on convergence with different widths. Remarkably, Split-Mix only requires 12.7% parameters and 19.8% MACs (by $\times 1$ ensemble) to achieve the comparable accuracy as the $\times 1$ individual network

Table 4.1: Test results of customizing model width. MACs and the number of parameters are counted at inference time. Grey texts indicate that the training cannot conform the predefined budget constraint. The 'M' after metric values means $\times 10^6$.

	Individual FedAvg			SHeteroFL			Split-Mix (ours)		
width	Acc	MACs	#Params	Acc	MACs	#Params	Acc	MACs	#Params
				CIFAR10 class non-*i.i.d* FL					
×0.125	43.4%	0.9M	0.2M	**49.1%**	0.9M	0.2M	48.0%	0.9M	0.2M
×0.25	~~45.7%~~	~~3.5M~~	~~0.7M~~	**51.4%**	3.5M	0.7M	51.1%	**1.8M**	**0.4M**
×0.5	~~50.3%~~	~~14.0M~~	~~2.8M~~	51.5%	14.0M	2.8M	**52.1%**	**3.6M**	**0.7M**
×1	~~53.3%~~	~~55.7M~~	~~11.2M~~	49.9%	55.7M	11.2M	**52.7%**	**7.2M**	**1.4M**
				Digits feature non-*i.i.d* FL					
×0.125	86.1%	0.1M	0.2M	**86.8%**	0.1M	0.2M	84.6%	0.1M	0.2M
×0.25	~~87.3%~~	~~0.4M~~	~~0.9M~~	**87.9%**	0.4M	0.9M	87.5%	**0.2M**	**0.4M**
×0.5	~~88.7%~~	~~1.3M~~	~~3.6M~~	86.9%	1.3M	3.6M	**89.0%**	**0.5M**	**0.9M**
×1	~~89.6%~~	~~4.8M~~	~~14.2M~~	81.3%	4.8M	14.2M	**89.8%**	**0.9M**	**1.8M**
				DomainNet feature non-*i.i.d* FL					
×0.125	67.2%	2.5M	0.9M	66.9%	2.5M	0.9M	**68.4%**	2.5M	0.9M
×0.25	~~69.8%~~	~~7.5M~~	~~3.6M~~	67.8%	7.5M	3.6M	**71.9%**	**5.0M**	**1.8M**
×0.5	~~72.6%~~	~~25.5M~~	~~14.3M~~	66.9%	25.5M	14.3M	**73.0%**	**9.9M**	**3.6M**
×1	~~72.9%~~	~~92.5M~~	~~57.1M~~	58.7%	92.5M	57.1M	**74.2%**	**19.8M**	**7.2M**

on Digits. Results on CIFAR10 and DomainNet show a potential limitation of our method. The accuracy of Split-Mix is not comparable with the wider individual models due to the locally limited complexity. However, the limitation is more from the problem setting itself, and will not undermine our advantage in budget-limited FL, since wide individual models cannot be trained in this case.

Client-wise evaluation. In addition to comparisons of inference on average, we also demonstrate the statistics of test accuracy and the training and communication efficiency in Fig. 4.5. Conforming the budget constraints, our method outputs more accurate full-width models, transfer fewer parameters per round and execute fewer multiplication-and-add operations for gradient descent than SHeteroFL, either in terms of average or variance. Because only the individual ×0.125 net can fit into the budget constraint, FedAvg requires the least MACs and parameters, which however significantly sacrifices the final accuracy.

Domain-wise evaluation. To understand why Split-Mix outperforms SHeteroFL, we

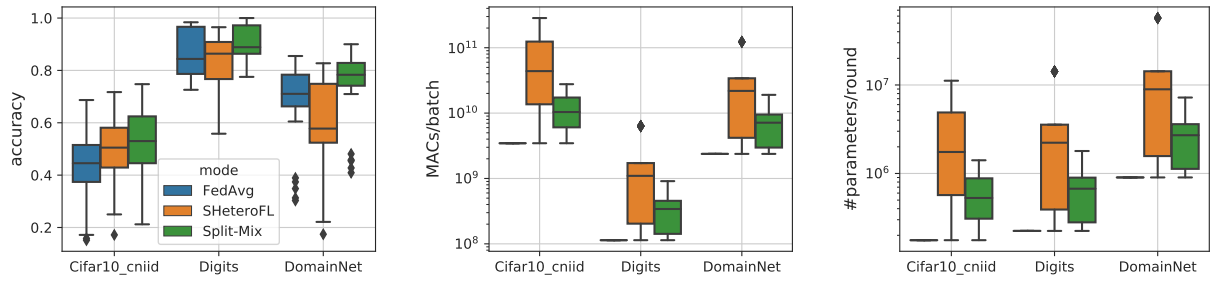


Figure 4.5: Client-wise statistics of test accuracy, training and communication efficiency by budget constraints. The MACs quantify the complexity of one batch optimization in a client, and the number of parameters per round round are the ones uploaded to (or downloaded from) a server. Test accuracy is by the full-width networks. The results of FedAvg are from budget-compatible $\times 0.125$ nets.

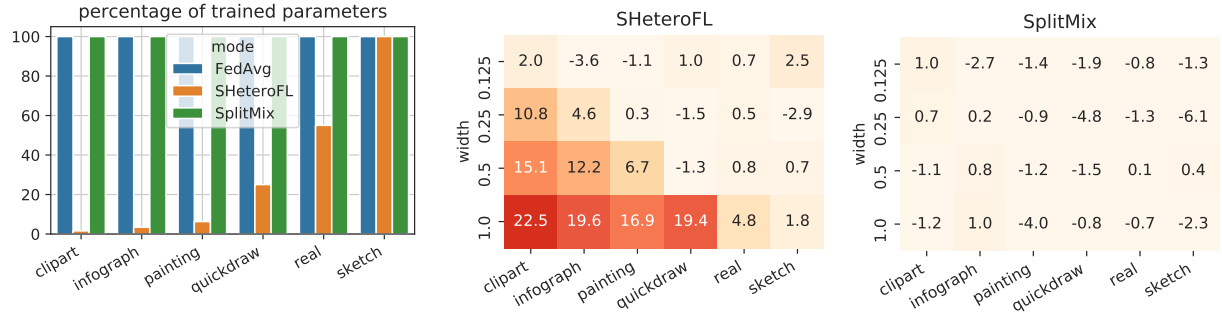
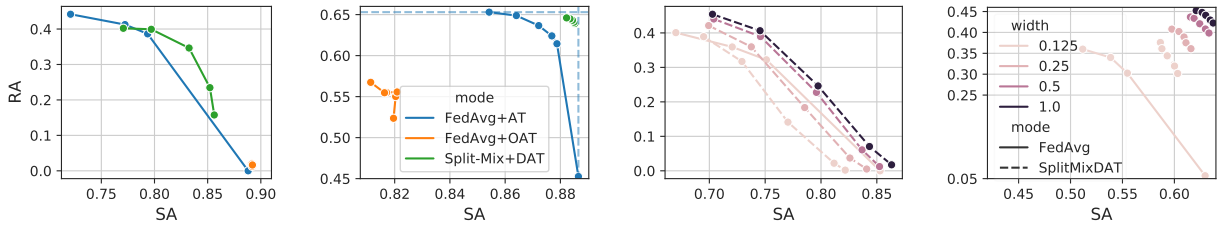


Figure 4.6: Per domain in DomainNet, the total percentage of parameters that are locally trained (the left figure) and the accuracy (%) drops compared to FedAvg individual models (right two figures).

investigate the total percentage of parameters that can be trained in each domain, in Fig. 4.6 (Left). We count the total parameters that were ever trained in clients of a domain during the learning. Thanks to the base-model sampling strategy (i.e., Algorithm 4.2), Split-Mix allows all base models, rather than a subset, to be trained on all clients. On the other hand, varying client budgets greatly impacted SHeteroFL by limiting the width of models trained in budget-insufficient clients, e.g., in the clipart, infograph and painting domains. Hence, SHeteroFL leaves a large amount of parameters under-trained and suffers from larger accuracy losses in the three domains and wider models. In comparison, Split-Mix not only has less accuracy drop but is also more stable in all domains.

4.5.2 Customize robustness

Training and evaluation. For local AT, we use an n-step projected gradient descent (PGD) attack [94] with a constant noise magnitude ϵ. Following [94], we set $(\epsilon, n) = (8/255, 7)$, and attack inner-loop step size $2/255$, for training, validation, and test. For simplicity, we temporarily *relax the budget constraint R_k and let the base model be $\times 1$ net.* For comparison, we extend FedAvg with AT which yields individual models optimized with different trade-off variables, i.e., $\lambda_n \in \{0, 0.1, 0.2, 0.3, 0.5, 1\}$. Also, we extend FedAvg with state-of-the-art in-situ trade-off method, OAT [140], as a baseline. Split-Mix+DAT is an extension of Split-Mix by the proposed DBN-based AT in Algorithm C.1. We evaluate and contrast models in two metrics: *standard accuracy (SA)* on the clean test samples and *robust accuracy (RA)* on adversarial images generated from the clean test set by the PGD attack. Both metric values are averaged by users. We evaluate the trade-off effectiveness by comparing the RA-SA curves, which is better approaching the right-upper corner. To plot the curves for FedAvg+OAT and Split-Mix+DAT, we vary their condition variable λ in $\{0, 0.2, 0.5, 0.8, 1\}$.



(a) CIFAR10 (b) Digits (c) CIFAR10 (d) DomainNet

Figure 4.7: Trade-off between robust accuracy (RA) and standard accuracy (SA) with full width (a,b) and customizable widths (c,d).

Trade-off curves are presented in Fig. 4.7 (a) and (b). Since the naive extension of OAT with FedAvg adopts heterogeneous objectives and over-parameterization of conditional layers which suffers from convergence issues, its adversarial training does not converge and RA is incredibly poor in some cases. As a result, the trade-off curve is not smooth. Instead, the proposed Split-Mix+DAT method has a smoother trade-off curve without heavy conditional

training or over-parameterization. By training in one pass, Split-Mix+DAT even outperforms and is more efficient than the FedAvg+AT baselines.

Joint customization of width and robustness under budget constraints. Now we consider the width customization and the training budgets as Section 4.5.1. Due to the constraint, FedAvg can only train $\times 0.125$ net. We omit OAT for its unstable convergence and use SplitMixDAT as a short name of Split-Mix+DAT. In Fig. 4.7 (c) and (d), the trade-off curves with different widths are depicted. As the width increases, both RA and SA of SplitMixDAT are improved, when they are smoothly traded off. The results demonstrate the flexibility and the modular nature of our method.

CHAPTER 5

OUTSOURCING TRAINING WITHOUT UPLOADING DATA

As deep learning blooms with growing demand for computation and data resources, outsourcing model training to a powerful cloud server becomes an attractive alternative to training at a low-power and cost-effective end device. Traditional outsourcing requires uploading device data to the cloud server, which can be infeasible in many real-world applications due to the often sensitive nature of the collected data and the limited communication bandwidth. To tackle these challenges, we propose to leverage widely available *open-source data*, which is a massive dataset collected from public and heterogeneous sources (e.g., Internet images). We develop a novel strategy called Efficient Collaborative Open-source Sampling (ECOS) to construct a proximal proxy dataset from open-source data for cloud training, in lieu of client data. ECOS probes open-source data on the cloud server to sense the distribution of client data via a communication- and computation-efficient sampling process, which only communicates a few compressed public features and client scalar responses. Extensive empirical studies show that the proposed ECOS improves the quality of automated client labeling, model compression, and label outsourcing when applied in various learning scenarios.

5.1 Introduction

Nowadays, powerful machine learning services are essential in many devices that supports our daily routines. Delivering such services is typically done through client devices that are power-efficient and thus very restricted in computing capacity. The client devices can collect data through built-in sensors and make predictions by machine learning models. However, their stringent computing power often makes the local training prohibitive, especially for high-capacity deep models. One widely adopted solution is to outsource the cumbersome training to cloud servers equipped with massive computational power, using machine-learning-as-a-service (MLaaS). Amazon Sagemaker [83], Google ML Engine [15], and Microsoft Azure ML Studio [10] are among the most successful industrial adoptions,

where users upload training data to designated cloud storage, and the optimized machine learning engines then handle the training. One major challenge of the outsourcing solution in many applications is that the local data collected are sensitive and protected by regulations, therefore prohibiting data sharing. Notable examples include General Data Protection Regulation (GDPR) [1] and Health Insurance Portability and Accountability Act (HIPPA) [4].

On the other hand, recent years witnessed a surging amount of general-purpose and massive datasets authorized for public use, such as ImageNet [26], CelebA [90], and MIMIC [67]. Moreover, many task-specific datasets used by local clients can be well considered as biased subsets of these large public datasets [110, 80]. Therefore, the availability of these datasets allows us to use them to model confidential local data, facilitating training outsourcing without directly sharing the local data. One approach is to use the private client dataset to craft pseudo labels for a public dataset in a confidential manner [168, 109], assuming that the public and local data are identically-and-independently-distributed (*iid*). In addition, Alon *et al.* showed that an *iid* public data can strongly supplement client learning, which greatly reduces the private sample complexity [7]. However, the *iid* assumption can often be too strong for general-purpose *open-source* datasets, since they are usually collected from heterogeneous sources with distributional biases from varying environments. For example, a search of 'digits' online yields digits images from handwriting scans, photos, to artwork of digits.

In this paper, we relax the *iid* assumption in training outsourcing and instead consider the availability of an open-source dataset. We first study the gap between the *iid* data and the heterogeneous open-source data in training outsourcing, and show the low sample efficiency of open-source data. We show that in order to effectively train a model from open-source data that is transferable to the client data, the open-source data needs to communicate more samples than those of *iid* data. The main reason behind such low sample efficiency is that we accidentally included out-of-distribution (OoD) samples, which poison the training and significantly degrade accuracy at the target (client) data distribution [14]. We propose a

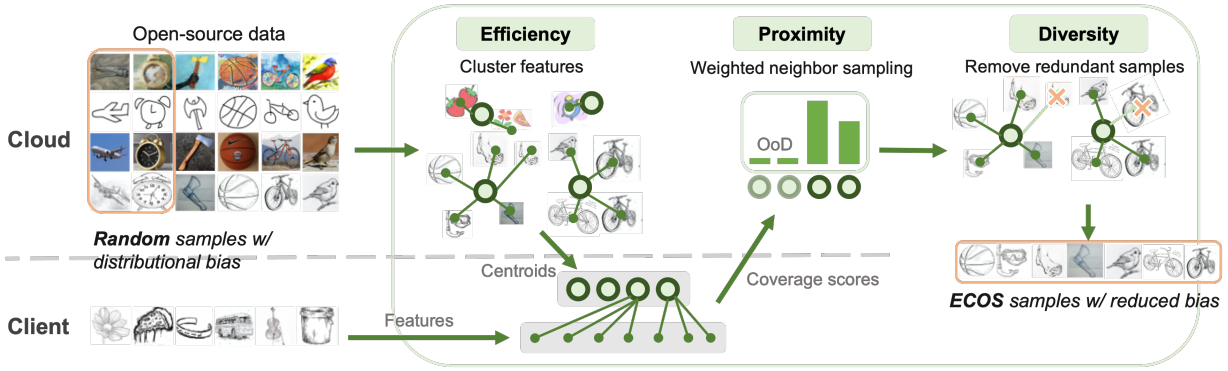


Figure 5.1: Illustration of the proposed ECOS framework. Instead of uploading local data for cloud training, ECOS downloads the centroids of clustered open-source features to *efficiently* sense the client distribution, where the client counts the local neighbor samples of the centroids as the coverage score. Based on the the scores of centroids, the server adaptively samples *proximal* and *diverse* data for training a transferable model on the cloud.

novel framework called Efficient Collaborative Open-source Sampling (ECOS) to tackle this challenge, which filters the open-source dataset through an efficient collaboration between the client and server and does not require client data to be shared. During the collaboration, the server sends compressed representative features (centroids) of the open-source dataset to the client. The client then identifies and excludes OoD centroids and returns their privately computed categorical scores to the server. The server then adaptively and diversely decompresses the neighbors of the selected centroids. The main idea is illustrated in Fig. 5.1.

Our major contributions are summarized as follows:

• *New problem and insight*: Motivated by the strong demands for efficient and confidential outsourcing, using public data in place of the client data is an attractive solution. However, the impact of heterogeneous sources of the public data, namely open-source data, is rarely studied in existing works. Our empirical study shows the potential challenges due to such heterogeneity.

• *New sampling paradigm*: We propose a new unified sampling paradigm, where the server only sends very few query data to the client and requests very few responses that efficiently and privately guide the cloud for various learning settings on open-source data. To our best knowledge, our method enables efficient cloud outsourcing under the most practical

assumption of open-source public data, and does not require accessing raw client data or executing cumbersome local training.

• *Compelling results*: In all three practical learning scenarios, our method improves the model accuracy with pseudo, manual or pre-trained supervisions. Besides, our method shows competitive efficiency in terms of both communication and computation.

