2023

Learning with constraints on processing and supervision

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

LEARNING WITH CONSTRAINTS ON PROCESSING AND SUPERVISION

by

DURMUŞ ALP EMRE ACAR

B.S., Antalya Bilim University, 2017

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2023

Approved by

First Reader

Venkatesh Saligrama, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Computer Science

Second Reader

Brian Kulis, PhD
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering
Associate Professor of Computer Science

Third Reader

Alexander Olshevsky, PhD
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering

Fourth Reader

Ananda Theertha Suresh, PhD
Senior Research Scientist
Google Research, New York

Dedicated to the people who lost their lives during the 7.8 magnitude earthquake on February 6, 2023 in Türkiye.

Acknowledgments

I would like to thank and express my gratitude to my advisor Prof. Venkatesh Saligrama. Working with him has always been exciting and inspirational to me. He taught me how to conduct research by encouraging me to simplify the problems and develop a fundamental understanding of each aspect of the issues. I also would like to thank my committee members Prof. Brian Kulis, Prof. Alex Olshevsky, and Dr. Ananda Theertha Suresh, for agreeing to be a member of my defense committee and enriching my thesis with their valuable feedback.

I thank my friends from BU for spending time with me, where I got a chance to relax and enjoy time together. They made the Ph.D. road a joyful journey.

Finally, I would like to thank my relatives and friends (especially Merve Yılmaz, Esat Artuğ, Doğukan Kalenderoğlu, and Esra Dikbaş) for their unconditional support and encouragement despite the distance and time gap. Talking to them and having video calls with them when needed gave me the strength to complete my studies.

This work was supported by a gift from ARM corporation (DA), CCF-2007350 (VS), CCF-2022446(VS), CCF-1955981 (VS), the Data Science Faculty Fellowship from the Rafik B. Hariri Institute Army Research Office Grant W911NF2110246, the Office of Naval Research Grant N0014-18-1-2257, and AFRL Contract no. FA8650-22-C-1039.

LEARNING WITH CONSTRAINTS ON PROCESSING AND SUPERVISION

DURMUŞ ALP EMRE ACAR

Boston University, College of Engineering, 2023

Major Professor: Venkatesh Saligrama, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Computer Science

ABSTRACT

Collecting a sufficient amount of data and centralizing them are both costly and privacy-concerning operations. These practical concerns arise due to the communication costs between data collecting devices and data being personal such as text messages of an end user. The goal is to train generalizable machine learning models with constraints on data without sharing or transferring the data.

In this thesis, we will present solutions to several aspects of learning with data constraints, such as processing and supervision. We focus on federated learning, online learning, and learning generalizable representations and provide setting-specific training recipes.

In the first scenario, we tackle a federated learning problem where data is decentralized through different users and should not be centralized. Traditional approaches either ignore the heterogeneity problem or increase communication costs to handle it. Our solution carefully addresses the heterogeneity issue of user data by imposing a dynamic regularizer that adapts to the heterogeneity of each user without extra transmission costs. Theoretically, we establish convergence guarantees. We extend our ideas to personalized federated learning, where the model is customized to each end user, and heterogeneous federated learning, where users support different model architectures.

As a next scenario, we consider online meta-learning, where there is only one user, and the data distribution of the user changes over time. The goal is to adapt new data distributions with very few labeled data from each distribution. A naive way is to store data from different distributions to train a model from scratch with sufficient data. Our solution efficiently summarizes the information from each task data so that the memory footprint does not scale with the number of tasks.

Lastly, we aim to train generalizable representations given a dataset. We consider a setting where we have access to a powerful teacher (more complex) model. Traditional methods do not distinguish points and force the model to learn all the information from the powerful model. Our proposed method focuses on the learnable input space and carefully distills attainable information from the teacher model by discarding the over-capacity information.

We compare our methods with state-of-the-art methods in each setup and show significant performance improvements. Finally, we discuss potential directions for future work.

Contents

List of Tables

List of Figures

List of Abbreviations

ACID	Active Class Induced Diversity
ALID	Anonymous Label Induced Diversity
CE	Cross Entropy
CTM	Current Task Metric
DiSK	Distilling Selective Knowledge
DNN	Deep Neural Networks
EMA	Exponential Moving Average
FedDyn	Federated Dynamic Regularizer
FedHeN	Federated Learning in Heterogeneous Networks
FL	Federated Learning
IID	Independently and Identically Distributed
KD	Knowledge Distillation
LTM	Long-Term Task Metric
MOML	Memory Efficient Online Meta Learning
PFL	Personalized Federated Learning
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

1.1 Learning With Constraints On Processing And Supervision: Motivation

Machine learning shows incredible advances in many applications, such as natural language understanding (Vaswani et al., 2017) and large-scale image classification (Huang et al., 2017). One essential condition of this success is access to massive data for training (Marcus, 2018). However, directly accessing a large amount of data results in serious concerns. Firstly, some data reveal private information, so bringing them into one place raises privacy concerns (McMahan et al., 2017a). Secondly, collecting all data in one place is a costly operation (Halgamuge et al., 2009). A more practical approach is to train models with data constraints such as processing and supervision, which allows for avoiding privacy concerns and communication costs.

The thesis aims to train models with data constraints by using the available excess information. In the applications we consider, a small amount of centralized data exists for a model train. We adjust the model's training to avoid overfitting to the small dataset and encourage it to use the extra information of different forms. We list the following applications we target in the thesis as motivation.

- **Next Word Prediction.** (McMahan et al., 2017b). Virtual keyboards suggest the next word while typing a text on most mobile phones. A machine learning model can be used for this task. Since one user's data is limited, more is needed to train a predictor model. Instead of centralizing data from all users, federated learning collaboratively trains a global model on all user data. We target fundamental practical problems in the current federated learning field, such as data heterogeneity, personalization, and support for different architectures. We propose communication-efficient novel solutions for each setting by obeying the no-data-sharing policy of federated learning.

- **Online Learning.** (Shalev-Shwartz, 2007). Data collection is a continuous process, and data distribution might change over time. For instance, the context of the message inputs would vary during election time and a major sport event. Online learning models this setting and aims to train a model on a sequence of data points arriving in phases. A small data set is revealed in each round, and it is not enough to train a machine-learning model. While there are methods where data sets from the previous rounds are stored to train a model, we propose a novel online meta-learning solution that does not store the data of earlier rounds and efficiently trains a model.