5.2 Related Work

There are a series of efforts studying how to leverage the data and computation resources on the cloud to assist client model training, especially when client data cannot be shared [156, 142]. We categorize them as follows: *1) Feature sharing*: Methods like group knowledge transfer [49], split learning [136] and domain adaptation [32, 30] transfer edge knowledge by communicating features extracted by networks. To provide a theoretical guarantee of privacy protection, [105] proposed an advanced information removal to disentangle sensitive attributes from shared features. In the notion of rigorous privacy definition, Liu *et al.* leverage public data to assist private information release [89]. Earlier, data encryption was used for outsourcing which however is too computation-intensive for a client and less applicable for large-scale data and deep networks [21, 76]. Federated Learning (FL) considered the same constraint on data sharing but the private knowledge is shared via models trained locally [97]. *2) Private labeling:* PATE and its variants were proposed to generate client-approximated labels for unlabeled public data, on which a model can be trained [108, 109]. Without training multiple models by clients, Private kNN was a more efficient alternative which explored the private neighborhood of public images for labeling [168]. These approaches are based on a strong assumption of the availability of public data that is *iid* as the local data. This paper considers a more practical yet challenging setting where public data are from multiple agnostic sources with heterogeneous features.

Sampling from public data has been explored in central settings. For example, Xu *et al.* [154] used a few target-domain samples as a seed dataset to filter the open-domain datasets by positive-unlabeled learning [91]. In self-supervised contrastive learning, model-

aware K-center (MAK) used a model pre-trained on the seed dataset to find desired-class samples from open-world dataset [65]. Though these methods provided effective sampling, they are less applicable when the seed dataset is placed at the low-energy edge, because the private seed data at the edge cannot be shared with the cloud for filtering and the edge device is incapable of computation-intensive training. To address these challenges, we develop a new sampling strategy requiring only light-weight computation at the edge.

5.3 Outsourcing Model Training With Open-Source Data

5.3.1 Problem Setting and Challenges

Motivated in Section 5.1, we aim to outsource the training process from computation-constrained devices to the powerful cloud server, where a proxy public dataset without privacy concerns is used in place of the client dataset for cloud training. One solution is (private) client labeling by k-nearest-neighbors (kNN) [168], where the client and cloud server communicate the pseudo-label of a public dataset privately and the server trains a classifier by the labeled and unlabeled samples in a semi-supervised manner. The success of this strategy depends on the key assumption that public data in the cloud and private data in the client are *iid*, which is rather strong in practice and thus prevents it from many real-world applications. In this work, we make a more *realistic* assumption that the public

Table 5.1: Test accuracy (%) with different client domains (columns). Cloud data are identically distributed as the client data (*ID*) or including more data from 5 distinct domains (*ID+OoD*) without overlapped samples. We first label a number of randomly selected cloud examples (i.e., sampling budget) privately by client data [168], and then train a classifier to recognize digit images. The privacy cost ϵ is accounted for in the notion of differential privacy. Larger budgets imply more privacy and communication costs. More results on different settings are enclosed in Section D.2.3.

Cloud Data	Sampling Budget	MNIST Acc (%) ↑	$\epsilon\downarrow$	SVHN Acc (%) ↑	$\epsilon\downarrow$	USPS Acc (%) ↑	$\epsilon\downarrow$	SynthDigits Acc (%) ↑	$\epsilon\downarrow$	MNIST-M Acc (%) ↑	$\epsilon\downarrow$	Average Acc (%) ↑	$\epsilon\downarrow$
ID	1000	**84.3**±2.4	4.48	**51.6**±1.4	4.08	**87.1**±0.5	4.51	**73.2**±1.5	4.57	**55.5**±1.0	4.46	**70.4**	4.42
ID+OoD	1000	78.0±3.5	4.30	40.6±1.6	3.75	82.2±2.7	4.32	62.1±1.6	4.41	49.1±1.0	4.27	62.4	4.21
	8000	82.2±4.1	5.89	47.9±1.8	5.89	85.4±0.5	5.89	64.4±3.6	5.89	53.3±2.2	5.89	66.6	5.89
	16000	82.6±1.4	7.17	48.5±1.7	7.17	86.7±1.9	7.17	67.5±2.3	7.17	52.0±3.0	7.17	67.4	7.17
	32000	84.1±1.6	9.32	49.4±0.2	9.32	86.8±2.0	9.32	68.5±0.1	9.32	53.0±2.7	9.32	68.4	9.32

datasets are as accessible as *open-source* data. The open-source data consists of biased features from multiple heterogeneous sources (feature domains), and therefore includes not only in-distribution (ID) samples similar to the client data but also multi-domain OoD samples.

The immediate question is how the OoD samples affect the outsourced training. In Table 5.1, we empirically study the problem by using a 5-domain dataset, Digits, where 50% of one domain is used on the client and the remained 50% together with the other 4 domains serve as the public dataset on the cloud. To conduct the cloud training, we leverage the client data to generate pseudo labels for the unlabeled public samples, following [168]. It turns out that the presence of OoD samples in the cloud greatly degrades the model accuracy. The inherent reason for the degradation is that the distributional shift of data [117] compromised the transferability of the model to the client data [146].

Problem formulation by sampling principles. Given a client dataset D^p and an open-source dataset D^q, the goal of open-source sampling is to find a proper subset S from D^q, whose distribution matches D^p. In [65], Model-Aware K-center (MAK) formulated the sampling as a principled optimization:

$$\min_{S \subset D^q} \Delta(S, D^p) - H(S \cup D^p; D^q), \tag{5.1}$$

where $\Delta(S, D^p) := \mathbb{E}_{x' \in D^p}[\min_{x \in S} \|x - x'\|^2]$ measures *proximity* as the set difference between S and D^p, and the latter $H(S \cup D^p; D^q) := \max_{x' \in D^q} \min_{x \in S \cup D^p} \|x - x'\|^2$ measures *diversity* by contradicting $S \cup D^p$ and D^q (suppose D^q is the most diverse set). Solving Eq. (5.1) results in an NP-hard problem that is intractable [23], and MAK provides an approximated solution by a coordinate-wise greedy strategy. It first pre-trains the model representations on D^p and finds a large candidate set with the best proximity to extracted features. Then, it incrementally selects the most diverse samples from the candidate set until the sampling budget is used up.

Though MAK is successful in the central setting, it is not applicable when D^p is isolated from cloud open-source data and is located at a resource-constrained client for two reasons:

1) *Communication inefficiency.* Uploading client data may result in privacy leakage, sending public data to the client is a direct alternative but the cost can be prohibitive. 2) *Computation inefficiency.* Pre-training a model on D^p or proximal sampling (which computes the distances between paired samples from D^q and D^p) induces unaffordable computation overheads for the low-energy client.

5.3.2 Proposed Solution: Efficient Collaborative Open-Source Sampling (ECOS)

To address the above challenges, we design a new strategy that 1) uses compressed queries to reduce the communication and computation overhead and 2) uses a novel principled objective to effectively *sample* from open-source data with the client responses of the compressed queries.

Construct communication-efficient and an informative query set $\hat{D}^q$ at cloud. Let d be the number of pixels of an image, the communication overhead of transmitting D^q to the client is given by $\mathcal{O}(d|D^q|)$. For communication efficiency, we optimize the following two factors:

i) *Data dimension d.* First, we transmit extracted features instead of images to reduce the communication overhead to $\mathcal{O}(d_e|D^q|)$ where d_e is a much smaller embedding dimension. For accurate estimation of the distance Δ, a discriminative feature space is essential. Depending on resources, one may consider hand-crafted features such as HOG [25], or an off-the-shelf pre-trained model such as ResNet pre-trained on ImageNet.

ii) *Data size $|D^q|$.* Even with compression, sending all data for querying is inefficient due to the large size of open-source data $|D^q|$. Meanwhile, too many queries would cast unacceptable privacy costs to the client. As querying on similar samples leads to redundant information in querying, we propose to reduce such redundancy by selecting informative samples. We use the classic clustering method KMeans [44] for compressing similar samples by clustering them, and collect the R mean vectors or *centroids* into $\hat{D}^q = \{c_r\}_{r=1}^R$. We denote R as the *compression size.* The selected centroids can be decompressed by the cluster assignment into corresponding original samples with rich features.

New sampling objective. We propose to optimize a communication-efficient surrogate loss:

$$\min_{\hat{S}\subset\hat{D}^q, S\subset D^q} \underbrace{\Delta(\hat{S},\hat{D}^p)+\Delta(\hat{S},S)}_{\text{proximity}} - \underbrace{H(S;D^q)}_{\text{diversity}}, \tag{5.2}$$

where $\hat{S}$ (or $\hat{D}^q$) is the compressed centroid substitute of S (or D^q). $\hat{D}^p$ consists of the features of D^p. Different from Eq. (5.1), we decompose the proximity term into two in order to facilitate communication efficiency leveraging $\hat{D}^q$. We solve the optimization problem in a greedy manner by two steps at the client and the cloud, respectively:

i) At the *client step*, we optimize $\Delta(\hat{S},\hat{D}^p)$ to find a set of centroids ($\hat{S}\subset\hat{D}^q$) that are proximal to the client set $\hat{D}^p$. Noticing that $\hat{D}^q$ contains the cluster centroids, we take advantage of the property to define a novel proximity measure of the cluster r: Centroid Coverage (CC), denoted as v_r. Upon receiving centroids from the cloud, the client use them to partition the local data into $\{\mathcal{C}_r^p\}_{r=1}^R$ where $\mathcal{C}_r^p$ denotes the r-th cluster partition of local data. We compute the CC score by the cardinality of the neighbor samples of the centroid r, i.e., $v_r=|\mathcal{C}_r^p|$. To augment the sensitivity to the proximal clusters, we scale the CC score by a non-linear function $v_r'=\psi_s(|\mathcal{C}_r^p|)$, where the scale function $\psi_s(x)=x^s$ is parameterized by s.

ii) At the *cloud step*, we optimize the proximity of S w.r.t. the proxy set $\hat{S}$, i.e., $\Delta(\hat{S},S)$, and remove redundant and irrelevant samples from the candidate set to encourage diversity, i.e., $-H(S;D^q)$. As samples among clusters are already diversified by KMeans, we only need to promote the in-cluster *diversity*. To this end, we reduce the sample redundancy per cluster at cloud by K-Center [122], which heuristically find the most diverse samples. Such design transfers the diversity operation to cloud and thus reduces the local computation overhead. To maintain the *proximity*, the K-Center is applied within each cloud cluster and the sampling budget per cluster is proportional to their vote numbers and the original cluster sizes. With the normalized scores, we compute the sampling budget per cluster which is upper bounded by the ratio of the cluster in the cloud set.

Algorithm 5.1: Efficient collaborative open-source sampling (ECOS)

Require: Client dataset D^p, cloud query dataset D^q, sampling budget B, compression size R, distance function $\Delta(x, S) = \min_{y \in S} \|x - y\|$, feature extractor $\phi(\cdot)$, initial sample set $S = \emptyset$, score scale function $\psi_s(x) = x^s$.

Cloud creates a compressed dataset $\hat{D}^q = \text{KMeans}_R(\{\phi(x) | x \in D^q\})$; ▷ Compress R Centroids

▷▷▷ **Client End** ▷▷▷

Download the feature extractor ϕ and $\hat{D}^q$;

Use centroids $\hat{D}^q$ to partition $\hat{D}^p = \{\phi(x) | x \in D^p\}$ into clusters $\{\mathcal{C}_r^p\}_{r=1}^R$;

Compute the Centroid Coverage (CC) scores: $v_r = |\mathcal{C}_r^p|, \ \forall r \in \{1, \ldots, R\}$;

Upload scaled cluster scores $\{v_r' = \psi_s(v_r)\}_{r=1}^R$; ▷ Proximity

◁◁◁ **Cloud End** ◁◁◁

Partition D^q into clusters $\{\mathcal{C}_r^q\}_{r=1}^R$ by centroids $\hat{D}^q$;

Compute per-cluster sampling budget $b_r = \min \left\{ \frac{|\mathcal{C}_r^q|}{\sum_j |\mathcal{C}_j^q|}, \frac{v_r'}{\sum_j v_j'} \right\} \cdot B$;

for r in $\{1, \ldots, R\}$ **do** ▷ Decompress Centroids

    Initialize $S' = \{x\}$ by a random sample from $\mathcal{C}_r$;

    while $|S'| < b_r$ **do** ▷ Diverse Sampling

        $u = \arg\max_{x \in \mathcal{C}_r^q} \Delta(x, S')$;

        $S' = \{u\} \cup S'$;

    $S = S' \cup S$;

return S

We summarize our algorithm in Algorithm 5.1, which enables the clients to enjoy better computation efficiency than local training and better communication efficiency than the centralized sampling (e.g., MAK). **1) Computation efficiency**. Since our method only requires inference operations on the client device, which should be efficiently designed for the standard predictive functions of the device, and is training-free for the client, the major complexity of ECOS is on computing centroid coverage and is much lower than gradient-based algorithms whose complexity scales with the model size and training iterations. As computing the CC scores only requires the nearest centroid estimation and ranking, the filtering can be efficiently done. The total *time complexity* is $\mathcal{O}(C_\phi |D^p| + (d_e + 1)R|D^p|)$, dominated by the first term, where C_ϕ is the complexity of extracting features depending on the specific method. The second term $(d_e + 1)R|D^p|$ is for computing the pair-wise distances between $\hat{D}^p$ and $\hat{D}^q$ and estimating the nearest centroids per sample (or partitioning client data). In a brief comparison, the complexity of local T-iteration gradient-descent training could be

approximately $\mathcal{O}(TC_\phi|D^p|)$ which is much more expensive since typically $TC_\phi \gg d_e$. To complete the analysis, the *space complexity* is $\mathcal{O}(C'_\phi + (d+d_e)|D^p| + d_eR + |D^p|R)$ for the memory footprint of ϕ, the images (d) or features (d_e) of client and cloud centroid data, and the distance matrix. **2) Communication efficiency**. The downloading complexity will be $\mathcal{O}(d_eR)$ for R d_e-dimensional centroid features and the uploading complexity is $\mathcal{O}(R)$ including indexes of samples. Thus, the data that will be communicated between the client and cloud is approximately of $\mathcal{O}(d_eR + R)$ complexity in total. In comparison, downloading the whole open-source dataset by central sampling (e.g., MAK) requires $\mathcal{O}(d|D^q| + B)$ complexity. As $R < d_eR \ll d_e|D^q| \ll d|D^q|$, our method can significantly scale down the computation cost.

Privacy protection and accountant. When the cloud server is compromised by an attacker, uploading CC scores leak private information of the local data samples, for example, the presence of an identity [126]. To mitigate the privacy risk, we protect the uploaded scores by a Gaussian noise mechanism, i.e., $\tilde{v}_r = v_r + \mathcal{N}(0, \sigma^2)$, and account for the privacy cost in the notion of differential privacy (DP) [38]. DP quantifies the numerical influence of the absence of a private sample on $[v_1, \cdots, v_R]$, which is connected to the chance of exposing the sample to the attacker. To obtain a tight bound on the privacy cost, we utilize the tool of Rényi Differential Privacy (RDP) [99] and leverage the Poisson sampling to further amplify the privacy [167]. With the noise mechanism governed by σ, the resultant privacy cost in the sense of (ϵ, δ)-DP can be accounted as $\epsilon = \mathcal{O}(\gamma\sqrt{\log(1/\delta)}/\sigma)$ where γ is the Poisson subsampling rate and δ is a user-specified parameter. A larger ϵ implies higher risks of privacy leakage in the probability of δ. Formal proofs can be found in Section D.1.3.

5.4 Empirical Results

Datasets. We use datasets from two tasks: digit recognition and object recognition. Distinct from prior work [168], in our work, the open-source data contains samples out of the client's distribution. With the same classes as the client dataset, we assume open-source data are from different environments and therefore include different feature distributions, for ex-

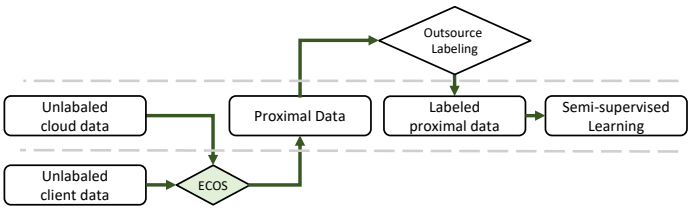
ample, DomainNet [110] and Digits [80]. DomainNet includes large-sized 244×244 everyday images on 6 domains and Digits consists of 28×28 digit images on 5 domains. Instead of using an overly large volume of data from a single domain like [168], we tailor each-domain subset to contain fewer images than standard digit datasets, for instance, MNIST with 7438 images, which was previously adopted in the distributed learning setting [80] and mitigates the hardness of collecting enormous data. In practice, collecting tens of thousands of images by a single client, e.g., 50000 images from MNIST domain, will be unrealistic. Similarly, DomainNet will be tailored to only include 10 classes with 2000-5000 images per domain.

Splits of client and cloud datasets. For Digits, we use one domain for the client and the rest domains for the cloud as open-source set. For DomainNet, we randomly select 50% samples from one domain for the client and leave the rest samples together with all other domains to the cloud. The difference of configurations for the two datasets is caused by their different domain gaps. Even without ID data, it is possible for Digits to transfer the knowledge across domains.

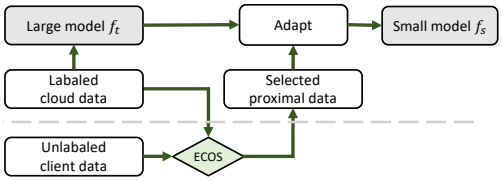
Baselines. For a fair comparison, we compare our method to baselines with the same sampling budget. Each experiment case is repeated for three times with seed $\{1, 2, 3\}$. We account for the privacy cost by Poisson-subsampling RDP [167] and translate the cost to the general privacy notion, (ϵ, δ)-DP when $\delta = 10^{-5}$. Here, we use the *random* sampling as a naive baseline. We also adopt a coreset selection method, K-Center [122], to select informative samples within the limited budget. Both baselines are perfectly private without accessing private information from clients. Details of hyper-parameters are deferred to Section D.2.1.

5.4.1 Evaluations on Training Outsourcing

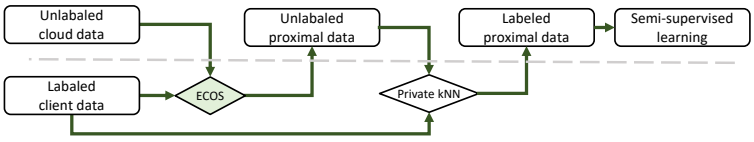
To demonstrate the general applicability of ECOS, we present three practical use case of outsourcing, categorized by the form of supervisions. The conceptual illustrations are in Fig. 5.2. Per use case, we train a model on partially labeled cloud data (outsourced training) and accuracy on the client test set are reported together with standard deviations.



(a) Selective manual labeling.



(b) Adaptive model compression.



(c) Automated client labeling.

Figure 5.2: Our method is applicable to various cloud training cases, where ECOS filters the open-source samples to improve the model performance trained on (a) manual, (b) pre-trained model (teacher), and (c) pseudo supervisions.

Table 5.2: Test accuracy (%) by selective labeling on Digits (top) and DomainNet (bottom).

Sampling Budget	Method	MNIST Acc ↑	MNIST $\epsilon\downarrow$	SVHN Acc ↑	SVHN $\epsilon\downarrow$	USPS Acc ↑	USPS $\epsilon\downarrow$	SynthDigits Acc ↑	SynthDigits $\epsilon\downarrow$	MNIST-M Acc ↑	MNIST-M $\epsilon\downarrow$	Average Acc ↑
2000	Ours	**97.3**±0.1	0.22	**68.7**±0.3	0.22	90.8±0.1	0.22	**84.4**±0.6	0.22	70.4±0.6	0.22	**82.3**
	K-Center	96.7±0.3	0.00	65.1±1.3	0.00	90.1±0.7	0.00	80.2±1.1	0.00	70.1±0.3	0.00	80.4
	Random	96.5±0.3	0.00	64.0±0.3	0.00	91.6±1.0	0.00	83.8±0.3	0.00	70.9±0.6	0.00	81.4
5000	Ours	**98.1**±0.2	0.22	**74.6**±1.0	0.22	**93.5**±0.3	0.22	**91.2**±0.4	0.22	74.5±0.5	0.22	**86.4**
	K-Center	97.9±0.2	0.00	72.3±0.6	0.00	92.7±0.9	0.00	89.6±0.3	0.00	74.0±0.5	0.00	85.3
	Random	97.6±0.3	0.00	70.0±0.3	0.00	93.0±0.6	0.00	89.7±0.4	0.00	73.9±0.7	0.00	84.8

Sampling Budget	Method	Clipart Acc ↑	Clipart $\epsilon\downarrow$	Infograph Acc ↑	Infograph $\epsilon\downarrow$	Painting Acc ↑	Painting $\epsilon\downarrow$	Quickdraw Acc ↑	Quickdraw $\epsilon\downarrow$	Real Acc ↑	Real $\epsilon\downarrow$	Sketch Acc ↑	Sketch $\epsilon\downarrow$	Average Acc ↑
1000	Ours	**88.4**±1.5	0.58	**52.6**±0.9	0.58	**90.4**±1.7	0.58	**84.3**±1.6	0.58	92.1±1.2	0.58	**87.2**±0.5	0.58	**82.5**
	K-Center	86.8±0.3	0.00	50.5±0.9	0.00	89.1±1.4	0.00	27.2±1.8	0.00	**92.5**±0.1	0.00	85.6±1.4	0.00	72.0
	Random	86.9±0.8	0.00	47.4±2.7	0.00	88.6±0.1	0.00	77.9±2.4	0.00	91.4±0.3	0.00	86.9±0.5	0.00	79.9
3000	Ours	93.2±0.4	0.58	**58.1**±0.6	0.58	92.5±1.1	0.58	**89.2**±0.9	0.58	**94.4**±0.2	0.58	92.8±0.2	0.58	**86.7**
	K-Center	93.5±1.1	0.00	56.3±0.3	0.00	**92.9**±0.3	0.00	60.5±8.7	0.00	94.1±0.2	0.00	92.1±0.7	0.00	81.6
	Random	92.5±0.6	0.00	53.6±1.4	0.00	91.7±0.8	0.00	86.1±0.4	0.00	93.5±0.3	0.00	93.0±0.2	0.00	85.1

We present the results in Tables 5.2 to 5.4 case by case, where we vary the domain of the client by columns. In each column, we highlight the best result unless the difference is not statistically significant.

Case 1: Selective manual labeling. We assume that the cloud will label the filtered in-domain samples by using a third-party label service, e.g., Amazon Mechanical Turk [107], or by asking the end clients for manual labeling. As the selected samples are non-private, they can be freely shared with a third parity. But the high cost of manual labeling or service

is the pain point, which should be carefully constrained within a finite *budget* of demanded labels.

Given a specified sampling budget, we compare the test accuracy of semi-supervised learning (FixMatch) on sampled data in Table 5.2. Since the ECOS tends to select in-distribution samples, it ease the transfer of cloud-trained model to the client data. On the Digits dataset, we find that our method attains more accuracy gains as budget increases, demonstrating that more effective labels are selected. On the DomainNet dataset, our method outperforms baselines on 5 out of 6 domains and is stable in most domains given a small budget (1000) and is superior on average. Given more budgets, the accuracy of all methods is improved, when our method is outstanding on Infograph and Quickdraw domains and is comparable to the baselines on other domains.

Table 5.3: Evaluate adaptive model compression on DomainNet by test accuracy, Acc (%) and differential-privacy cost ϵ.

Sampling		Clipart		Infograph		Painting		Quickdraw		Real		Sketch		Average
Budget	Method	Acc ↑	$\epsilon\downarrow$	Acc ↑	$\epsilon\downarrow$	Acc ↑	$\epsilon\downarrow$	Acc ↑	$\epsilon\downarrow$	Acc ↑	$\epsilon\downarrow$	Acc ↑	$\epsilon\downarrow$	Acc ↑
1000	Ours	82.9±0.7	0.58	**48.5**±2.7	0.58	**85.4**±1.0	0.58	**81.3**±2.5	0.58	91.4±0.6	0.58	**82.4**±1.1	0.58	**78.6**
	K-Center	81.2±0.9	0.00	44.6±0.9	0.00	84.5±2.2	0.00	41.7±1.9	0.00	**92.4**±0.5	0.00	80.1±2.1	0.00	70.8
	Random	**83.8**±0.7	0.00	44.4±2.1	0.00	83.8±1.6	0.00	76.3±2.2	0.00	90.1±0.6	0.00	80.1±0.7	0.00	76.4
3000	Ours	**90.6**±0.6	0.58	**51.4**±1.9	0.58	**89.6**±1.2	0.58	**87.6**±0.3	0.58	93.6±0.7	0.58	**88.4**±1.5	0.58	**83.5**
	K-Center	88.9±2.3	0.00	51.2±0.4	0.00	89.4±0.9	0.00	57.6±4.4	0.00	**94.5**±0.5	0.00	86.9±0.6	0.00	78.1
	Random	88.4±0.8	0.00	47.6±1.9	0.00	89.4±1.0	0.00	84.7±0.2	0.00	93.0±1.0	0.00	86.3±1.2	0.00	81.6

Case 2: Adaptive model compression. Due to the large volume of the open-source dataset, a larger model is desired for better capturing the various features, which however is so inefficient to fit into the resource-constrained client devices or specialize for the data distribution of the client. Confronting this challenge, model compression [17] is a conventional idea to forge a memory-efficient model by transferring knowledge from large models to small ones. Specifically, we first pre-train a large *teacher* model f_t on all cloud data by the supervised learning, assuming labels are available in advance. Still, we use an ImageNet-pretrained model to initialize the feature extractor $\phi(\cdot)$ of a *student* model f_s. Then the client will use the downloaded feature extractor ϕ to filter samples. Here, we utilize knowledge

distillation [52] to finetune the f_s with an additional classifier head upon the ϕ. On the selected samples, we train a linear classifier head for 30 epochs under the supervision of true labels and the teacher model f_t, and then fine-tune the full network f_s for 500 epochs. The major challenge comes from distributional biases between the multi-source open-source data and the client data. Leveraging ECOS, we may sample data near the client distribution and reduce the bias in the follow-up compression process.

We simulate the case on the large-sized image dataset, DomainNet, which is demanding for large-scale networks, e.g., ResNet50 here, to effectively learn the complicated features. Here, we compress ResNet50 into a smaller network, ResNet18, by using an adaptively selected subset of the cloud dataset. We omit the experiment for digit images where a large model may not be necessary for such a small image size. In Table 5.3, we present the test accuracy of compressed ResNet18 using 3000 samples from DomainNet in finetuning. With a small portion of privacy cost ($\epsilon < 0.6$), our method improves the accuracy on Clipart and Real domains against the baselines. Note that the model accuracy here is lower than label outsourcing in Table 5.2, and reason is that the supervisions from the larger models are just an approximation of the full dataset. Without using the full dataset for compression, the training can be completed fast and responsively on the demand of a client.

Table 5.4: Test accuracy (%) and privacy cost *epsilon* of client labeling on two datasets: Digits (top) and DomainNet (bottom).

Sampling Budget	Method	MNIST Acc ↑	MNIST ϵ ↓	SVHN Acc ↑	SVHN ϵ ↓	USPS Acc ↑	USPS ϵ ↓	SynthDigits Acc ↑	SynthDigits ϵ ↓	MNIST-M Acc ↑	MNIST-M ϵ ↓	Average Acc ↑
5000	Ours	**84.2**±2.3	5.35	47.9±3.1	5.32	**86.1**±1.0	5.35	68.6±1.6	5.35	**58.4**±1.9	5.35	**69.0**
	K-Center	81.9±3.4	5.34	48.4±1.2	5.33	82.1±1.2	5.34	69.4±1.9	5.34	55.4±2.0	5.34	67.4
	Random	81.8±4.1	5.34	45.3±3.0	5.29	81.2±2.3	5.34	65.9±2.7	5.34	55.5±2.6	5.34	65.9

Sampling Budget	Method	Clipart Acc ↑	Clipart ϵ ↓	Infograph Acc ↑	Infograph ϵ ↓	Painting Acc ↑	Painting ϵ ↓	Quickdraw Acc ↑	Quickdraw ϵ ↓	Real Acc ↑	Real ϵ ↓	Sketch Acc ↑	Sketch ϵ ↓	Average Acc ↑
3000	Ours	33.2±5.9	4.46	**23.8**±2.2	3.50	**47.4**±3.3	4.51	**39.8**±7.7	4.87	62.9±1.9	4.92	**51.7**±1.2	4.94	**43.2**
	K-Center	**39.3**±3.6	4.57	18.2±2.7	3.61	46.6±2.5	4.52	36.1±2.9	4.94	**63.8**±1.9	4.96	47.7±2.8	4.96	42.0
	Random	30.7±2.2	4.41	21.6±4.9	3.43	44.0±3.1	4.53	39.0±5.6	4.82	59.9±3.6	4.75	47.2±3.4	4.94	40.4

Case 3: Automated client labeling. When the client obtained labeled samples, for example, photos labeled by phone users, the cloud only needs to collect an unlabeled public

dataset. Therefore, we may automate the labeling process leveraging the client supervision knowledge to reduce cost or users' efforts on manual labeling. To be specific, we let the client generate pseudo labels for the cloud data based on their neighbor relation, as previously studied by [168] (private kNN). However, the private kNN assumes that the client and the cloud follow the same distribution, which weakens its applicability confronting the heterogeneity and the large scale of open-source data. Thus, we utilize ECOS as a pre-processing of open-source data before being labeled by private kNN. Therefore, we have two rounds of communication for transferring client knowledge: proximal-data sampling by the ECOS and client labeling by the private kNN [168]. To compose the privacy costs from these two steps, we utilize the analytical moment accountant technique to get a tight privacy bound [144]. Interested readers can refer to Section D.1.1 for a brief introduction to private kNN and our implementations.

In experiments, we use the state-of-the-art private pseudo-labeling method, private kNN [168], to label the subsampled open-source data with the assistance from the labeled client dataset. To reduce the sensitivity of private kNN w.r.t. the threshold, we instead enforce the number of the selected labels to be 600 and balance the sizes by selecting the top-60 samples with the highest confidence per class. Then, we adopt the popular semi-supervised learning method, FixMatch [127], to train the classifier. In Table 5.4, we report the results when cloud features are distributionally biased from the client ones but they share the same class set. For each domain choice of client data, we will use the other domains as the cloud dataset. Distinct from prior studies, e.g., in [168] or [109], we assume 80-90% of the cloud data are out of the client distribution and are heterogeneously aggregated from different domains, casting greater challenges in learning. On the Digits, we eliminate all ID data from the cloud set to harden the task. Both on Digits and DomainNet datasets, our method consistently outperforms the two sampling baselines under the similar privacy costs. The variance of privacy costs is mainly resulted from the actual sampling sizes. Though simply adopting K-Center outperforms the random sampling, it still presents larger gaps compared to our method in

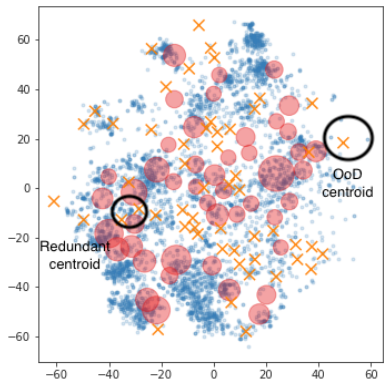


Figure 5.3: Demonstration of the centroids qualified by private CC. We use blue dots to represent the client data from Real domain of DomainNet. For the data of the cloud domains, larger circles represent centroids with higher CC and orange crosses are rejected OoD centroids.

multiple domains. For instance, in Quickdraw domain, given 5000 sampling budget, the K-Center method performs poorly and is even worse than a random sampling.

5.4.2 Qualitative Study

To better understand the proposed method, we conduct a series of qualitative studies on client labeling. We use two DomainNet datasets as a representative benchmark in the studies. 1) **Ablation study**. In Table 5.5, we conduct ablation studies to evaluate the effect of different objectives introduced by ECOS, following the client labeling benchmark with a 3000 sample budget. Without proximity and diversity objectives, we let the baseline be the naive random sampling. We first include the proximity objective, where we greedily select samples from top-scored clusters until the budget is fulfilled. However, we find that the naive proximity objective results in a quite negative effect compared to the random baseline. The failure can be attributed to the nature of clustering that will include more similar samples, namely lacking diversity. When diversity is encouraged and combined with the proximal votes, we find the performance is improved significantly in multiple domains and on average. Now with diversity, the proximity objective can further improve the sampling in Painting, Quickdraw, and Sketch domains significantly. 2) **Visualize cluster selection**. In Fig. 5.3, we demonstrate that the CC can effectively reject OoD centroids. Also, it is interesting to

observe that when multiple centroids are distributed closely, then they will compete with each other and reject the redundant ones consequently. 3) **Efficiency**. In Section 5.3, we studied the communication efficiency theoretically. Empirically, ECOS only need to upload 100 bytes of the CC scores in all experiments, while traditional outsourcing needs to upload the dataset, which is about 198MB for the lowest load in DomainNet (50% of Sketch domain data for client). Even counting the download load, ECOS only need to download 45MB of the pre-trained ResNet18 feature extractor together with 51KB data of centroid features, which is much less than data uploading. More detailed evaluation of the efficiency is placed at Section D.2.4.

Table 5.5: Ablation study of the proposed method on DomainNet. Test accuracy of the client labeling case is reported.

Proximity	Diversity	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Average
✗	✗	30.7±2.2	21.6±4.9	44.0±3.1	39.0±5.6	59.9±3.6	47.2±3.4	40.4
✓	✗	25.5 ±5.7	21.1±0.5	41.4±5.3	31.3±1.3	60.3±2.3	31.9±2.9	35.2
✗	✓	**39.3**±3.6	18.2±2.7	46.6±2.5	36.1±2.9	**63.8**±1.9	47.7±2.8	42.0
✓	✓	33.2±5.9	**23.8**±2.2	**47.4**±3.3	**39.8**±7.7	**62.9**±1.9	**51.7**±1.2	**43.2**

CHAPTER 6

OVERVIEW

In this chapter, we summarize our contributions to data-centric privacy-preserving learning.

Centralized learning. When a privacy budget is provided for a certain learning task, one has to carefully schedule the privacy usage throughout the learning process. Uniformly scheduling the budget has been widely used in literature whereas increasing evidence suggests that dynamical schedules could empirically outperform uniform ones. Our theoretical work on dynamic differentially-private learning provided a principled analysis of the problem of optimal budget allocation and connected the advantages of dynamic schedules to both the loss structure and the learning behavior. We further validated our results through empirical studies.