- **Distillation.** (Hinton et al., 2015). Distillation refers to a process where a powerful (teacher) model is utilized during the training of a student model to avoid overfitting. Distillation improves the generalization performance of the student models, where only the teacher model is used without any extra data. Current methods encourage the student and the teacher to give similar predictions. We view distillation differently and emphasize that the student should only follow the teacher on some input space. We propose a novel selective distillation technique where we force the student to follow teacher predictions on learnable inputs and do not distill information on the over-capacity hard-to-learn inputs.

1.2 Problem Formulation

This section states the general learning problem and provides specific forms for the subsequent problems. We mainly focus on supervised learning where the dataset consists of N training tuples denoted as $\boldsymbol{x} \in \mathcal{X}$ and corresponding labels $y \in \mathcal{Y}$ drawn independently and identically (IID) from an unknown distribution, .i.e $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$. We note that our methods can be applied to other problems by changing the loss definition.

We consider the following problem.

$$\arg\min_{\boldsymbol{w} \in \mathcal{W}} \frac{1}{N} \sum_{n \in [N]} \ell\left(\boldsymbol{w}; \boldsymbol{x}_n, y_n\right) \quad \text{subject to} \quad \mathcal{C}\left(\boldsymbol{w}, \{\boldsymbol{x}_n, y_n\}_{n=1}^N\right) \tag{1.1}$$

where $\boldsymbol{w}$ is the parameters of the deep neural network (DNN), $\ell\left(\boldsymbol{w}; \boldsymbol{x}_n, y_n\right)$ is the loss of the network for data point $(\boldsymbol{x}_n, y_n)$, and $\mathcal{C}\left(\boldsymbol{w}, \{\boldsymbol{x}_n, y_n\}_{n=1}^N\right)$ is a constraint depending on the problem setting.

We note that the no constraint setting corresponds to the standard DNN training, which results in overfitting with limited data.

Eq. 1.1 is an abstract definition, and we list the variants based on the setting we study below.

1.2.1 Federated Learning

Federated learning distributively learns a global model on the data from many users. Let there be m devices in the system. Each device k has its own labeled dataset denoted as $\{\boldsymbol{x}_n^k, y_n^k\}_{n=1}^{N_k}$. Each device k targets the following problem.

$$\arg\min_{\boldsymbol{w}_k \in \mathcal{W}} \frac{1}{N_k} \sum_{n \in [N_k]} \ell\left(\boldsymbol{w}_k; \boldsymbol{x}_n^k, y_n^k\right) \quad \text{subject to} \quad \boldsymbol{w}_k = \boldsymbol{w}^s \qquad \forall k \in [m] \tag{1.2}$$

where $\boldsymbol{w}^s$ is the server model. The constraint in Eq. 1.2 forces the local models to be consistent with each other so that one common global model minimizing the all data is learned.

Personalized Federated Learning

Personalized federated learning further customizes the global model based on user data. The goal is to learn a common global meta-model and obtain a customized device-specific model using an adaptation function for each user. Personalized federated learning considers the following problem for each device k,

$$\arg\min_{\boldsymbol{w}_k \in \mathcal{W}} \frac{1}{N_k} \sum_{n \in [N_k]} \ell\left(\overline{\boldsymbol{w}_k}; \boldsymbol{x}_n^k, y_n^k\right) \quad \text{subject to} \quad \overline{\boldsymbol{w}_k} = T_k(\boldsymbol{w}_k); \quad \boldsymbol{w}_k = \boldsymbol{w}^s \quad \forall k \in [m] \tag{1.3}$$

where $\boldsymbol{w}_k$ is the meta model for device k, $\boldsymbol{w}^s$ is the server meta model, $\overline{\boldsymbol{w}_k} = T_k(\boldsymbol{w}_k)$ is the adapted model for device k using its dataset. The problem targets to align the meta models using the server meta model to avoid local overfitting. We refer to Chapter 3 for a precise definition of the adaptation function.

Federated Learning in Heterogeneous Networks

Eq. 1.2 assumes the model architectures to be the same for local and global models through $\boldsymbol{w}_k = \boldsymbol{w}^s$ equality. Practically, the model architecture depends on the capacity of each device. Then, $\boldsymbol{w}_k = \boldsymbol{w}^s$ equality no longer holds since $\{\boldsymbol{w}_k\}_{k \in [m]}$ lives in different dimensions. In this case, we reformulate Eq. 1.2 for device k as,

$$\arg\min_{\boldsymbol{w}_k \in \mathcal{W}} \frac{1}{N_k} \sum_{n \in [N_k]} \ell\left(\boldsymbol{w}_k; \boldsymbol{x}_n^k, y_n^k\right) \quad \text{subject to} \quad \mathcal{R}\left(\boldsymbol{w}_k; \{\boldsymbol{w}_j\}_{j \in [m]}\right) \qquad \forall k \in [m] \tag{1.4}$$

where $\mathcal{R}\left(\boldsymbol{w}_k; \{\boldsymbol{w}_j\}_{j\in[m]}\right)$ forces the same capacity models to be the same and relates device models from different capacities, .i.e $\boldsymbol{w}_k = \boldsymbol{w}_j$ if device k and device j support the same architecture and $\boldsymbol{w}_k \approx \boldsymbol{w}_j$ if device k and device j supports different architectures. We refer to Chapter 4 for a full definition of the relation function.

1.2.2 Online Meta Learning

Online meta-learning aims to model human learning. New tasks arrive in rounds, and they are revealed with limited supervision .i.e a small dataset denoted as $\{\boldsymbol{x}_n^t, y_n^t\}_{n=1}^{N_t}$ is given at round t . The learner tries to learn each task with the available data. Learning from scratch for each task is not feasible since each task has very limited supervision. Instead, the learner should transfer its experience from earlier tasks to new ones. The learner solves the following problem,

$$\arg\min_{\boldsymbol{w}\in\mathcal{W}} \frac{1}{T}\sum_{t\in[T]} \frac{1}{N_t}\sum_{n\in[N_t]} \ell\left(\overline{\boldsymbol{w}}; \boldsymbol{x}_n^t, y_n^t\right) \quad \text{subject to} \quad \overline{\boldsymbol{w}} = T_t(\boldsymbol{w}) \tag{1.5}$$

where datasets, $\{\boldsymbol{x}_n^t, y_n^t\}_{n=1}^{N_t}$, are revealed in rounds so that the the learner does not have direct access to the learning problem in Eq. 1.5. The performance of the learner is measured with regret. We refer to Chapter 5 for more details.