Federated learning with heterogeneous devices. In this work, we proposed a novel federated learning approach for in-situ and on-demand customization to address challenges arising from resource heterogeneity and inference dynamics. We proposed a Split-Mix strategy that efficiently transfer clients' knowledge to collaboratively learn a customizable model. Extensive experiments demonstrate the effectiveness of the principle in adjusting model widths and robustness when much fewer parameters are used compared to baselines.

Outsourcing training without uploading data. In this work, we explore the possibility of outsourcing model training without access to client data. To reconcile the data shortage from the target domain, we propose to find a surrogate dataset from the source-agnostic public dataset. We find that the heterogeneity of the open-source data greatly compromises the performance of trained models. To tackle this practical challenge, we propose a collaborative sampling solution, ECOS, that can efficiently and effectively filter open-source samples and thus benefits follow-up learning tasks. We envision this work as a milestone for private and efficient outsourcing from low-power and cost-effective end devices. We also recognize open questions of the proposed solution for future studies. For example, the public dataset may require additional data processing, e.g., aligning and cropping for

improved prediction accuracy. In our empirical studies, we only consider the computer vision tasks, though no assumption was made on the data structures. We expect the principles to be adapted to other data types with minimal effort. More data types, including tabular and natural-language data, will be considered in the follow-up works.

BIBLIOGRAPHY

[1] General data protection regulation.

[2] Facebook–cambridge analytica data scandal. *Wikipedia*, October 2022.

[3] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM.

[4] Accountability Act. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

[5] P. S. Aisen, S. Andrieu, C. Sampaio, M. Carrillo, Z. S. Khachaturian, B. Dubois, H. H. Feldman, R. C. Petersen, E. Siemers, R. S. Doody, S. B. Hendrix, M. Grundman, L. S. Schneider, R. J. Schindler, E. Salmon, W. Z. Potter, R. G. Thomas, D. Salmon, M. Donohue, M. M. Bednar, J. Touchon, and B. Vellas. Report of the task force on designing clinical trials in early (predementia) ad. *Neurology*, 76(3):280–286, January 2011.

[6] Zeyuan Allen-Zhu, Yuanzhi Li, and Zeyuan Allen-Zhu. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv:2012.09816 [cs.LG]*, December 2020.

[7] Noga Alon, Raef Bassily, and Shay Moran. Limits of private learning with access to public data. *Advances in neural information processing systems*, 32, 2019.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. January 2017.

[9] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403, July 2018.

[10] Jeff Barnes. Azure machine learning. *Microsoft Azure Essentials. 1st ed, Microsoft*, 2015.

[11] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473, October 2014.

[12] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Guha Thakurta. Private stochastic convex optimization with optimal rates. In *Advances in Neural Information Processing Systems 32*, pages 11282–11291. Curran Associates, Inc., 2019.

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

[14] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[15] Ekaba Bisong. Google cloud machine learning engine (cloud mle). In *Building machine learning and deep learning models on google cloud platform*, pages 545–579. Springer, 2019.

[16] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA, 2017. ACM.

[17] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 535–541, New York, NY, USA, August 2006. Association for Computing Machinery.

[18] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography*, volume 9985, pages 635–658. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[19] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv:1812.01097 [cs, stat]*, December 2019.

[20] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[21] Fei Chen, Tao Xiang, Xinyu Lei, and Jianyong Chen. Highly efficient linear regression outsourcing to a cloud. *IEEE transactions on cloud computing*, 2(4):499–508, 2014.

[22] Junhong Cheng, Wenyan Liu, Xiaoling Wang, Xingjian Lu, Jing Feng, Yi Li, and Chaofan Duan. Adaptive distributed differential privacy with sgd. *Workshop on Privacy-Preserving Artificial Intelligence*, page 6, 2020.

[23] William J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. Combinatorial optimization. *Oberwolfach Reports*, 5(4):2875–2942, 2009.

[24] Rachel Cummings, Sara Krehbiel, Kevin A Lai, and Uthaipon Tantipongpipat. Dif-

ferential privacy for growing databases. In *Advances in Neural Information Processing Systems 31*, pages 8864–8873. Curran Associates, Inc., 2018.

[25] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[27] Damien Desfontaines and Balázs Pejó. Sok: Differential privacies. *arXiv:1906.01337 [cs]*, June 2019.

[28] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

[29] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, June 2020.

[30] Jiahua Dong, Yang Cong, Gan Sun, Zhen Fang, and Zhengming Ding. Where and how to transfer: Knowledge aggregation-induced transferability perception for unsupervised domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[31] Jiahua Dong, Yang Cong, Gan Sun, Zhen Fang, and Zhengming Ding. Where and how to transfer: Knowledge aggregation-induced transferability perception for unsupervised domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[32] Jiahua Dong, Yang Cong, Gan Sun, Bineng Zhong, and Xiaowei Xu. What can be transferred: Unsupervised domain adaptation for endoscopic lesions segmentation. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 4022–4031, June 2020.

[33] Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, September 2019.

[34] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 1–12. Springer Berlin Heidelberg, 2006.

[35] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications*

of Models of Computation, Lecture Notes in Computer Science, pages 1–19. Springer Berlin Heidelberg, 2008.

[36] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 214–226, Cambridge, Massachusetts, January 2012. Association for Computing Machinery.

[37] Cynthia Dwork, Alan Karr, Kobbi Nissim, and Lars Vilhuber. On privacy in the age of covid-19. *Journal of Privacy and Confidentiality*, 10(2), June 2020.

[38] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, Lecture Notes in Computer Science, pages 265–284. Springer Berlin Heidelberg, 2006.

[39] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013.

[40] Eric Enge. Mobile vs. desktop usage in 2020. https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage, March 2021.

[41] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems*, June 2020.

[42] Vitaly Feldman, Tomer Koren, and Kunal Talwar. Private stochastic convex optimization: optimal rates in linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 439–449, New York, NY, USA, June 2020. Association for Computing Machinery.

[43] Hao-Zhe 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. *arXiv:2011.09757*, 2021.

[44] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics. Journal of the International Biometric Society*, 21:768–769, 1965.

[45] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA, 2015. ACM.

[46] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backprop-

agation. In *International Conference on Machine Learning*, pages 1180–1189. PMLR, June 2015.

[47] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv:1406.2661 [cs, stat]*, June 2014.

[48] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*, pages 1929–1938. PMLR, July 2018.

[49] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, November 2020.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition*, 2016.

[52] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531 [cs, stat]*, March 2015.

[53] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 603–618, New York, NY, USA, 2017. ACM.

[54] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1989–1998. PMLR, July 2018.

[55] Junyuan Hong, Jeffrey Kaye, Hiroko H. Dodge, and Jiayu Zhou. Detecting mci using real-time, ecologically valid data capture methodology: How to improve scientific rigor in digital biomarker analyses. *Alzheimer's & Dementia*, 16(S5):e044371, 2020.

[56] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Federated robustness propagation: Sharing adversarial robustness in federated learning. *arXiv:2106.10196 [cs, stat]*, June 2021.

[57] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Learning model-based privacy protection under budget constraints. In *AAAI*, page 9, 2021.

[58] Junyuan Hong, Zhangyang Wang, and Jiayu Zhou. Dynamic privacy budget allocation improves data efficiency of differentially private gradient descent. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, pages 11–35, New York, NY, USA, June 2022. Association for Computing Machinery.

[59] 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, pages 617–627, New York, NY, USA, August 2021. Association for Computing Machinery.

[60] Gao Huang, Danlu Chen, Tianhong Li, and Felix Wu. Multi-scale dense networks for resource efficient image classification. *International Conference on Learning Representations*, page 14, 2018.

[61] Xixi Huang, Jian Guan, Bin Zhang, Shuhan Qi, Xuan Wang, and Qing Liao. Differentially private convolutional neural networks with adaptive gradient descent. In *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, pages 642–648, June 2019.

[62] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[63] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[64] Prateek Jain, Dheeraj Nagaraj, and Praneeth Netrapalli. Making the last iterate of sgd information theoretically optimal. In *Conference on Learning Theory*, pages 1752–1755, June 2019.

[65] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Improving contrastive learning on imbalanced data via open-world sampling. In *Advances in Neural Information Processing Systems*, May 2021.

[66] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning | The MIT Press*. MIT Press, 2008.

[67] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

[68] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and

proximal-gradient methods under the polyak-łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 795–811, Cham, 2016. Springer International Publishing.

[69] Jeffrey A. Kaye, Shoshana A. Maxwell, Nora Mattek, Tamara L. Hayes, Hiroko Dodge, Misha Pavel, Holly B. Jimison, Katherine Wild, Linda Boise, and Tracy A. Zitzelberger. Intelligent systems for assessing aging changes: Home-based, unobtrusive, and continuous assessment of aging. *The Journals of Gerontology: Series B*, 66B(suppl_1):i180–i190, July 2011.

[70] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *Proceedings of the 25th Annual Conference on Learning Theory, COLT '12*, page 40, 2012.

[71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *the 3rd International Conference for Learning Representations*, San Diego, CA, 2015.

[72] Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization:distributed optimization beyond the datacenter. *arXiv:1511.03575 [cs, math]*, November 2015.

[73] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[74] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[75] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1656–1665, New York, NY, USA, 2018. ACM.

[76] Xinyu Lei, Xiaofeng Liao, Tingwen Huang, Huaqing Li, and Chunqiang Hu. Outsourcing large matrix inversion computation to a public cloud. *IEEE Transactions on cloud computing*, 1(1):1–1, 2013.

[77] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv:1910.03581 [cs, stat]*, October 2019.

[78] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Conference on Systems and Machine Learning Foundation (MLSys)*, April 2020.

[79] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *International Conference on Learning Representations*,

September 2019.

[80] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *International Conference on Learning Representations*, September 2020.

[81] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. *ICML*, October 2020.

[82] Jian Liang, Dapeng Hu, Yunbo Wang, Ran He, and Jiashi Feng. Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *ArXiv*, 2020.

[83] Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, et al. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pages 731–737, 2020.

[84] Ming Lin, Pinghua Gong, Tao Yang, Jieping Ye, Roger L. Albin, and Hiroko H. Dodge. Big data analytical approaches to the nacc dataset: Aiding preclinical trial enrichment. *Alzheimer Disease & Associated Disorders*, 32(1):18–27, 2018.

[85] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems*, June 2020.

[86] Aishan Liu, Shiyu Tang, Xianglong Liu, Xinyun Chen, Lei Huang, Zhuozhuo Tu, Dawn Song, and Dacheng Tao. Towards defending multiple adversarial perturbations via gated batch normalization. *arXiv:2012.01654 [cs]*, December 2020.

[87] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *arXiv:1701.00299 [cs, stat]*, March 2018.

[88] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[89] Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Steven Wu. Leveraging public data for practical private query release. In *International conference on machine learning*, pages 6968–6977. PMLR, 2021.

[90] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of international conference on computer vision (ICCV)*, December 2015.

[91] Mohammad Reza Loghmani, Markus Vincze, and Tatiana Tommasi. Positive-unlabeled learning for open set domain adaptation. *Pattern Recognition Letters*, 136, June 2020.

[92] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. *arXiv:1705.10667 [cs]*, December 2018.

[93] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3384–3393. PMLR, July 2018.

[94] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[95] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1860, 2019.

[96] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, April 2017.

[97] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[98] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, February 2018.

[99] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, Santa Barbara, CA, USA, August 2017. IEEE.

[100] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, May 2017.

[101] Yurii Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, August 2006.

[102] 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.

[103] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

[104] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems 29*, pages 3387–3395. Curran Associates, Inc., 2016.

[105] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R. Rabiee. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):54–66, January 2020.

[106] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[107] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[108] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *International conference on learning representations*, 2016.

[109] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *International Conference on Learning Representations*, February 2018.

[110] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[111] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *International Conference on Learning Representations*, September 2019.

[112] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv:1710.06924 [cs]*, November 2017.

[113] Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv:1908.07643 [cs, stat]*, October 2019.

[114] B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computa-*

tional Mathematics and Mathematical Physics, 3(4):864–878, January 1963.

[115] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, January 1964.

[116] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, January 1999.

[117] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[118] Sashank J. Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323, June 2016.

[119] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[120] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality*, 2012.

[121] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg, September 2010. Springer-Verlag.

[122] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International conference on learning representations*, 2018.

[123] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, May 2014.

[124] Shai Shalev-Shwartz, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *Proceedings of the 22nd Annual Conference on Learning Theory, COLT '09*, page 11, 2009.

[125] Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury. Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv:2107.10663 [cs, stat]*, July 2021.

[126] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*

(SP), pages 3–18, May 2017.

[127] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, November 2020.

[128] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems 28*, pages 3025–3033. Curran Associates, Inc., 2015.

[129] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019.

[130] Om Thakkar, Galen Andrew, and H. Brendan McMahan. Differentially private learning with adaptive clipping. *arXiv:1905.03871 [cs, stat]*, May 2019.

[131] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[132] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *International Conference on Learning Representations*, September 2019.

[133] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[134] B. Vellas, R. Bateman, K. Blennow, G. Frisoni, K. Johnson, R. Katz, J. Langbaum, D. Marson, R. Sperling, A. Wessels, S. Salloway, R. Doody, and P. Aisen. Endpoints for pre-dementia ad trials: A report from the eu/us/ctad task force. *The journal of prevention of Alzheimer's disease*, 2(2):128–135, June 2015.

[135] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.

[136] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *ICLR 2019 Workshop on AI for social good*, December 2018.

[137] Christina Wadsworth, Francesca Vera, and Chris Piech. Achieving fairness through

adversarial learning: an application to recidivism prediction. *arXiv:1807.00199 [cs, stat]*, June 2018.

[138] Di Wang, Changyou Chen, and Jinhui Xu. Differentially private empirical risk minimization with non-convex loss functions. In *International Conference on Machine Learning*, pages 6526–6535, May 2019.

[139] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems 30*, pages 2722–2731. Curran Associates, Inc., 2017.

[140] Haotao Wang, Tianlong Chen, Shupeng Gui, Ting-Kuei Hu, Ji Liu, and Zhangyang Wang. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. *Advances in Neural Information Processing Systems*, November 2020.

[141] Haotao Wang, Chaowei Xiao, Jean Kossaifi, Zhiding Yu, Anima Anandkumar, and Zhangyang Wang. Augmax: Adversarial composition of random augmentations for robust training. *arXiv:2110.13771 [cs]*, October 2021.

[142] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[143] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. *ECCV*, July 2018.

[144] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235, April 2019.

[145] Michael W. Weiner, Dallas P. Veitch, Paul S. Aisen, Laurel A. Beckett, Nigel J. Cairns, Robert C. Green, Danielle Harvey, Clifford R. Jack, William Jagust, Enchi Liu, John C. Morris, Ronald C. Petersen, Andrew J. Saykin, Mark E. Schmidt, Leslie Shaw, Li Shen, Judith A. Siuciak, Holly Soares, Arthur W. Toga, and John Q. Trojanowski. The alzheimer's disease neuroimaging initiative: A review of papers published since its inception. *Alzheimer's & Dementia*, 9(5):e111–e194, September 2013.

[146] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[147] Martin A Weiss and Kristin Archick. US-EU data privacy: from safe harbor to privacy shield, 2016.

[148] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based

analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1307–1322, New York, NY, USA, 2017. ACM.

[149] Zhenyu Wu, Haotao Wang, Zhaowen Wang, Hailin Jin, and Zhangyang Wang. Privacy-preserving deep action recognition: An adversarial learning framework and a new dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[150] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. *arXiv:1711.08393 [cs]*, January 2019.

[151] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *International Conference on Learning Representations*, December 2019.

[152] Yun Xie, Peng Li, Chao Wu, and Qiuling Wu. Differential privacy stochastic gradient descent with adaptive privacy budget allocation. In *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pages 227–231, January 2021.

[153] Mengwei Xu, Jiawei Liu, Yuanqiang Liu, Felix Xiaozhu Lin, Yunxin Liu, and Xuanzhe Liu. A first look at deep learning apps on smartphones. *arXiv:1812.05448 [cs]*, January 2021.

[154] Yixing Xu, Yunhe Wang, Hanting Chen, Kai Han, Chunjing XU, Dacheng Tao, and Chang Xu. Positive-unlabeled compression on the cloud. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[155] Zhiying Xu, Shuyu Shi, Alex X. Liu, Jun Zhao, and Lin Chen. An adaptive and fast convergent approach to differentially private deep learning. *the Proceedings of IEEE International Conference on Computer Communications*, 2020.

[156] Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, Anpeng Wu, Fengda Zhang, Ziqi Tan, Kun Kuang, Chao Wu, Fei Wu, Jingren Zhou, and Hongxia Yang. Edge-cloud polarization and collaboration: A comprehensive survey. *arXiv:2111.06061 [cs]*, November 2021.

[157] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Gradient perturbation is underrated for differentially private convex optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3117–3123, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization.

[158] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv:1812.08928 [cs]*, December 2018.

[159] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. Differentially private model publishing for deep learning. *proceedings of 40th IEEE Symposium on Security and Privacy*, April 2019.

[160] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146 [cs]*, June 2017.

[161] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, pages 335–340, New York, NY, USA, December 2018. Association for Computing Machinery.

[162] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: Differentially private model fitting using genetic algorithms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 665–676, New York, NY, USA, 2013. ACM.