1.2.3 Knowledge Distillation

Distillation considers a setting where a powerful teacher model exists during the training of a student model. The student is given a labeled dataset, .i.e $\{\boldsymbol{x}_n, y_n\}_{n\in[N]}$. The student is encouraged to learn the ground truth labels and to produce teacher-like probability vectors on the training data. Distillation solves the following the problem.

$$\arg\min_{\boldsymbol{w}\in\mathcal{W}} \frac{1}{N}\sum_{n\in[N]} \ell\left(\boldsymbol{w}; \boldsymbol{x}_n, y_n\right) \quad \text{subject to} \quad \mathcal{R}\left(\boldsymbol{w}; \boldsymbol{w}_{\text{teacher}}, \{\boldsymbol{x}_j, y_j\}_{j\in[N]}\right) \tag{1.6}$$

where the constraint forces to align the probability output of the model and the teacher. We give detailed formulation in Chapter 6.

1.3 Challenges

In this section, we highlight the main challenges of the problems we consider in the thesis.

Federated Learning

(McMahan et al., 2017a) give four fundamental challenges in federated learning based on practical concerns. First, the server and device communication is lossy; only a small subset of devices are available each round. Second, the data in the devices are heterogeneous; minimization based on one device dataset does not generalize to other device datasets. Third, there is a massive number of devices in the system and there is very little data in each device. Lastly, the dataset sizes in each device vary. Apart from these challenges, federated learning has a no-data-share policy due to privacy concerns; data can not be transferred between devices and the server. The ultimate goal in federated learning is to train a model with as little as server-to-device transmissions due to communication costs.

Personalized Federated Learning

Personalized federated learning targets the data heterogeneity problem by finding a meta-server model and customizing each model using the user's limited data. Solving for a meta-model introduces new challenges, such as adapting training to meta-model training and handling meta-model bias in the device datasets.

Federated Learning in Heterogeneous Networks

Above federated learning settings inherently assumes a common model architecture for all devices by forcing a consensus on the learned model parameters. However, devices support different model architectures based on their capacity levels. In different architecture scenarios, the definition of consensus, .i.e training a common global model, no longer holds. Transferring data knowledge between different capacity devices becomes a challenge.

Online Meta Learning

Online meta-learning (Finn et al., 2019) consists of learning rounds. A small task dataset that is not enough to learn a model is revealed in each learning round. The learner is expected to extract crucial information from each round and train a model that performs well in all rounds. The main challenge in this setting is to learn a model without storing data from earlier rounds. Storing data from earlier tasks increases memory footprint and leads to practical consequences.

Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) trains a model on a dataset with the help of a powerful teacher model by forcing the model outputs to be similar to that of the teacher model. The challenge in this problem is defining how the teacher helps the model during training. Consider a setting where the teacher is much more powerful than the model. Some of the outputs of the teacher would be over the model's capacity, and merely forcing the model to follow the teacher's all outputs would not help.

1.4 Contributions

We propose novel algorithms for the problems we consider, test our methods with real-world datasets, and show significant performance improvements compared to competitors. We list the main contributions of this thesis in the following paragraphs.

Federated Learning

We propose a method to dynamically regularize each device's local losses to account for the heterogeneity problem. Unlike prior works (Karimireddy et al., 2019), our solution does not increase transmission costs; we transmit only one model between the server and devices. We give convergence proof by being agnostic to the heterogeneity levels.

Personalized Federated Learning

Our solution trains a meta server model that is customized to end users. Motivated by privacy concerns, we further allow devices to hide the class information in their datasets. Our method allows for arbitrary anonymization of the classes in each device and works for arbitrary heterogeneity levels. We provide convergence guarantees on the meta-model.

Federated Learning in Heterogeneous Networks

Motivated by early exit literature (Kaya et al., 2019), we propose a novel method to enforce similarity between different model architectures where we train the simple architecture as a subset of the complex architecture. The information transfer is handled without extra communication costs.

Online Meta Learning

Our novel method allows discarding data from earlier rounds by effectively summarizing its information. Unlike the prior work (Finn et al., 2019), the memory footprint of our approach does not linearly grow with the number of tasks.

Knowledge Distillation

We propose a novel solution where we scaffold the student model to focus on learnable information from the teacher model. Using our method, the student ignores overcapacity information from the teacher model and masters the learnable knowledge.

1.5 Related Work

We present a summary of the related works in this section. We refer to each chapter for a detailed comparison of the prior works.

Federated Learning

Federated learning (Kairouz et al., 2019; Li et al., 2020) solves a distributed optimization problem where the data nodes (devices) are connected to a server and collaboratively train a global model by sharing models and keeping all data local. One key characteristic of federated learning is the non-IID nature of the local datasets (McMahan et al., 2017a). Device users have different preferences, which lead to different local data distributions. Prior work handles the non-IID problem by either controlling the heterogeneity levels in FedAvg (McMahan et al., 2017a) and FedProx (Li et al., 2020a) for convergence or transmitting bias corrections along with the models as in SCAFFOLD (Karimireddy et al., 2019). We propose a novel debiasing method that does not increase per-round transmission costs, and we prove convergence guarantees by being agnostic to the heterogeneity levels.

Personalized Federated Learning

One common server model is trained for all devices in federated learning. However, customizing models based on user needs is a better-motivated practical problem. Personalized federated learning (Chen et al., 2018; Fallah et al., 2020a) changes federated learning where the devices collaboratively train a meta-server model, which is then customized to each device based on its local dataset. Prior work, Per-FedAvg (Fallah et al., 2020a), uses FedAvg-based optimization and MAML-type customization (Finn et al., 2017). Differently, we propose to use heterogeneity agnostic optimization techniques to avoid meta-bias due to the non-IID nature of local datasets. Furthermore, we use the nearest neighbor customization (Snell et al., 2017), which enhances label privacy by allowing users to index their class labels randomly.