[163] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[164] Xinyue Zhang, Jiahao Ding, Maoqiang Wu, Stephen T. C. Wong, Hien Van Nguyen, and Miao Pan. Adaptive privacy preserving deep learning algorithms for medical data. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1169–1178, 2021.

[165] Yingxue Zhou, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Arindam Banerjee. Private stochastic non-convex optimization: Adaptive algorithms and tighter generalization bounds. *arXiv:2006.13501 [cs, stat]*, August 2020.

[166] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems 32*, pages 14774–14784. Curran Associates, Inc., 2019.

[167] Yuqing Zhu and Yu-Xiang Wang. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*, pages 7634–7642. PMLR, May 2019.

[168] Yuqing Zhu, Xiang Yu, Manmohan Chandraker, and Yu-Xiang Wang. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11854–11862, 2020.

[169] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *Proceedings of the 38th International Conference on Machine Learning*, June 2021.

APPENDIX A

DYNAMIC PRIVACY BUDGET ALLOCATION

A.1 Social Impact

The wide usage of personal data in training machine learning has led to huge successes in many application domains but is also accompanied by rising concerns on privacy protection due to the sensitive information in the data. The development of privacy-preserving algorithms has become one of critical research areas of machine learning, in which the key challenge is to train high performance models under the constraint of a given privacy budget, or how much sensitive information can be accessed during the training phase. Differential privacy provided a principled framework to quantify the privacy budget, under which researchers proposed various schemes to schedule the budget usage during a learning process, yet there is a lack of systematical studies on when and why some schedules are better than other. Our efforts in this paper are among the first to study and compare the effectiveness of these schedules from a rigorous optimization perspective. Our theoretical results can benefit any privacy-preserving machine learning practitioners to efficiently and effectively choose proper privacy schedules tailored to their learning tasks.

A.2 Comparison of algorithms

Here we elaborate the meaning of algorithm names in Table 2.1. Asymptotic upper bounds are achieved when sample size N approaches infinity. Both R and $R_{\epsilon,\delta}$ with $R_{\epsilon,\delta} < R$ are the privacy budgets of corresponding algorithms. Specifically, $R_{\epsilon,\delta} = \epsilon^2/\ln(1/\delta) < R$ when the private algorithm is (ϵ, δ)-DP with $\epsilon \leq 2\ln(1/\delta)$.

PGD+Adv. Adv denotes the Advanced Composition method [11]. The method assumes that loss function is 1-strongly convex which implies the PL condition and optimized variable is in a convex set of diameter 1 w.r.t. l_2 norm.

PGD+MA and the adjusted-utility version. MA denotes the Moment Accountant [3] which improve the composed privacy bound versus the Advanced Composition. The im-

provement on privacy bound lead to a enhanced utility bound, as a result.

PGD+Adv+BBImp. The dynamic method assumes that the loss is 1-strongly convex and data comes in stream with $n \leq N$ samples at each round. Their utility upper bound is achieved at some probability p with any positive c.

Adam+MA. The authors prove a convergence bound for the gradient norms which is extended to loss bound by using PL condition. They also presents the results for AdaGrad and GD which are basically of the same upper bound. Out theorems improve their bound by using the recursive derivation based on the PL condition, while their bound is a simple application of the condition on the gradient norm bound.

GD, Non-Private. This method does not inject noise into gradients but limit the number of iterations. With the bound, we can see that our utility bound are optimal with dynamic schedule.

GD+zCDP. We discussed the static and dynamic schedule for the gradient descent method where the dynamic noise influence is the key to tighten the bound.

Momentum+zCDP. Different from the GD+zCDP, momentum methods will have two phase of utility upper bound. When T is small than some positive constant $\hat{T}$, the bound is as tight as the non-private one. Afterwards, the momentum has a bound degraded as the GD bound.

A.3 Preliminaries

A.3.1 Privacy

Lemma 2 (Composition & Post-processing)**.** *Let two mechanisms be $M : \mathcal{D}^n \to \mathcal{Y}$ and $M' : \mathcal{D}^n \times \mathcal{Y} \to \mathcal{Z}$. Suppose M satisfies (ρ_1, a)-zCDP and $M'(\cdot, y)$ satisfies (ρ_2, a)-zCDP for $\forall y \in \mathcal{Y}$. Then, mechanism $M'' : \mathcal{D}^n \to \mathcal{Z}$ (defined by $M''(x) = M'(x, M(x))$) satisfies $(\rho_1 + \rho_2)$-zCDP.*

Definition 7 (Sensitivity). *The sensitivity of a gradient query ∇_t to the dataset $\{x_i\}_{i=1}^N$ is*

$$\begin{aligned}\Delta_2(\nabla_t) &= \max_n \left\| \frac{1}{N} \sum\nolimits_{j=1, j\neq n}^{N} \nabla_t^{(j)} - \frac{1}{N} \sum\nolimits_{j=1}^{N} \nabla_t^{(j)} \right\|_2 \\ &= \frac{1}{N} \max_n \left\| \nabla_t^{(n)} \right\|_2 \end{aligned} \tag{A.1}$$

where $\nabla_t^{(n)}$ denotes the gradient of the n-th sample.

Lemma 3 (Gaussian mechanism [18]). *Let $f : \mathcal{D}^n \to \mathcal{Z}$ have sensitivity Δ. Define a randomized algorithm $M : \mathcal{D}^n \to \mathcal{Z}$ by $M(x) \leftarrow f(x) + \mathcal{N}(0, \Delta^2\sigma^2 I)$. Then M satisfies $\frac{1}{2\sigma^2}$-zCDP.*

Lemma 4 ([18]). *If M is a mechanism satisfying ρ-zCDP, then M is $(\rho+2\sqrt{\rho \ln(1/\delta)}, \delta)$-DP for any $\delta > 0$.*

By solving $\rho + 2\sqrt{\rho \ln(1/\delta)} = \epsilon$, we can get $\rho = \epsilon + 2\ln(1/\delta) + 2\sqrt{\ln(1/\delta)(\epsilon + \ln(1/\delta)}$.

A.4 Proofs

Detailed proofs please refer to [58].

APPENDIX B

FEDERATED ADVERSARIAL DEBIASING FOR FAIR AND TRANSFERABLE REPRESENTATIONS

B.1 Proofs

Proof of Theorem 9. Without loss of generality, we can write $\sigma_g \xi_t + G\sigma_t \nu_t$ as $\tilde{\sigma}_t \zeta_t$ where $\tilde{\sigma}_t \triangleq \sqrt{\sigma_g^2 + (G\sigma_t)^2}$ and ζ_t is a random vector with $\mathbb{E}\zeta_t = 0$ and $\mathbb{E}\left\|\zeta_t\right\|^2 \leq D$. Therefore, we replace ν_t by ζ_t and σ_t^2 by $\tilde{\sigma}_t^2/G^2 = \sigma_g^2/G^2 + \sigma_t^2$. Now, we only need to update $U_3(\sigma, T)$ as

$$\begin{aligned}
\tilde{U}_3 &= \frac{1}{G^2}\sum\nolimits_{t=1}^{T} \gamma^{T-t} \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum\nolimits_{i=1}^{t} \beta^{2(t-i)} \tilde{\sigma}_i^2 \\
&= \sum\nolimits_{t=1}^{T} \gamma^{T-t} \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum\nolimits_{i=1}^{t} \beta^{2(t-i)} (\frac{1}{G^2}\sigma_g^2 + \sigma_t^2) \\
&= U_3^g + U_3
\end{aligned}$$

where we define

$$U_3^g \triangleq \frac{1}{G^2}\sigma_g^2 \sum\nolimits_{t=1}^{T} \gamma^{T-t} \frac{(1-\beta)^2}{(1-\beta^t)^2} \sum\nolimits_{i=1}^{t} \beta^{2(t-i)}.$$

We can upper bound U_3^g by

$$\begin{aligned}
U_3^g &= \frac{1}{G^2}\sigma_g^2 \sum\nolimits_{t=1}^{T} \gamma^{T-t} \frac{(1-\beta)^2}{(1-\beta^t)^2} \frac{1-\beta^{2t}}{1-\beta^2} \\
&= \frac{1}{G^2}\sigma_g^2 \sum\nolimits_{t=1}^{T} \gamma^{T-t} \frac{1-\beta}{1-\beta^t} \frac{1+\beta^t}{1+\beta} \\
&\leq \frac{1}{G^2}\sigma_g^2 \sum\nolimits_{t=1}^{T} \gamma^{T-t} \\
&\leq \frac{1}{G^2}\sigma_g^2 \frac{1}{1-\gamma} \\
&= \frac{1}{G^2}\kappa\sigma_g^2.
\end{aligned}$$

Combine with the factors of U_3 in the PGD bounds:

$$\alpha R' U_3^g \leq \frac{\alpha R'}{G^2}\kappa\sigma_g^2 = \frac{\alpha R'}{G^2}\kappa\sigma_g^2 = \frac{D\sigma_g^2}{2\mu n^2 (f(\theta_1) - f(\theta^*))} \leq \frac{D\sigma_g^2}{2\mu N^2 R (f(\theta_1) - f(\theta^*))}.$$

□

Proof of Theorem 13. Substitute $D^*_{\alpha_1,\alpha_2}(z)$ into Eq. (3.4):

$$\begin{aligned}
\tilde{\mathbf{D}}_{p_1,p_2} &= \mathbb{E}_{p_1}[\alpha_1 \log \frac{\alpha_1 p_1(z)}{\alpha_1 p_1(z) + \alpha_2 p_2(z)}] \\
&\quad + \mathbb{E}_{p_2}[\alpha_2 \log \frac{\alpha_2 p_2(z)}{\alpha_1 p_1(z) + \alpha_2 p_2(z)}] \\
&= \alpha_1 \mathrm{KL}\left[p_1 \middle| \frac{\alpha_1 p_1 + \alpha_2 p_2}{\alpha_1 + \alpha_2}\right] \\
&\quad + \alpha_2 \mathrm{KL}\left[p_2 \middle| \frac{\alpha_1 p_1 + \alpha_2 p_2}{\alpha_1 + \alpha_2}\right] \\
&\quad + \alpha_1 \log \alpha_1 + \alpha_2 \log \alpha_2 \\
&\quad + (\alpha_1 + \alpha_2) \log(\alpha_1 + \alpha_2) \\
&\geq \alpha_1 \log \alpha_1 + \alpha_2 \log \alpha_2 \\
&\quad + (\alpha_1 + \alpha_2) \log(\alpha_1 + \alpha_2)
\end{aligned}$$

where the last inequality is from the non-negative property of KL divergence.

Note when $p_1 = p_2$, both KL divergence is 0. Thus, we can conclude that $p_1 = p_2$ is the sufficient condition. □

Proof of Theorem 14. For the ease of derivation, we assume α_1 and α_2 are normalized s.t. $\alpha_1 + \alpha_2 = 1$. From $|\log p_1(x) - \log p_2(x)| \leq \epsilon$, we can get

$$\begin{aligned}
e^{-\epsilon} \leq p_1(x)/p_2(x) \leq e^{\epsilon}, \\
e^{-\epsilon} \leq p_2(x)/p_1(x) \leq e^{\epsilon}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathrm{KL}\left[p_1 \,|\alpha_1 p_1 + \alpha_2 p_2\right] &= \int_x p_1 \log\left(\frac{p_1}{\alpha_1 p_1 + \alpha_2 p_2}\right) \\
&\leq \int_x p_1 \log\left(\frac{1}{\alpha_1 + \alpha_2 e^{-\epsilon}}\right) \\
&= \epsilon - \log(\alpha_1 e^{\epsilon} + \alpha_2).
\end{aligned}$$

Similarly,

$$\begin{aligned}\mathrm{KL}\left[p_2 \,|\alpha_1 p_1 + \alpha_2 p_2\right] &= \int_x p_2 \log\left(\frac{p_2}{\alpha_1 p_1 + \alpha_2 p_2}\right) \\ &\leq \int_x p_2 \log\left(\frac{1}{\alpha_1 e^{-\epsilon} + \alpha_2}\right) \\ &= \epsilon - \log(\alpha_1 + \alpha_2 e^{\epsilon}).\end{aligned}$$

Therefore,

$$\begin{aligned}\tilde{\mathbf{D}}_{p_1,p_2} &= \alpha_1[\epsilon - \log(\alpha_1 e^{\epsilon} + \alpha_2)] \\ &\quad + \alpha_2[\epsilon - \log(\alpha_1 + \alpha_2 e^{\epsilon})] \\ &\quad + \alpha_1 \log \alpha_1 + \alpha_2 \log \alpha_2 \\ &= \epsilon - \alpha_1 \log(e^{\epsilon} + \alpha_2/\alpha_1) \\ &\quad - \alpha_2 \log(e^{\epsilon} + \alpha_1/\alpha_2) \\ &\leq \mathcal{O}((1-\alpha_2)\epsilon) \\ &= \mathcal{O}(\alpha_1 \epsilon/(\alpha_1 + \alpha_2))\end{aligned}$$

where we manually add $(\alpha_1 + \alpha_2)$ to normalize α_1. □

B.2 Experiment Details

B.2.1 Dynamic schedules

We use dynamic schedules for learning rates and the adversarial parameter λ following previous work [46]. Specifically,

$$\begin{aligned}\eta_t &= \frac{1}{(1 + 10\frac{t}{T_{\max}K})^{0.75}} \\ \lambda_t &= \frac{2}{1 + \exp(-10t/T_{\max})} - 1\end{aligned}$$

where K is the number of local iterations and $T_{\max}$ is the number of global rounds. Notably, η_t is schedule locally and λ_t is scheduled globally.

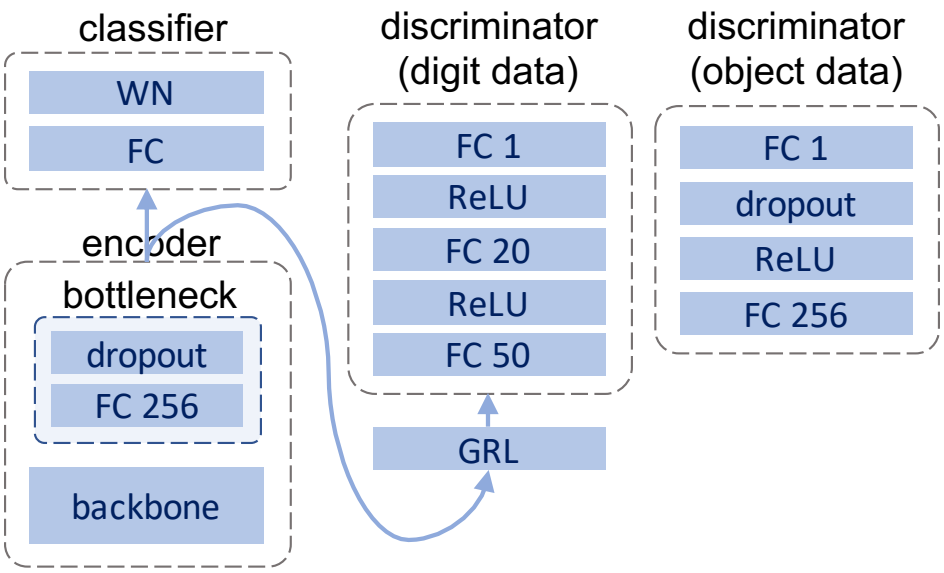


Figure B.1: Network architectures for digit and object datasets. WN denotes the weight-norm layer [81] and FC 256 denotes fully-connected layer with 256 units. GRL is the gradient reversal layer [46].

B.2.2 Network architectures

Federated UDA. The network architectures are presented in Fig. B.1.

Batch normalization in FADE. During training, we share the parameters of ResNet between users. Notably, in ResNet, batch normalization (BN) layer is densely embedded in different depth. The BN layer is known to be important for transferring between distinct domains, because the hidden representations will be normalized with mean and variance estimated from a batch. Because such estimation could be easily biased by a small batch, running estimation by accumulating results from previous batches is a common practice. Thus, it is also important for all users to get the global estimate of the mean and variance by communication. However, sharing such a running estimate of representation mean and standard variance may leak the private information [103, 104]. For example, given a feature vector at a specific layer, the input image can be reverted using a conditional generative network [103, 104]. Instead of sharing the mean and variance (*BN states*), we keep the values the same as values pre-trained on ImageNet.

Fair federated learning. We depict the network architectures for Adult and MCI

datasets in Fig. B.2. For the Adult dataset, we aim to evaluate the performance of deep networks. Thus, we use a deeper network other than a shallow one for central algorithms [93]. Because of the small size of the MCI dataset, we adopt a small network architecture where only two layers of LSTM are used for feature extraction and one layer for classification or group identifying.