Federated Learning in Heterogeneous Networks

A common way of merging different local model information into one server model is to average the local model parameters, which is used in federated learning and personalized federated learning. However, devices can only support the models based on their capacities. Motivated by the mentioned practical problem, federated learning in heterogeneous networks (Diao et al., 2021) proposes to have different architectures in the devices based on their capacities. Prior work, HeteroFL (Diao et al., 2021), changes the server aggregation step where the simple architecture is zero-padded to be in the same dimensions as complex architectures and keeps the device training as in FedAvg. We propose a device training scheme where the complex devices further update the simple networks motivated by early exit literature (Bolukbasi et al., 2017; Kaya et al., 2019). Adapting local optimization to different architectures results in high savings.

Online Meta Learning

Online Meta Learning (Finn et al., 2019) models human learning where an agent sequentially adapts tasks using meta-learning. Each task consists of a small labeled training set that is not sufficient to train a model from scratch. The agent must transfer the meta information between tasks to have a sub-linear regret. Prior work, FTML (Finn et al., 2019), trains a meta-model on all seen task data points in each round which leads to a linear memory footprint. Differently, we propose a method that does not need to store seen datasets and achieves sub-linear regret.

Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) targets a centralized training problem consisting of a supervised dataset, a student model, and a pre-trained, more complex teacher model. The student model severely overfits the supervised dataset without teacher feedback. We can train the student network by distilling information from teacher supervision to improve generalization. Prior work, KD (Hinton et al., 2015), forces the student to follow the prediction probability vector of the teacher model for all examples. Differently, we claim that distilling information on all data points is not helpful as such, there exists some over-capacity points where the student should not follow teacher supervision. Our method selectively distills information by filtering over-capacity examples.

1.6 Organization

We list the content of the chapters in the following.

- Chapter 2 introduces the data heterogeneity problem in federated learning. We propose a solution that handles the heterogeneity problem without extra transmission costs. We present convergence guarantees.

- Chapter 3 targets the personalization problem in federated learning, where the global model is customized to each end user. Our solutions give high-performant personalized models with provable convergence guarantees.

- Chapter 4 further extends federated learning into the setting where clients have different architectures. Supporting different client architectures raises problems regarding transferring knowledge between clients. We propose a method motivated by the ideas from early exit works.

- Chapter 5 considers an online learning setup where datasets arrive in time for one client. We propose a solution that gets optimal regret guarantees with significant performance improvements without storing the data revealed in each round.

- Chapter 6 targets the distillation problem, where we would like to improve the generalization of a model using a powerful teacher. Our solution distinguishes easy vs. over-capacity examples and distills the knowledge of the teacher model using only attainable examples.

- Chapter 7 gives some exciting directions for future research. We use generalization works and state-of-the-art data generation techniques for better training in federated learning.

Chapter 2

Federated Learning in Heterogeneous Data Partitions

In this chapter, we target the standard federated learning (FL) problem. The basic FL problem can be cast as one of empirical minimization of a global loss objective, which is decomposable as a sum of device-level empirical loss objectives. The number of communication rounds, along with the amount of bits communicated per round, has emerged as a fundamental *gold standard* for FL problems. Many mobile and IoT devices are bandwidth constrained, and wireless transmission and reception is significantly more power hungry than computation (Halgamuge et al., 2009). As such schemes that reduce communication are warranted. While distributed SGD is a viable method in this context, it is nevertheless communication inefficient.

A Fundamental Dilemma. Motivated by these ideas, recent work has proposed to push optimization burden onto the devices, in order to minimize amount of communications. Much of the work in this context, propose to optimize the local risk objective based on running SGD over mini-batched device data, analogous to what one would do in a centralized scenario. On the one hand, training models on local data that minimize local empirical loss appears to be meaningful, but yet, doing so, is fundamentally *inconsistent* with minimizing the global empirical loss[1] (Malinovsky

[1] To see this consider the situation where losses are differentiable. As such stationary points for global empirical loss demand that only the sum of the gradients of device empirical losses are zero, and not necessarily that the individual device gradients are zero. Indeed, in statistically heterogeneous situations, such as where we have heterogeneous dominance of classes, stationary points of local empirical functions do not coincide.

et al., 2020; Khaled et al., 2020a). Prior works (McMahan et al., 2017a; Karimireddy et al., 2019; Reddi et al., 2020) attempt to overcome this issue by running fewer epochs or rounds of SGD on the devices, or attempt to stabilize server-side updates so that the resulting fused models correspond to inexact minimizations and can result in globally desirable properties.

Dynamic Regularization. To overcome these issues, we revisit the FL problem, and view it primarily from a communication perspective, with the goal of minimizing communication, and as such allowing for significantly more processing and optimization at the device level, since communication is the main source of energy consumption (Yadav and Yadav, 2016; Latré et al., 2011). This approach, while increasing computation for devices, leads to substantial improvement in communication efficiency over existing state-of-the-art methods, uniformly across the four FL scenarios (unreliable links, massive distribution, substantial heterogeneity, and unbalanced data). Specifically, in each round, we dynamically modify the device objective with a penalty term so that, in the limit, when model parameters converge, they do so to stationary points of the global empirical loss. Concretely, we add linear and quadratic penalty terms, whose minima is consistent with the global stationary point. We then provide an analysis of our proposed FL algorithm and demonstrate convergence of the local device models to models that satisfy conditions for local minima of global empirical loss with a rate of $O\left(\frac{1}{T}\right)$ where T is number of rounds communicated. For convex smooth functions, with m devices, and P devices active per round, our convergence rate for average loss with balanced data scales as $O\left(\frac{1}{T}\sqrt{\frac{m}{P}}\right)$, substantially improving over the state-of-art (SCAFFOLD $O\left(\frac{1}{T}\frac{m}{P}\right)$). For non-convex smooth functions, we establish a rate of $O\left(\frac{1}{T}\frac{m}{P}\right)$.

We perform experiments on both visual and language real-world datasets including MNIST, EMNIST, CIFAR-10, CIFAR-100 and Shakespeare. We tabulate performance studying cases that are reflective of FL scenarios, namely, for (i) varying device participation levels, (ii) massively distributed data, (iii) various levels of heterogeneity, as well as (iv) unbalanced local data settings. Our proposed algorithm, FedDyn, has similar overhead to competing approaches, but converges at a significantly faster rate. This results in a substantial reduction in communication compared to baseline approaches such as conventional FedAvg (McMahan et al., 2017a), FedProx (Li et al., 2020) and SCAFFOLD (Karimireddy et al., 2019), for achieving target accuracy. Furthermore, our approach is simple to implement, requiring far less hyperparameter tuning compared to competing methods.

Contributions. We summarize our main results here.

- We present, FedDyn, a novel dynamic regularization method for FL. Key to FedDyn is a new concept, where in each round the risk objective for each device is dynamically updated so as to ensure that the device optima is asymptotically consistent with stationary points of the global empirical loss,

- We prove convergence results for FedDyn in both convex and non-convex settings, and obtain sharp results for communication rounds required for achieving target accuracy. Our results for convex case improves significantly over state-of-art prior works. FedDyn in theory is unaffected by heterogeneity, massively distributed data, and quality of communication links,

- On benchmark examples FedDyn achieves significant communication savings over competing methods uniformly across various choices of device heterogeneity and device participation on massively distributed large-scale text and visual datasets.

This work is published in (Acar et al., 2021a).

2.1 Related Work

FL is a fast evolving topic, and we only describe closely related approaches here. Comprehensive field studies have appeared in (Kairouz et al., 2019; Li et al., 2020; Wang et al., 2021). The general FL setup involves two types of updates, the server and device, and each of these updates are associated with minimizing some local loss function, which by itself could be updated dynamically over different rounds. At any round, there are methods that attempt to fully optimize or others that propose inexact optimization. We specifically focus on relevant works that address the four FL scenarios (massive distribution, heterogeneity, unreliable links, and unbalanced data) here.

One line of work proposes local SGD (Stich, 2019) based updates, wherein each participating device performs a single local SGD step. The server then averages received models. In contrast to local SGD, our method proposes to minimize a local penalized empirical loss.

FedAvg (McMahan et al., 2017a) is a generalization of local SGD, which proposes a larger number of local SGD steps per round. Still, FedAvg inexactly solves device side optimization. Identifying when to stop minimizing so that one gets a good accuracy-communication trade-off is based on tuning the number of epochs and the learning rate (McMahan et al., 2017a; Li et al., 2020b). Despite the strong empirical performance of FedAvg in IID settings, performance degrades in non-IID scenarios (Zhao et al., 2018).

Several modifications of FedAvg have been proposed to handle non-IID settings. These variants include using a decreasing learning rate (Li et al., 2020b); modifying device empirical loss dynamically (Li et al., 2020a); or modifying server side updates (Hsu et al., 2019; Reddi et al., 2020). Methods that use a decreasing learning rate or customized server side updates still rely on local SGD updates within devices. While these works do recognize the incompatibility of local and global stationary points, their proposed fix is based on inexact minimization. Additionally, in order to establish convergence for non-IID situations, these works impose additional "bounded-non-IID" conditions.

FedProx (Li et al., 2020a) is related to our method. Like us they propose a dynamic regularizer, which is modified based on server supplied models. This regularizer penalizes updates that are far away from the server model. Nevertheless, the resulting regularizer does not result in aligning the global and local stationary points, and as such inexact minimization is warranted, and they do so by carefully choosing learning rates and epochs. Furthermore, tuning requires some knowledge of statistical heterogeneity.

In a similar vein, there are works that augment updates with extra device variables that are also transmitted along with the models (Karimireddy et al., 2019; Shamir et al., 2014). These works prove convergence guarantees through adding device-dependent regularizers. Nevertheless, they suffer additional communication costs and they are not extensively experimented with deep neural networks. Among them, SCAFFOLD (Karimireddy et al., 2019) is a closely related work even though it transmits extra variables and a more detailed comparison is given in Section 2.2.

Another line of distributed optimization methods (Konečnỳ et al., 2016; Makhdoumi and Ozdaglar, 2017; Shamir et al., 2014; Yuan and Ma, 2020; Pathak and Wainwright, 2020; Liang et al., 2019; Li et al., 2020c; Condat et al., 2020) could be considered in this setting. Moreover, there are works that extend analysis of SGD type methods to FL settings (Gorbunov et al., 2020; Khaled et al., 2020b; Li and Richtárik, 2020). However, these algorithms are proposed for full device participation case which fails to satisfy one important aspect of FL. FedSVRG (Konečnỳ et al., 2016) and DANE (Shamir et al., 2014) need gradient information from all devices at each round and they are not directly applicable to partial FL settings. For example, FedDANE (Li et al., 2019a) is a version of DANE that works in partial participation. However, FedDANE performs worse than FedAvg empirically with partial participation (Li et al., 2019a). Similar to these works, FedPD (Zhang et al., 2020) method is proposed in distributed optimization with a different participation notion. FedPD activates either all devices or no devices per round which again fails to satisfy partial participation in FL.

Another set of works aims to decrease communication costs by compressing the transmitted models (Dutta et al., 2019; Mishchenko et al., 2019; Alistarh et al., 2017). They save communication costs through decreasing bit-rate of the transmission. These ideas are complementary to our work and they can be integrated to our proposed solution.

We further cover some of the new works published after our study. (Yang et al., 2022) modify federated learning and give freedom to devices when to participate in server aggregation and how much local computation to do. (Luo et al., 2021) tackle the non-IID problem in federated learning, empirically show that training under non-IID local datasets leads to different bias levels in layers, and propose calibrating the weights with virtual representations. (Zhang et al., 2022) focus on server aggregation and propose a distillation method to obtain the server model using a generator. The proposed methods in (Yang et al., 2022) and (Zhang et al., 2022) are complementary to FedDyn and can be used together.

2.2 Definition and Method

We assume there is a cloud server which can transmit and receive messages from m client devices. Each device, $k \in [m]$ consists of N_k training instances in the form of features, $\boldsymbol{x} \in \mathcal{X}$ and corresponding labels $y \in \mathcal{Y}$ that are drawn IID from a deviceindexed joint distribution, $(\boldsymbol{x}, y) \sim P_k$.