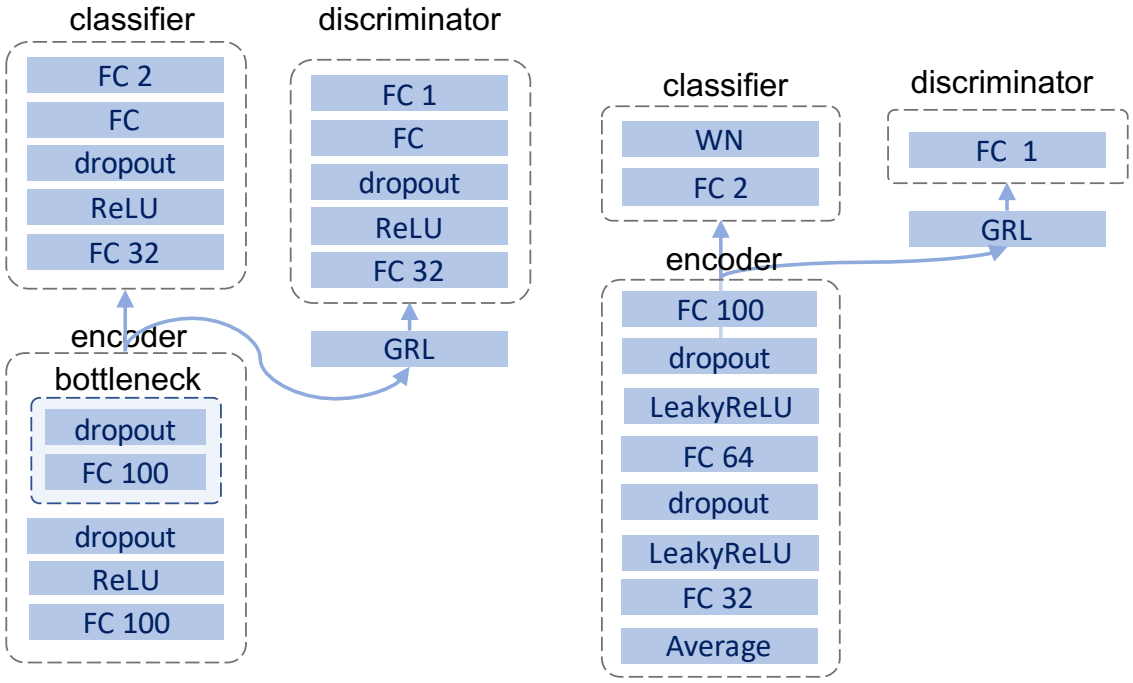


Figure B.2: Network architectures for Adult and MCI datasets. LSTM 100 indicates a Long Short-Term Memory (LSTM) cell with 100 hidden units.

B.2.3 Details of MCI datasets

Dataset Due to the mild symptoms and expansive cost of clinic diagnosis, early detection of MCI is a hard task. To address the challenge, MCI detection models is built on a MCI dataset, which is collected with Intelligent Systems for Assessing Aging Change (ISAAC), a longitudinal cohort study [69, 55]. A total of 152 participants were enrolled beginning in 2017. 12 variables are extracted from the participants' sensor data and clinical diagnoses was done once a year. Meanwhile, four kinds of demographic information are also recorded, including age, gender, education, and ethnicity, which are potentially unfair features for each patient.

Though prior work has shown the effectiveness of machine learning methods in diagnosis prediction [84, 55], the possibility of training such a model fairly in a distributed framework remains unknown. We assume the sensor data can be immediately trained locally and only the trained models are sent to the server. The distributed framework brings in several new challenges. First, users' data are kept locally and many users only have one-class data which makes the local model less discriminative. For example, 13 users are always diagnosed as MCI during his/her recording. Second, it is difficult to do adversarial learning like Fig. 3.1b. Because the users' group information, e.g., gender, can not be revealed to others, the server has no idea who will be the adversarial group. Therefore, we utilize the FADE framework to tackle these issues as illustrated in Fig. 3.1c. As far as privacy is concerned, in the ISAAC protocol, the sensor data were collected periodically by engineers such that the user data are kept away from others. But we argue that our extension to federated setting is practical because the data are not directly shared.

Preprocessing. Since the records of some patients are missing due to occasionally off-line of sensor systems, and these incomplete samples can introduce uncertainty in our experiments, we choose to remove some samples according to a certain missing value. To generate samples, hundreds of days of records for each patient will then be sliced by a moving window, and each slice is used as a sample for training or to be predicted. The slicing is done inside each person's sequence without overlap. The time window is moved in a step of 7 days. Only a subsequence of a small enough ratio of missing values will be maintained for the current study. The number of sequences for each patient is related to the amount of data the patient has. For some of the patients, they have only a small number of records. We also remove the samples of those patients to avoid inaccurate prediction.

We have 12 varaibles in total, including gender (Rsex), years of education (Ryrschool), race/ethnicity (Rethnic), age at each date (ageyrs), total computer use (compuse), computer sessions (numcsess), track sensor line (linenum), walks (numwalks), mean walking speed (meanws), upper quartile of walking speed (wsq3), coefficient of var of walking speed

(wscv) and std deviation of walking speed (wsstddev). We preprocess special variables in the following specified methods. For linenum which is a sensor metric identity value, its integer values are transformed into a one-hot encoding form that uses the position of a single one to indicate the ID value. RSex and Rethic variables are encoded in the same way. The ages are transformed by 3-bin discretization. All continuous variables are normalized within $[-1, 1]$ by min-max scaling such that no significant variance will occur between different variables and their coefficients could be trained in a numerically robust way.

All the data features are collected in a relatively redundant way, for which they should be carefully selected for better prediction performance. We select features using mutual information, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and the higher value means higher dependency. A special case is the linenum variable which only makes sense when other walking speed features are used. As a result, when a walking feature is selected according to the above metrics, the linenum variable is automatically included.

APPENDIX C

EFFICIENT FEDERATED LEARNING FOR ON-DEMAND AND IN-SITU CUSTOMIZATION

C.1 Experiments

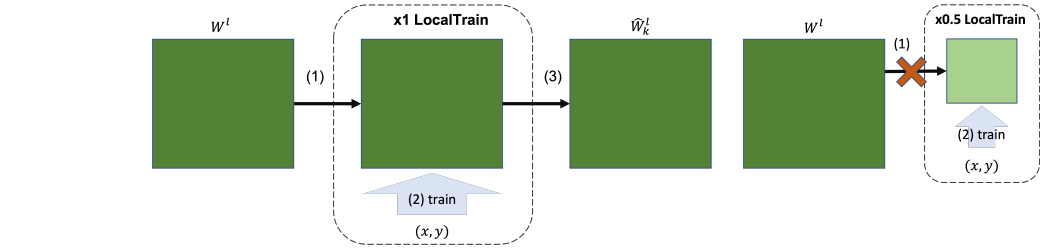
In this section, we provide more details about our experiments and additional evaluation results.

C.1.1 Parallel implementation of Split-Mix and convergence

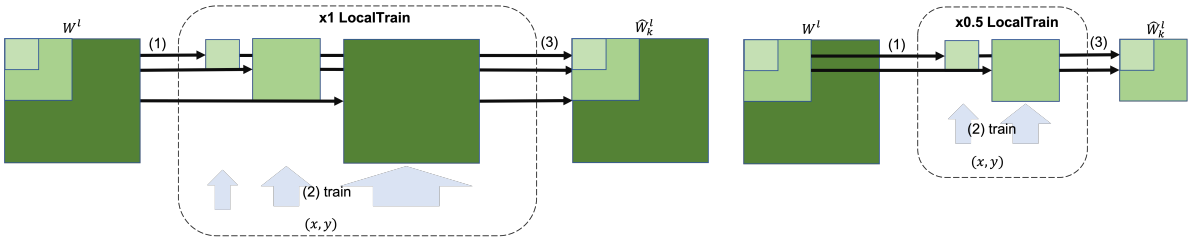
In this section, we elaborate on the implementation and efficiency of Split-Mix. In Fig. C.1, we conceptually compare the training by three methods, when two clients capable of training $\times 1$ and $\times 0.5$ are considered. FedAvg can train the $\times 1$ net on the $\times 1$-capable client but not on the $\times 0.5$-capable client. Through parameter sharing among different model widths, SHeteroFL can train multiple widths in a sequential manner and can fit into $\times 0.5$-capable clients. Unlike SHeteroFL, Split-Mix trains four base models in one parallel pass and therefore is the most efficient and flexible method. Remarkably, in the worst case, training $\times 1$ net of Split-Mix is as efficient as FedAvg and more efficient than SHeteroFL, if all the base weights, $w^l_{0,r}, \ldots, w^l_{3,r} \in \mathbb{R}^{r \times r}$ (where $r = 0.25$ here), are embedded into a $4r \times 4r$ weight matrix[1]. When base models are embedded into a full net, non-trainable parameters can be masked out (grey areas) to avoid interference between base models.

With the aforementioned implementation, we compare the convergence versus the wall-clock time in Fig. C.2. We implement all algorithms in `PyTorch 1.4.1` run on a single NVIDIA RTX A5000 GPU and a 104-thread CPU. Fig. C.2 shows the elapsed computation time from the initialization to a maximal number of iterations. The maximal number of iterations is set to be the same for all methods, such that it is easier to compare the stopping time. In Fig. C.2, Split-Mix is much more efficient than the SheteroFL. Note that FedAvg can only train the $\times 0.125$ net which includes much fewer parameters, For this reason, Split-Mix

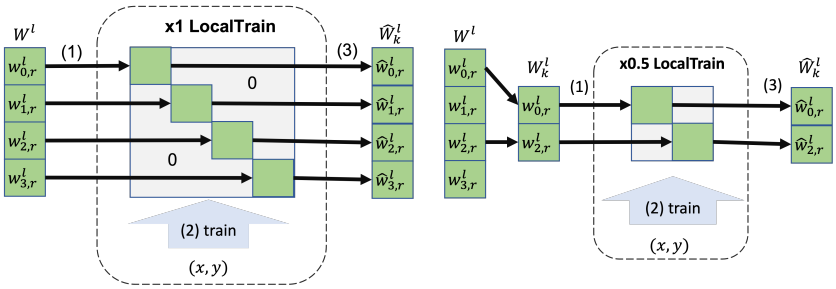
[1]For the simplicity of notations, we assume the width of layer l is r, though the width could vary by layer in general.



(a) FedAvg



(b) SHeteroFL



(c) Split-Mix

Figure C.1: Illustration of training weight matrices on a $\times 1$-net-capable or $\times 0.5$-net-capable client. (1) Download the global weight matrix W^l of layer l or a selected subset W_k^l. (2) Train weights on a batch data (x, y). (3) Upload trained weight matrix $\hat{W}_k^l$.

is slightly slower than FedAvg, but the degradation of efficiency trades in better accuracy than FedAvg.

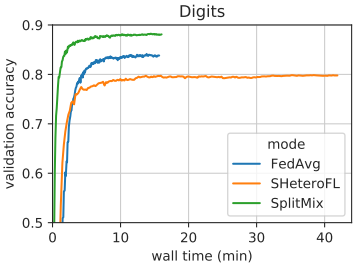


Figure C.2: Validation accuracy of the budget-compatibly-widest nets by wall-clock time. All algorithms are run with the same number of iterations (200).

C.1.2 Experimental configurations

In Algorithm C.1, we elaborate the local training of DBN. Specifically, each BN is trained independent with different input samples. The maximization problem in Algorithm C.1 can be solved by an n-step projected gradient descent, and is commonly known as *PGD attack* [94].

Algorithm C.1: LocalTrain(W_k, D_k, E, η) with DBN and adversarial training

Initialize models $\hat{W}_k$ by W_k
for $e \in \{1, \cdots, E\}$ **do**
  for mini-batch $B = \{(x, y)\}$ in D_k **do**
    for $\hat{w}_{i,r} \in \hat{W}_k$ in parallel **do**
      Set f to use clean BN
      $L \leftarrow \frac{1}{|B|} \sum_{(x,y)\in B} L_{CE}(f(x; \hat{w}_{i,r}), y)$
      Set f to use noise BN
      $\tilde{B} = \emptyset$
      for $x \in B$ **do**
        Perturb $\tilde{x} = x + \delta$ with $\delta \leftarrow \arg\max_{\|\delta\|_\infty \leq \epsilon} L_{CE}(f(x + \delta; \hat{w}_{i,r})), y)$
        $\tilde{B} \leftarrow \tilde{B} \cup \{(\tilde{x}, y)\}$
      $L \leftarrow \frac{1}{2}\left\{L + \frac{1}{|\tilde{B}|} \sum_{(\tilde{x},y)\in\tilde{B}} [L_{CE}(f(\tilde{x}; \hat{w}_{i,r}), y)]\right\}$
      $\hat{w}_{i,r} \leftarrow \hat{w}_{i,r} - \eta \frac{\partial L}{\partial \hat{w}_{i,r}}$
Return $\hat{W}_k$

Data. Both CIFAR10 and Digits are 10-way classification tasks. We follow the non-*i.i.d* benchmark of [80] to extract 10 classes from DomainNet, which is publicly available in FedBN codes. To illustrate the multiple-domain datasets, we sample several images from Digits and DomainNet datasets in Fig. C.3.

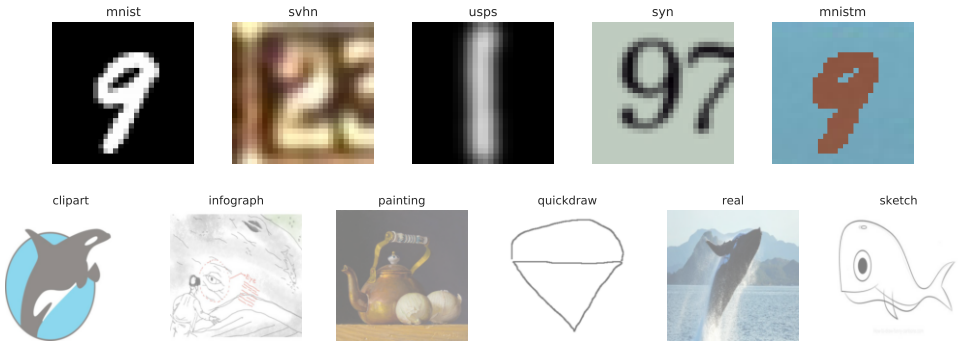


Figure C.3: Sample images from multiple domain datasets.

Table C.1: Network architecture for Digits dataset.

Layer	Details
feature extractor	
conv1	Conv2D(64, kernel size=5, stride=1, padding=2)
bn1	DBN2D, RELU, MaxPool2D(kernel size=2, stride=2)
conv2	Conv2D(64, kernel size=5, stride=1, padding=2)
bn2	DBN2D, ReLU, MaxPool2D(kernel size=2, stride=2)
conv3	Conv2D(128, kernel size=5, stride=1, padding=2)
bn3	DBN2D, ReLU
classifier	
fc1	FC(2048)
bn4	DBN2D, ReLU
fc2	FC(512)
bn5	DBN1D, ReLU
fc3	FC(10)

Hyper-parameters. In general, for local optimization we use stochastic gradient descent (SGD) with 0.9 momentum and 5×10^{-4} weight decay. Dataset specific settings are stated as follows. CIFAR10: Following HeteroFL [28], we train with 5 local epochs and 400 global communication rounds. Globally, we initialize the learning rate as 0.01 and adjust the learning rate at 150, 250 communication rounds with a scale rate of 0.1. Locally, we use a larger batch size of 128, to speed up the training in simulation. Digits: We use a cosine annealing learning rate decaying from 0.1 to 0 across 400 global communication rounds. SGD is executed with one epoch for each local client. DomainNet: We use a constant learning rate 0.01 and run 400 communication rounds in total. Similar to Digits, SGD is executed with one epoch for each local client.

Network architectures. Architectures of modified AlexNet (for DomaiNet) and CNN (for Digits) can be found in [80] and public codes [2]. For reader's reference, we provide the layer details in in Tables C.1 and C.2. For the convolutional layer (Conv2D or Conv1D), the first argument is the number channel. For a fully connected layer (FC), we list the number of hidden units as the first argument. The implementation of the preactivated ResNet can

[2]https://github.com/med-air/FedBN

Table C.2: Network architecture for DomainNet dataset.

Layer	Details
feature extractor	
conv1	Conv2D(64, kernel size=11, stride=4, padding=2)
bn1	DBN2D, ReLU, MaxPool2d(kernel size=3, stride=2)
conv2	Conv2D(192, kernel size=5, stride=1, padding=2)
bn2	DBN2D, ReLU, MaxPool2d(kernel size=3, stride=2)
conv3	Conv2D(384, kernel size=3, stride=1, padding=1)
bn3	DBN2D, ReLU
conv4	Conv2D(256, kernel size=3, stride=1, padding=1)
bn4	DBN2D, ReLU
conv5	Conv2D(256, kernel size=3, stride=1, padding=1)
bn5	DBN2D, ReLU, MaxPool2d(kernel size=3, stride=2)
avgpool	AdaptiveAvgPool2d(6, 6)
classifier	
fc1	FC(4096)
bn6	DBN1D, ReLU
fc2	FC(4096)
bn7	DBN1D, ReLU
fc3	FC(10)

be found in the repository of HeteroFL [28].

Batch-normalization for customizable model sizes. As pointed out by [28], HeteroFL relies on mini-batch batch-normalization to stabilize the training with multiple model widths and compute the statistics afterwards. Another advantage of such strategy is on federation of multi-domain clients. [80] showed that using local batch-normalization helps model personalization for non-*i.i.d* local features. As min-batch BN statistics simulate the local BN idea, it should enjoy the similar benefit. To eliminate the potential biases in personalization or convergence caused by different estimation methods of BN statistic, we let *all* compared algorithms using the same mini-batch strategy. To reveal the effect of different BN statistic solutions for Split-Mix, we provide a detailed ablation study regarding the estimation of BN statistics in Section C.1.3.

Loss function for class non-*i.i.d* FL. As some classes are missing locally in non-*i.i.d* setting, class-specific parameters in the classifier head may be updated without proper

supervision and results in random updates. To mitigate the effect of missing classes locally, we use the same masked cross-entropy loss as introduced by HeteroFL, where absent classes are masked out.