Our objective is to solve

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \left[\ell(\boldsymbol{w}) \triangleq \frac{1}{m} \sum_{k \in [m]} L_k(\boldsymbol{w}) \right]$$

where, $L_k(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_k}[\ell_k(\boldsymbol{w}; (\boldsymbol{x}, y))]$ is the empirical loss of the kth device, and $\boldsymbol{w}$ are the parameters of our neural network, whose structure is assumed to be identical across the devices and the server. We denote by $\boldsymbol{w}^*$ a local minima of the global empirical loss function.

FedDyn Method. Our proposed method, FedDyn, is displayed in Algorithm 1. In each round, $t \in [T]$, a subset of devices $\mathcal{P}_t \subset [m]$ are active, and the server transmits its current model, $\boldsymbol{w}^{t-1}$, to these devices. Each active device then optimizes a local empirical risk objective, which is the sum of its local empirical loss and a penalized risk function. The penalized risk, which is dynamically updated, is based on current local device model, and the received server model:

$$\boldsymbol{w}_k^t = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left[\mathfrak{R}_k(\boldsymbol{w}; \boldsymbol{w}_k^{t-1}, \boldsymbol{w}^{t-1}) \triangleq \; L_k(\boldsymbol{w}) - \langle \nabla L_k(\boldsymbol{w}_k^{t-1}), \boldsymbol{w} \rangle + \frac{\alpha}{2} \|\boldsymbol{w} - \boldsymbol{w}^{t-1}\|^2 \right]. \tag{2.1}$$

Devices compute their local gradient, $\nabla L_k\left(\boldsymbol{w}_k^{t-1}\right)$, recursively, by noting that the first order condition for local optima must satisfy,

$$\nabla L_k(\boldsymbol{w}_k^t) - \nabla L_k(\boldsymbol{w}_k^{t-1}) + \alpha(\boldsymbol{w}_k^t - \boldsymbol{w}^{t-1}) = \mathbf{0} \tag{2.2}$$

Algorithm 1 Federated Dynamic Regularizer - (FedDyn)

```
Input: $T, \boldsymbol{w}^0, \alpha > 0, \nabla L_k(\boldsymbol{w}_k^0) = \mathbf{0}$.
for $t = 1, 2, \dots T$ do
    Sample devices $\mathcal{P}_t \subseteq [m]$ and transmit $\boldsymbol{w}^{t-1}$ to each selected device,
    for each device $k \in \mathcal{P}_t$, and in parallel do
        Set $\boldsymbol{w}_k^t = \operatorname*{argmin}_{\boldsymbol{w}} L_k(\boldsymbol{w}) - \langle \nabla L_k(\boldsymbol{w}_k^{t-1}), \boldsymbol{w} \rangle + \frac{\alpha}{2}\|\boldsymbol{w} - \boldsymbol{w}^{t-1}\|^2$,
        Set $\nabla L_k(\boldsymbol{w}_k^t) = \nabla L_k(\boldsymbol{w}_k^{t-1}) - \alpha\,(\boldsymbol{w}_k^t - \boldsymbol{w}^{t-1})$,
        Transmit device model $\boldsymbol{w}_k^t$ to server,
    end for
    for each device $k \notin \mathcal{P}_t$, and in parallel do
        Set $\boldsymbol{w}_k^t = \boldsymbol{w}_k^{t-1}$, $\nabla L_k(\boldsymbol{w}_k^t) = \nabla L_k(\boldsymbol{w}_k^{t-1})$,
    end for
    Set $\boldsymbol{h}^t = \boldsymbol{h}^{t-1} - \alpha \frac{1}{m}\left(\sum_{k \in \mathcal{P}_t} \boldsymbol{w}_k^t - \boldsymbol{w}^{t-1}\right)$,
    Set $\boldsymbol{w}^t = \left(\frac{1}{|\mathcal{P}_t|}\sum_{k \in \mathcal{P}_t} \boldsymbol{w}_k^t\right) - \frac{1}{\alpha}\boldsymbol{h}^t$
end for
```

Stale devices do not update their models. Updated device models, $\boldsymbol{w}_k^t$, $k \in \mathcal{P}_t$ are then transmitted to server, which then updates its model to $\boldsymbol{w}^t$ as displayed in Algorithm 1.

Intuitive Justification. To build intuition into our method, we first highlight a fundamental issue about the Federated Dynamic Regularizer setup. It is that stationary points for device losses, in general, do not conform to global losses. Indeed, a global stationary point, $\boldsymbol{w}^*$ must necessarily satisfy,

$$\nabla \ell(\boldsymbol{w}^t) \triangleq \frac{1}{m}\sum_{k \in [m]} \nabla L_k(\boldsymbol{w}_*) = \sum_{k \in [m]} \mathbb{E}_{(\boldsymbol{x},y)\sim \mathcal{D}_k} \nabla \ell_k(\boldsymbol{w}_*; (\boldsymbol{x}, y)) = \mathbf{0}. \tag{2.3}$$

In contrast a device's stationary point, $\boldsymbol{w}_k^*$ satisfies, $\nabla L_k(\boldsymbol{w}_k^*) = \mathbf{0}$, and in general due to heterogeneity of data ($P_k \neq P_j$ for $k \neq j$), the individual device-wise gradients are non-zero $\nabla L_k(\boldsymbol{w}_*) \neq \mathbf{0}$. This means that the dual goals of (i) seeking model convergence to a consensus, namely, $\boldsymbol{w}_k^t \to \boldsymbol{w}^t \to \boldsymbol{w}_*$, and (ii) the fact that model updates are based on optimizing local empirical losses is inconsistent[2].

[2] As pointed in related work prior works based on SGD implicitly account for the inconsistency

Dynamic Regularization. Our proposed risk objective in Eq. 2.1 dynamically modifies local loss functions, so that, if in fact local models converge to a consensus, the consensus point is consistent with stationary point of the global loss. To see this, first note that if we initialize at a consensus point, namely, $\boldsymbol{w}_k^{t-1} = \boldsymbol{w}^{t-1}$, we have, $\nabla\mathfrak{R}(\boldsymbol{w}, \boldsymbol{w}_k^{t-1}, \boldsymbol{w}^{t-1}) = \mathbf{0}$ for $\boldsymbol{w} = \boldsymbol{w}^{t-1}$. Thus our choice can be seen as modifying the device loss so that the stationary points of device risk is consistent with server model.