C.1.3 Ablation study of network scaling and BN statistics

In this section, we evaluate how the network rescaling and BN statistics affect the performance. For rescaling, we consider the parameter initialization (*rescale init*) and layer outputs (*rescale layer*). For BN statistics, we consider four options. The *batch average* one will estimate the statistics by one batch of data. The *post average* one will use the batch average strategy during training but re-estimate the statistics using client data afterward, which was adopted by HeteroFL. To gain better BN statistics, we run the model on a training set for 20 epochs. The *tracked* one will track statistics during training. The *locally tracked* one will also track statistics but the statistics will not be shared with the server for averaging, which can benefit clients' privacy and personalization [80].

We report full ablation results in Table C.3. **1)** First we compare the use of BNs without in-training tracking. The post-average BN performs best compared to other BN choices. With similar performance, the batch average BN does not need multiple rounds of evaluation of BN statistics, which is more efficient for inference. **2)** Then we compare the use of BNs tracked during training. Consistent with the prior study [80], locally tracked BN performs better than the globally averaged one. **3)** Rescaling layer outputs barely affect the accuracy, but it could be poisonous for tracked BN statistics. **4)** The initialization rescaling greatly improves the performance regardless of the choice of BN statistics.

In conclusion, either locally tracked or batch averaged BN statistics can yield both efficient and accurate performance. Post-averaged BN statistics may be preferred if post averaging is bearable for efficiency, especially for large-scale datasets. Rescaled initialization is an essential ingredient for Split-Max to perform well. Layer rescale is not recommended if batch average or tracked BN statistics are utilized.

Table C.3: Ablation study of network scaling and BN statistics on Digits dataset. Accuracy of different customized widths are presented.

BN stat	rescale init	rescale layer	×0.125	×0.25	×0.5	×1
batch average	✗	✗	81.1%	84.3%	86.2%	87.3%
	✗	✓	81.1%	84.2%	86.2%	87.2%
	✓	✗	84.5%	87.5%	88.9%	89.8%
	✓	✓	84.6%	87.5%	89.0%	89.8%
post average	✗	✗	81.2%	84.5%	86.2%	87.4%
	✗	✓	81.2%	84.4%	86.2%	87.3%
	✓	✗	84.5%	87.5%	89.0%	89.9%
	✓	✓	**84.9**%	**87.8**%	**89.3**%	**90.2**%
tracked	✗	✗	79.6%	82.8%	84.8%	85.7%
	✗	✓	9.4%	10.6%	10.6%	10.6%
	✓	✗	83.5%	86.4%	87.9%	88.7%
	✓	✓	8.8%	8.8%	8.8%	10.6%
locally tracked	✗	✗	81.1%	84.3%	86.3%	87.3%
	✗	✓	9.8%	10.6%	10.1%	11.7%
	✓	✗	**84.9**%	87.7%	89.1%	90.0%
	✓	✓	10.5%	10.6%	12.1%	11.1%

C.1.4 Experiments with i.i.d FL

In addition to non-*i.i.d* FL settings, we experiment with *i.i.d* FL where each client will own data of 10 classes from the CIFAR10 dataset. Results using 100% and 50% training data are included in Table C.4. We observe a great increase in the accuracy compared to the non-*i.i.d* experiments, which is a common phenomenon that non-iid FL will perform worse globally. The similar performance degradation was observed in [28], as well. In Table C.4, our method performs better in larger widths with fewer training data, whose performance approaches that of unconstrained individual FedAvg. Our method provide a monotonous relation between model size and accuracy (larger models are more accurate) and uses fewer parameters and MACs even compared to wider baseline networks, though performs worse in smaller widths because the slimmest networks are updated less frequently in budget-insufficient clients compared to the SHeteroFL or FedAvg. Worth to mention, our method uses much fewer parameters and operation counts for the same accuracy. For example, Split-Mix requires 7.2M MACs and 1.4M parameters for 81.1% accuracy while SHeteroFL needs

twice of the complexity, given the 50% CIFAR10 *i.i.d* configuration.

Table C.4: Test results of customizing model width on the class non-*i.i.d* CIFAR10 dataset.

	Individual FedAvg			SHeteroFL			Split-Mix (ours)		
width	Acc	MACs	#Params	Acc	MACs	#Params	Acc	MACs	#Params
				CIFAR10 *i.i.d* FL (100%)					
×0.125	**82.2%**	0.9M	0.2M	81.9%	0.9M	0.2M	80.9%	0.9M	0.2M
×0.25	~~86.1%~~	~~3.5M~~	~~0.7M~~	**85.2%**	3.5M	0.7M	83.4%	**1.8M**	**0.4M**
×0.5	~~89.8%~~	~~14.0M~~	~~2.8M~~	**86.5%**	14.0M	2.8M	85.2%	**3.6M**	**0.7M**
×1	~~91.0%~~	~~55.7M~~	~~11.2M~~	85.9%	55.7M	11.2M	**86.0%**	**7.2M**	**1.4M**
				CIFAR10 *i.i.d* FL (50%)					
×0.125	**77.3%**	0.9M	0.2M	77.2%	0.9M	0.2M	74.2%	0.9M	0.2M
×0.25	~~79.9%~~	~~3.5M~~	~~0.7M~~	**79.7%**	3.5M	0.7M	77.9%	**1.8M**	**0.4M**
×0.5	~~83.2%~~	~~14.0M~~	~~2.8M~~	**80.1%**	14.0M	2.8M	79.5%	**3.6M**	**0.7M**
×1	~~84.6%~~	~~55.7M~~	~~11.2M~~	75.5%	55.7M	11.2M	**81.1%**	**7.2M**	**1.4M**

C.1.5 More budget distributions

In our experiments, we generally use an exponential budget distribution: $R_k = (1/2)^{\lceil 4k/K \rceil}$. Though the distribution represents the imbalance between the budget-sufficient and budget-insufficient clients, real-world applications may encounter a wider variety of budget distributions. Thus, we extend our problem assumption to budget distributions with *more budget-sufficient clients* where we let more groups to have ×1 or ×0.5 net training capability and with *step-increase budgets* where we increase budgets by a fixed step (e.g., ×0.25). In addition, we consider a *log normal* distribution which concentrate around 0.45 budget with few wider or extremely budget-insufficient clients. We partition the budget distribution into 0.125-width bins and each client will only train the maximal compatible width varying from 0.125 to 1, which greatly increases the number of slimmable subnetworks (8 now compared to previous 4). To reduce the overhead of slimmable training, we use HeteroFL instead of SHeteroFL. Fig. C.4 reports the per-width accuracy for Split-Mix and SHeteorFL on the Digits dataset. For SHeteroFL, we only report the evaluated performance on trained widths. In other words, if the maximal width is ×0.5, we will not report results of the ×1 net. For Individually-trained FedAvg (Ind. FedAvg), models individually trained for

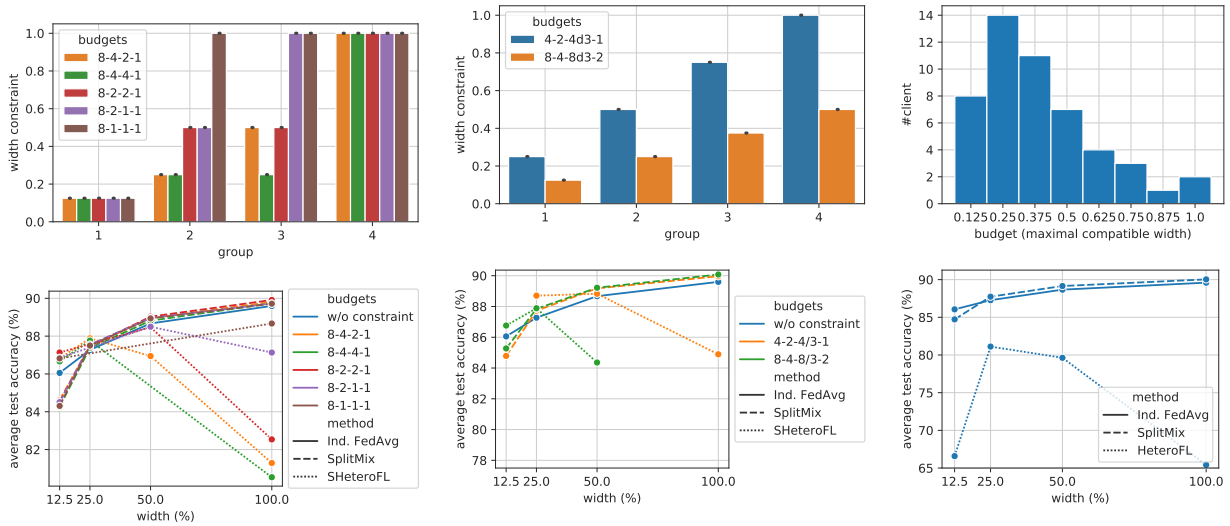
each width are reported, which ignores the width constraints by users and therefore only serves as reference upper bounds. Regardless of the budget distributions, for instance, more budget-sufficient clients (Fig. C.4a) or non-exponential distributions (Fig. C.4b), Split-Mix outperforms SHeteroFL with larger widths.

C.1.6 Effect of lower contact rates

Because of varying communication conditions in deployment, the times that clients actively and successfully upload their models could be fewer than expected. To evaluate the robustness of customization federated algorithms, we conduct federated experiments with a varying number of active clients per round in Fig. C.5. Because of the limited number of communication rounds (within 300 rounds), the test accuracy decreases by fewer contact clients. This is a common phenomenon because lower contact rates requires more communication rounds to reach the same performance as the full-contact competitors do. Though global performance generally decreases, we find that the wider networks are more accurate if trained by SplitMix. One source for the advantage is the modular base models in Split-Mix, which can be easily distributed into different rounds for training. Rather, the wider integrated networks in SHeteroFL lower their chance to be aggregated globally and therefore their global performance declines.

C.1.7 Negative impact of constrained budgets on wider networks

In Fig. 4.2, we show how the SHeteroFL fails to train wider networks with budget-constrained clients. To show the generality of such a problem, we extend the experiments to Digits and CIFAR10 datasets in Fig. C.6. Though wider networks converges faster at the beginning, they meanwhile overfit limited data in a few clients and therefore their validation accuracy no longer improves.



(a) More budget-sufficient clients (b) Step-increase budget distributions (c) Log-normal distributions

Figure C.4: Vary the budget distribution. The training budgets, i.e., width constraints, are depicted in the upper figures by group. The budget distribution name, for example, 8-4-2-1, means $\times 1/8$, $\times 1/4$, $\times 1/2$ and $\times 1$ width constraints for each group, respectively. The lower figures compare the performance of trained models with customized withs.

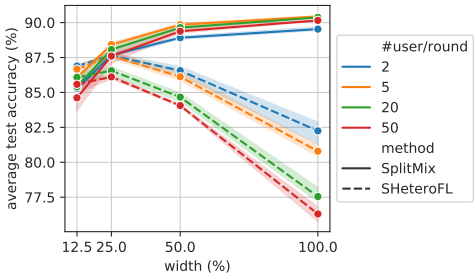


Figure C.5: Vary the number of clients uploading models per round.

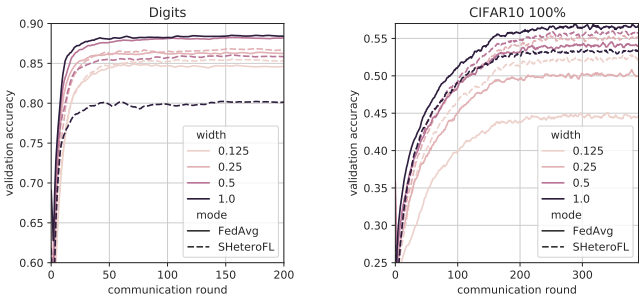


Figure C.6: Convergence of different-width models.

APPENDIX D

OUTSOURCING TRAINING WITHOUT UPLOADING DATA

D.1 More Method Details

In this section, we elaborate on additional technical details of our paper.

D.1.1 Automated client labeling via private kNN

Here, we briefly introduce the main idea of client labeling by private kNN. Given C classes, labeling is done by nearest neighbor voting:

$$f(x) = \arg\max_{c\in[C]} A_\epsilon(v_c),\ v_c = |\{(x', y') \in \mathbb{N}_K(x) | y' = c\}|,$$

where $\mathbb{N}_K(x)$ is a set including the K-nearest neighbors of x in the client dataset. A_ϵ is a privatization mechanism complying with the notion of ϵ-Differential Privacy (DP) [38]. In brief, privatization is done by adding Gaussian noise to a value with finite sensitivity. To filter out potential wrong labels, the client only returns high-confident samples by screening. Let the confidence of a pseudo label be $s(x) = \max_{c\in[C]} A_\epsilon(v_c)$ which is also privatized by the Gaussian noise mechanism. We find that the original version of screening may suffer from a large imbalance of pseudo labels. Per class, we screen the pseudo labels by selecting the top-60 confident samples given 600 labeling budget.

D.1.2 Improving client labeling

Because of the noise mechanism for privacy protection, the client labeling may be quite random if the selected samples are hard to discriminate. Thus, we propose to improve the discrimination of selected samples in advance, when the ECOS selects samples for labeling. First, we estimate the discrimination by the confidence in the ECOS. The ECOS confidence is defined by the vote count of the highest-voted class subtracting the one of the second highest one, denoted as v_r^{conf}. To merit the balancedness of samples, we filter the clusters to keep the top-70% samples with the highest confidence *per class*. When decompressing the clusters on the cloud, we incorporate the confidence into the sampling score by $v_r' = \psi\left[(v_r^{\text{conf}} + v_r)/2\right]$

where v_r is the original score.

D.1.3 Privacy Accountant for ECOS

To understand the privacy cost of ECOS, we review the techniques that are essential to establish the privacy bound.

Definition 8 (Differential Privacy [38]). *Suppose ϵ and δ are two positive constants. A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is (ϵ, δ)-DP if for every pair of neighboring datasets $X, X' \in \mathcal{X}$, and every possible measurable output set $Y \subset \mathcal{Y}$ the following inequality holds:*

$$Pr[\mathcal{M}(X) \in Y] \leq e^{\epsilon} Pr[\mathcal{M}(X') \in Y] + \delta,$$

where $Pr[\cdot]$ denotes the probability of a given event.

DP provides a way to quantify the privacy risk (termed as the difference between two probability given a pair of similar but different inputs) in the probability of δ. Though DP is a simple notion for risks, the estimation of a tight privacy bound is still challenging. For this reason, RDP is proposed an alternative tool.

Definition 9. *(Rényi Differential Privacy (RDP) [99]) A randomized algorithm $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is (α, ϵ)-RDP with order $\alpha > 1$ if for all neighboring datasets X, X' the following holds*

$$D_{\alpha}(\mathcal{M}(X)||\mathcal{M}(X')) \leq \epsilon,$$

where $D_{\alpha}(\cdot||\cdot)$ is the Rényi divergence between two distributions.

The RDP and DP can be connected by Lemma 5.

Lemma 5. *If a mechanism $\mathcal{M}$ satisfies (α, ϵ)-RDP, then it also satisfies $\frac{(\epsilon + \log 1/\delta}{\alpha - 1, \delta)}$-DP.*

To reveal the potential privacy risks, we give a theoretical bound on the privacy cost based on DP in Lemma 6. The proof of Lemma 6 is similar to Theorem 8 in [168] without confidence screening.

Lemma 6. *Suppose the subsampling rate γ and noise magnitude σ of the ECOS are positive values such that $\gamma \leq \min\left\{0.1, \sigma\sqrt{\log(1/\delta)/6}\right\}$ and $\sigma \geq 2\sqrt{5}$. The total privacy bound of the ECOS scoring $m = |\hat{D}^q|$ query samples with $n = |\hat{D}^p|$ private client samples is (ϵ, δ)-DP with $\delta > 0$ and*

$$\epsilon = \mathcal{O}(\frac{\gamma}{\sigma}\sqrt{\log(1/\delta)}), \tag{D.1}$$

if v_r in Algorithm 5.1 is estimated by using $\lceil \gamma n \rceil$ samples randomly subsampled from $\hat{D}^p$ with replacement and is processed by $\tilde{v}_r = v_r + \mathcal{N}(0, \sigma^2 I)$.

Proof. When one sample is absent from the private client dataset, the scores $[v_1, \ldots, v_R]$ will differ by 2 if without subsampling. By Lemma 11 of [168], the subsampled Gaussian mechanism accounted by the RDP is

$$\epsilon(\alpha) \leq \frac{24\gamma^2\alpha}{\sigma^2}$$

for all $0 < \alpha \leq \frac{\sigma^2 \log(1/\gamma)}{2}$, $\gamma \leq 0.1$ and $\sigma \geq 2\sqrt{5}$. By Lemma 5, we can convert the RDP inequality to the standard (ϵ, δ)-DP as

$$\epsilon = \frac{24\gamma^2\alpha}{\sigma^2} + \frac{\log(1/\delta)}{\alpha - 1}.$$

Let α be $1 + \frac{\sqrt{\log(1/\delta)}}{\sqrt{\frac{24\gamma^2}{\sigma^2}}}$. Thus,

$$\epsilon = \frac{24\gamma^2}{\sigma^2} + 4\frac{\gamma}{\sigma}\sqrt{6\log(1/\delta)} \leq 8\frac{\gamma}{\sigma}\sqrt{6\log(1/\delta)},$$

where the last inequality is derived by the given range of γ. This thus completes the proof. □

The above bound implies that the privacy cost for our method is invariant w.r.t. the scale of the query dataset $\hat{D}^q$, and only depends on the DP parameters. Note that the Lemma 6 is an asymptotic bound which requires some strict conditions on γ or other variables. In practice, we leverage the tool of analytic privacy accountant through a numerical method [144], with which we can relax the strict conditions.