Key Property of Algorithm 1. If local device models converge, they converge to the server model, and the convergence point is a stationary point of the global loss. To see this, observe from Eq 2.2 that if $\boldsymbol{w}_k^t \to \boldsymbol{w}_k^\infty$, it generally follows that, $\nabla L_k(\boldsymbol{w}_k^t) \to \nabla L_k(\boldsymbol{w}_k^\infty)$, and as a consequence, we have $\boldsymbol{w}^t \to \boldsymbol{w}_k^\infty$. In turn this implies that $\boldsymbol{w}_k^\infty \to \boldsymbol{w}^\infty$, i.e., is independent of k. Putting all of this together with our server update equations we have that $\boldsymbol{w}^t$ convergence implies $\boldsymbol{h}^t \to 0$. Now the server state $\boldsymbol{h}^t \triangleq \sum_k \nabla L_k(\boldsymbol{w}_k^t)$, and as such in the limit we are left with $\sum_k \nabla L_k(\boldsymbol{w}_k^t) \to \sum_k \nabla L_k(\boldsymbol{w}^\infty) = \mathbf{0}$. This implies that we converge to a point that turns out to be a stationary point of the global risk.

2.2.1 Analysis of FedDyn

Properties outlined in the previous section, motivates our FedDyn convergence analysis of device and server models. We will present theoretical results for strongly convex, convex and non-convex functions.

Theorem 1 *Assuming a constant number of devices are selected uniformly at random in each round, $|\mathcal{P}_t| = P$, for a suitably chosen of $\alpha > 0$, Algorithm 1 satisfies,*

- *μ strongly convex and L smooth $\{L_k\}_{k=1}^m$ functions,*

$$E\left[\ell\left(\frac{1}{R}\sum_{t=0}^{T-1} r^t \boldsymbol{g}^t\right) - \ell_*\right] =$$

by performing inexact minimization, and additional hyperparameter tuning.

$$O\left(\frac{1}{r^T}\left(\beta\left\|\boldsymbol{w}^0-\boldsymbol{w}_*\right\|^2+\frac{m}{P}\frac{1}{\beta}\left(\frac{1}{m}\sum_{k\in[m]}\left\|\nabla L_k(\boldsymbol{w}_*)\right\|^2\right)\right)\right)$$

- *Convex and L smooth $\{L_k\}_{k=1}^m$ functions,*

$$E\left[\ell\left(\frac{1}{T}\sum_{t=0}^{T-1}\boldsymbol{g}^t\right)-\ell_*\right]=$$
$$O\left(\frac{1}{T}\sqrt{\frac{m}{P}}\left(L\left\|\boldsymbol{w}^0-\boldsymbol{w}_*\right\|^2+\frac{1}{L}\frac{1}{m}\sum_{k\in[m]}\left\|\nabla L_k(\boldsymbol{w}_*)\right\|^2\right)\right)$$

- *Nonconvex and L smooth $\{L_k\}_{k=1}^m$ functions,*

$$E\left\|\nabla\ell(\overline{\boldsymbol{g}}_T)\right\|^2=O\left(\frac{1}{T}\left(L\frac{m}{P}\left(\ell(\boldsymbol{w}^0)-\ell_*\right)+L^2\frac{1}{m}\sum_{k\in[m]}\left\|\boldsymbol{w}_k^0-\boldsymbol{w}^0\right\|^2\right)\right)$$

where $\boldsymbol{g}^t{=}\frac{1}{P}\sum_{k\in\mathcal{P}_t}\boldsymbol{w}_k^t$, $\boldsymbol{w}_*{=}arg\min_{\boldsymbol{w}}\ \ell(\boldsymbol{w})$, $\ell_*{=}\ell(\boldsymbol{w}_*)$, $r{=}\left(1+\frac{\mu}{\alpha}\right)$, $R{=}\sum_{t=0}^{T-1}r^t$ $\beta{=}\max\left(5\frac{m}{P}\mu,30L\right)$ *and* $\overline{\boldsymbol{g}}_T$ *is a random variable that takes values* $\{\boldsymbol{g}^s\}_{s=0}^{T-1}$ *with equal probability.*

Theorem 1 gives rates for strongly convex, convex and nonconvex local losses. For strongly convex and smooth functions, in expectation, a weighted average of active device averages converge at a linear rate. For convex and smooth functions, in expectation, the global loss of active device averages, converges at a rate $O\left(\frac{1}{T}\sqrt{\frac{m}{P}}\right)$. Following convention, this rate is for the empirical loss averaged across devices. As such this rate would hold with moderate data imbalance. In situations with significant imbalance, which scales with data size, these results would have to account for the variance in the amount of data/device. Furthermore, the $\sqrt{\frac{m}{P}}$ factor might appear surprising, but note that our bounds hold under expectation, namely, the error reflects the average over all random choices of devices. Similarly, for nonconvex and smooth functions, in expectation, average of active device models converges to a stationary point at $O\left(\frac{1}{T}\frac{m}{P}\right)$ rate. The expectation is taken over randomness in active device set at each round. Similar to known convergence theorems, the problem dependent constants are related to how good the algorithm is initialized. We refer to Appendix A.2.1 for a detailed proof.

FedDyn vs. SCAFFOLD (Karimireddy et al., 2019). While SCAFFOLD appears to be similar to our method, there are fundamental differences. Practically, SCAFFOLD communicates twice as many bits as FedDyn or Federated Dynamic Regularizer, transmitting back and forth, both a model and its gradient. The 2× increase in bits can be substantial for many low-power IoT applications, since energy consumption for communication dominates computation. Conceptually, we attribute the increased bit-rate to algorithmic differences. At the device-level, our modified risk incorporates a linear term, $\nabla L_k(\boldsymbol{w}_k^t)$ (which we can compute readily (Eq. 2.2)). Applying our perspective to SCAFFOLD, in full participation setting, we see SCAFFOLD as replacing our linear term $\langle \nabla L_k(\boldsymbol{w}_k^t), \boldsymbol{w}\rangle$ with $\left\langle \nabla L_k(\boldsymbol{w}^t) - \frac{1}{m}\sum_{k\in[m]} \nabla L_k(\boldsymbol{w}_k^t), \boldsymbol{w}\right\rangle$. While $\nabla L_k(\boldsymbol{w}^t)$ can be locally computed, after $\boldsymbol{w}^t$ is received, the term $\frac{1}{m}\sum_{k\in[m]} \nabla L_k(\boldsymbol{w}_k^t)$ is unknown and must be transmitted by the server, leading to increased bit-rate. Note that this is unavoidable, since ignoring this term, leads to freezing device updates (optimizing $L_k(\boldsymbol{w}) - \langle \nabla L_k(\boldsymbol{w}^t), \boldsymbol{w} - \boldsymbol{w}^t\rangle + \frac{\alpha}{2}\|\boldsymbol{w} - \boldsymbol{w}^t\|^2$ results in $\boldsymbol{w} = \boldsymbol{w}^t$). This extra term is a surrogate for $\nabla \ell(\boldsymbol{w}^t)$, which is unavailable. As such we believe that these differences are responsible for FedDyn's improved rate (in rounds) in theory as well as practice.