D.1.4 Social Impacts and Limitations

The conflict between the concerns on data privacy and demands for intensive computation resources for machine learning has composed the main challenge in training outsourcing. In this work, we devote our efforts to outsourcing with uploading data by leveraging authorized or public datasets. As the public datasets commonly available in many applications are collected from multiple data sources and thus tend to be non-identically distributed as the client data, it casts new challenges to use the public in place of the client dataset. Our method addresses this problem with accountable privacy cost and low communication and computation complexity. Therefore, the proposed ECOS provides a promising solution to mitigate the aforementioned conflict between the privacy and computation desiderata. Therefore, users from a broader spectrum can benefit from such a method to confidentially conduct cloud training.

We also recognize open questions of the proposed solution for future studies. For example, the public dataset may require additional data processing, e.g., aligning and cropping for better prediction accuracy. In our empirical studies, we only consider the tasks in the computer vision area, though no assumption was made on the data structures, though we expect the principles can be adapted to other data types with minimal efforts. More data types, including tabular and text data, will be considered in the follow-up works.

D.1.5 Connection to Federated Learning

Both our method and federated learning (FL) [97] consider protecting data privacy via not sharing data with the cloud. The key difference between FL and our concerned problem (training outsourcing) is that FL requires clients to do training while ECOS outsource the training to the cloud server. Since ECOS does not require local training, ECOS can be adhocly plugged into FL to obtain an auxiliary open-source dataset for enhancing the federated training. The ECOS can be used either before federated training (when a pre-trained model is required) or during federated training (when the pre-trained model can be replaced by the on-training model).

D.2 Experimental Details and More Experiments

Complementary to the main content, we provide the details of the experiment configurations to merit the reproducibility. We also conduct more qualitative experiments to understand the efficiency and effectiveness of the proposed method.

D.2.1 Experimental Details

We organize the case-specified configurations into three cases and discuss the general setups first.

For Digits, we train the model for 150 epochs. We adopt a convolutional neural network for Digits in Table D.1 and ResNet18 for DomainNet. To solve the learning problems including FixMatch and distillation-based compression, we use stochastic gradient descent with the momentum of 0.9 and the weight decay of 5×10^{-4}. We use $s = 5$ for the scale function ψ_s on DomainNet and $s = 1$ on Digits. When not specified, we noise the ECOS query with the magnitude as 25. In Case 1 and 2, we reduce the noise magnitude to 10 for DomainNet, since the two queries can bear more privacy costs to trade for higher accuracy.

Case 1: Selective manual labeling. We make use of the off-the-shelf ResNet18 is pre-trained on the ImageNet, which is widely accessible online. We adopt FixMatch for semi-supervised learning with the coefficient of 0.1 on the pseudo-labeled loss, the moving average factor of 0.9, and the batch size of 64 for DomainNet and 128 for Digits. To avoid feature distorting, we warm up the fine-tuning by freezing all layers except the last linear layer with a learning rate of 0.01. After 30 epochs, we fine-tune the model end to end until 80 epochs to avoid overfitting biased data distributions.

Case 3: Adaptive model compression. We first pre-train a ResNet50 using all labeled open-source data for 100 epochs with a cosine-annealed learning rate from 0.1. The same warm-up strategy as Case 1 is used here. To extract the knowledge from ResNet50, we combine the knowledge-distillation (KD) loss L_{KD} and cross-entropy loss L_{CE} by $0.1 \times L_{KD} + 0.9 \times L_{CE}$ and calculate the losses on the selected samples only. The temperature in the KD loss is set to be 10.

Case 2: Automated client labeling. For the cloud training, we adopt the same configuration as the selective manual labeling. For private kNN, we let the client release 600 labels with class-wise confidence thresholds described in the last section. We noise the labeling in the magnitude of 25 and the confidence in the magnitude of 75. For both datasets, we subsample 80% client data per labeling query to reduce the privacy cost.

Table D.1: The structure of the conventional neural network for the Digits dataset.

Layer name	PyTorch pseudo code
conv1	Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1))
bn1	BatchNorm2d(64, eps=1e-05, momentum=0.1)
conv1_drop	Dropout2d(p=0.5, inplace=False)
conv2	Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1))
bn2	BatchNorm2d(128, eps=1e-05, momentum=0.1)
conv2_drop	Dropout2d(p=0.5, inplace=False)
fc1	Linear(in_features=2048, out_features=384, bias=True)
fc2	Linear(in_features=384, out_features=192, bias=True)
fc3	Linear(in_features=192, out_features=11, bias=True)

We conduct our experiments on the Amazon Web Service platform with 4 Tesla T4 GPUs with 16GB memory and a 48-thread Intel CPU. All the code is implemented with `PyTorch` 1.11. To account for the privacy cost, we utilize the open-sourced `autodp` package following the private kNN.

D.2.2 Effect of Parameters

To better understand the proposed method, we study the effect of the important hyper-parameters. To this end, we consider the selective labeling task with Digits, keeping 50% of the SVHN dataset at the client end. Both the ID+OoD and OoD cases are evaluated to reveal the method's effectiveness under circumstances with various hardness. Also, we study how the score scale s affects the ID ratios (denoted as the ID TPR) in the selected set and the number of effective samples. We only examine the ID TPR corresponding to the proximity objective in Eq. (5.2) if known ID samples are present on the cloud, namely in the *ID+OoD* case. In the middle panes of Figs. D.1 and D.2, we show that our method can effectively improve the ID TPR against the inherent ratio of ID samples on the cloud.

Effect of the compression size R. In Fig. D.1, we evaluate R in terms of the test

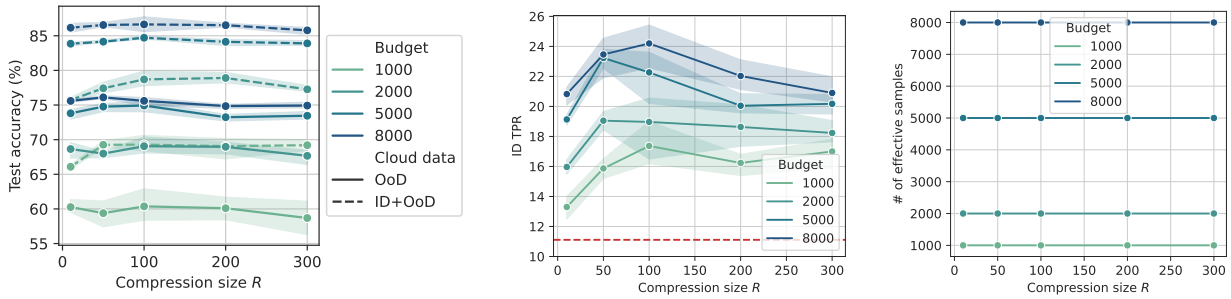


Figure D.1: Vary the compression size R and evaluate the test accuracy, ID ratios (%) in selected samples and the number of effectively selected samples. The red horizontal line indicates the ratio of ID samples in the whole cloud dataset.

accuracy. When the budget is small (1k and 2k budgets in the ID+OoD case), it is essential for the cloud server to sense the client distribution with higher accuracy via more queries. Therefore, a larger R is desired, which can increase the portion of ID samples in the selected set, as shown in the middle pane of Fig. D.1. Considering that a higher burden on communication comes with a larger R, the value of 100 leads to a fair trade-off to the accuracy in which case the ID TPR reaches a peak.

Given a larger budget, e.g., 8000, increasing R may lower the ID TPR. We attribute the decline to the limited size of the client dataset and privacy constraints. Given more clusters (i.e., R), the expected number of votes (proportional to the score) for each cluster will be reduced and is badly blurred by the DP noise. Thus, the ID TPR will decrease simultaneously, regardless of which the test accuracy is not significantly affected.

For the OoD case which is relatively harder for sampling due to the lack of true ID data, the parameter sensitivity is weakened, though the compression size of 100 is still a fair choice, for example, bringing in $1-2\%$ gains in the 5k, 8k cases comparing the worst cases.

Effect of the score scale s. The score scale s decides the sensitivity of the sampling in the sense of proximity. A larger s means that the ECOS will prioritize the proximity more during sampling. In Fig. D.2, we present the ablation study of s. A larger s is preferred when the budget becomes limited because it increases the ID TPR effectively. Though not significantly, an overly large s has a significantly negative influence on the accuracy,

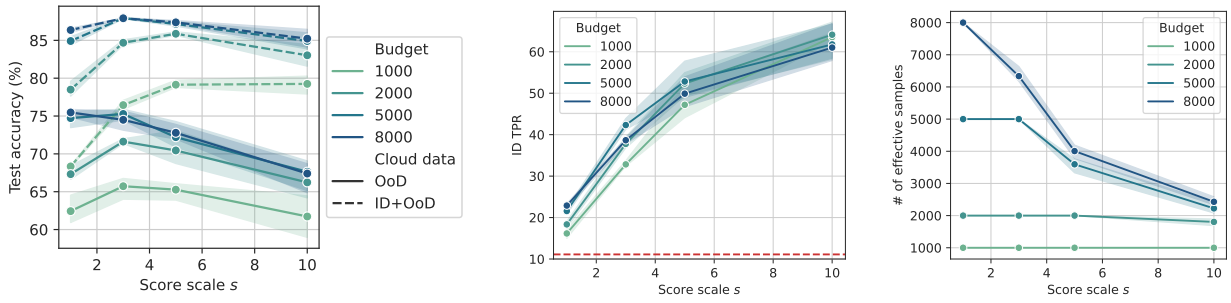


Figure D.2: Vary the score scale s in terms of test accuracy, ID ratios (%) in selected samples and the number of effectively selected samples (which could be smaller than the budget). The red horizontal line indicates the ratio of ID samples in the whole cloud dataset.

especially for the OoD case. The reason for the negative impact of s on a large budget can be understood by probing the number of effective samples. For budgets larger than 2000, the effectively selected samples are reduced with heavily scaled scores (e.g., $s \geq 3$) where the ECOS will concentrate its selection into very few clusters and eliminate the rest clusters strictly.

D.2.3 Evaluation of Sample and Privacy Efficiency

In Fig. D.3, we compare the sample efficiency in the selective labeling task with Digits, keeping 50% of the SVHN dataset at the client end. We obtain the upper-bound accuracy in the ideal case via random sampling when the cloud dataset distributes identically (ID) as the client dataset. When OoD data are included in the open-source cloud dataset (ID+OoD), the training becomes more demanding for the labeled samples. If none of the iid samples presents in the cloud set (OoD), the accuracy decreases quickly with the same labeled samples. In comparison, informative sampling by K-Center slightly improves the accuracy by different budgets and the proposed ECOS significantly promotes the sample efficiency. With ECOS, 8×10^3 labeled samples in the ID+OoD case achieves comparable accuracy as the ideal case, while baselines remain large gaps. Both in ID+OoD and OoD cases, our method yields competitive accuracy (at the 4×10^3 budget) versus the best baseline results using only half of the labeled data (at the 1×10^4 budget), dramatically cutting down the cost for manual labeling.

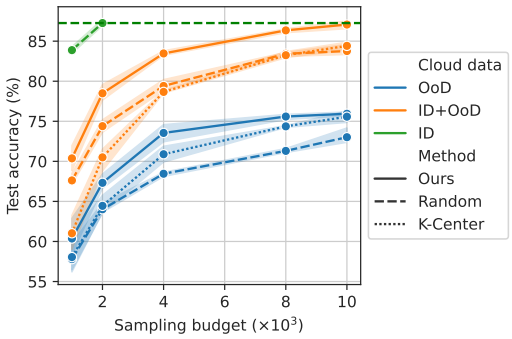


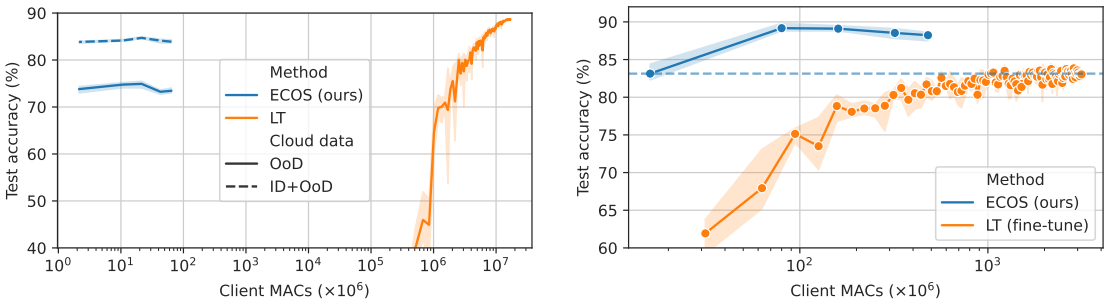
Figure D.3: Evaluation of the sample efficiency on selective labeling. The green horizontal line implies the ideal case when all ID cloud data are labeled.

On observing the gains in sample efficiency, readers may also notice that our method induces additional costs at privacy, as compared to the baselines. We point out that the cost is constant w.r.t. the sampling budget and is as neglectable as $(0.22, 10^{-5})$-DP. It is worth noticing that the cost is independent of the hyper-parameters of the ECOS because the ECOS communication is a *single* query for each private sample (so as for the private dataset), even if we increase the size of the query set (i.e., the compression size R). In practice, the client can control the privacy risk (namely, the privacy cost) flexibly by adjusting the noise magnitude and the subsampling rate.

D.2.4 Evaluation of Communication and Computation Efficiency

When improving the sample efficiency, we also need to take care of the communication and computation overheads brought by the ECOS. We examine the two kinds of efficiency by the same experiment configurations as in the last section.

Computation efficiency. In Fig. D.4, we compare the computation efficiency of our method to the local training (LT). We utilize the multiplication-and-addition counts (MACs) as the metric of computation (time) complexity, which is hardware-agnostic and therefore is preferred here. For a fair comparison, we tune the learning rate in $\{0.1, 0.01, 0.001\}$ with the cosine annealing during training and the number of epochs in $\{20, 50, 100\}$ of the LT to achieve a fair trade-off between the computation cost and test accuracy. For ECOS, since the computation cost linearly increases by the compression size (as shown in Fig. D.5), we



(a) Digits (b) DomainNet

Figure D.4: With the 5000 budget, we evaluate the computation efficiency. The efficiency of LT on the DomainNet is enhanced by linear fine-tuning.

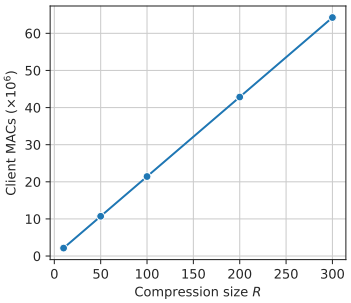


Figure D.5: The linearly growing computation cost by increasing the compression size on the Digits dataset.

vary the compression size to check the performance when increasing computation costs. On Digits, we observe a large computation save by our method, even if the cost of our method will gradually increase by the size R of the compressed query set.

Similar experiments are also run on the large-sized images using the DomainNet dataset (ID+OoD case), where the cost for extracting features is steeply increased by using a deep network (ResNet18). Recently, the most popular strategy for cloud training is two-phase learning: pre-training a model on the cloud using ImageNet and fine-tuning the linear classifier head on the client. Considering the large cost of feature extraction, we only let the client pre-extract features once only. Thus, the local training is as efficient as training a *linear* layer on extracted features. In Fig. D.4, our method outperforms the local training a lot using much fewer MACs for data matching. Because all training is outsourced to the

cloud, our method enables the end-to-end fine-tuning of the model resulting in better test accuracy. Even if the LT trains longer with higher computation costs, the test accuracy of the ECOS with the least MACs is comparable to the best performance of LT at around 10^9 MACs, where the ECOS only utilizes the 10% of MACs by LT.

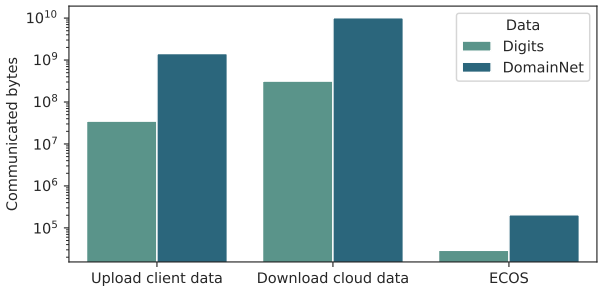


Figure D.6: Evaluate the communication efficiency.

Communication efficiency. We also compare the communication efficiency to the full cloud training (via uploading the whole client dataset) and fully client training (via downloading cloud dataset) in Fig. D.6. For the ECOS, we let the size of the query set be 100, which is the default configuration in our experiments. Because the ECOS only communicates a few low-dimensional features (for example, 512-dimensional ResNet-extracted features for DomainNet and 72-dimensional HOG features for Digits), it costs much fewer bytes compared to traditional outsourcing by uploading the client data. To be concrete, we also present the cost of downloading the cloud data and it is way more expensive than the rest two methods.