Finally, apart from conceptual differences, there are also implementation differences. SCAFFOLD runs SGD, and adapts hyperparameter tuning for a given number of rounds to maximize accuracy. In contrast, our approach, based on exact minimization, is agnostic to specific implementation, and as such we utilize significantly less tuning.

2.3 Experiments

Our goal in this section is to evaluate FedDyn against competing methods on benchmark datasets for various FL scenarios. Our results will highlight trade-offs and benefits of our exact minimization relative to prior inexact minimization methods. To ensure a fair comparison, the usual SGD procedure is adapted for the FedDyn algorithm in the device update as in FedAvg rather than leveraging an off the shelf optimization solver. We provide a brief description of the datasets and the models used in the experiments. A detailed description of our setup can be found in Appendix A.1.1. Partial participation was handled by sampling devices at random in each round independent of previous rounds.

Datasets. We used benchmark datasets with the same train/test splits as in previous works (McMahan et al., 2017a; Li et al., 2020a) which are MNIST (LeCun et al., 1998), CIFAR-10, CIFAR-100 (Krizhevsky, 2009), a subset of EMNIST (Cohen et al., 2017) (EMNIST-L), Shakespeare (Shakespeare, 1994) as well as a synthetic dataset. The IID split is generated by randomly assigning datapoints to the devices. The Dirichlet distribution is used on the label ratios to ensure uneven label distributions among devices for non-IID splits as in (Yurochkin et al., 2019). For example, in MNIST, 100 device experiments, each device has about 5 and 3 classes that consume 80% of local data at Dirichlet parameter settings of 0.6 and 0.3 respectively. To generate unbalanced data, we sample the number of datapoints from a lognormal distribution. Controlling the variance of lognormal distribution gives unbalanced data. For instance, in CIFAR-10, 100 device experiments, balanced and unbalanced data settings have standard deviation of device sample size of 0 and 0.3 respectively.

Models. We use fully-connected neural network architectures for MNIST and EMNIST-L with 2 hidden layers. The number of neurons in the layers are 200 and 100; and the models achieve 98.4% and 95.0% test accuracy in MNIST and EMNIST-

Table 2.1: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for moderate and large number of devices in IID and Dirichlet .3 settings. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Device Number	Dataset	Accuracy	FedDyn	SCAFFOLD	FedAvg	FedProx
Moderate	**IID**					
	CIFAR-10	84.5	637	1852(2.9×)	1000+(>1.6×)	1000+(>1.6×)
		82.3	240	512(2.1×)	994(4.1×)	825(3.4×)
	CIFAR-100	51.0	522	1854(3.6×)	1000+(>1.9×)	1000+(>1.9×)
		40.9	159	286(1.8×)	822(5.2×)	873(5.5×)
	Dirichlet (.3)					
	CIFAR-10	82.5	444	1880(4.2×)	1000+(>2.3×)	1000+(>2.3×)
		80.7	232	594(2.6×)	863(3.7×)	930(4.0×)
	CIFAR-100	51.0	561	1884(3.4×)	1000+(>1.8×)	1000+(>1.8×)
		42.3	170	330(1.9×)	959(5.6×)	882(5.2×)
Massive	**IID**					
	CIFAR-10	80.0	840	4000+(>4.8×)	2000+(>2.4×)	2000+(>2.4×)
		62.3	305	928(3.0×)	1277(4.2×)	1274(4.2×)
	CIFAR-100	50.1	1445	3982(2.8×)	2000+(>1.4×)	2000+(>1.4×)
		38.3	477	1408(3.0×)	1997(4.2×)	1974(4.1×)
	Dirichlet (.3)					
	CIFAR-10	80.0	831	4000+(>4.8×)	2000+(>2.4×)	2000+(>2.4×)
		70.6	350	2138(6.1×)	1962(5.6×)	1517(4.3×)
	CIFAR-100	47.0	969	4000(4.1×)	2000+(>2.1×)	2000+(>2.1×)
		39.9	467	2266(4.9×)	1913(4.1×)	1794(3.8×)

L respectively. The model used for MNIST is the same as used in (McMahan et al., 2017a). For CIFAR-10 and CIFAR-100, we use a CNN model, similar to (McMahan et al., 2017a), consisting of 2 convolutional layers with 64 5×5 filters followed by 2 fully connected layers with 394 and 192 neurons, and a softmax layer. The model achieves 85.2% and 55.3% test accuracy for CIFAR-10 and CIFAR-100 respectively. For the next character prediction task (Shakespeare), we use a stacked LSTM, similar to (Li et al., 2020a). This architecture achieves a test accuracy of 50.8% and 51.2% in IID and non-IID settings respectively. Both IID and non-IID performances are reported since splits are randomly regenerated from the entire Shakespeare writing.

Hence centralized data and the centralized model performance is different.

In passing, we note that while the accuracies reported are state-of-art for our chosen models, higher capacity models can achieve higher performance on these datasets. As such, our aim is to compare the relative performance of these models in FL using FedDyn and other strong baselines.

Comparison of Methods. We report the performance of FedDyn, SCAFFOLD, FedAvg and FedProx on synthetic and real datasets. We also experimented with distributed SGD, where devices in each round compute gradients on the server supplied model on local data, and communicate these gradients. Its performance was not competitive relative to other methods. Therefore, we do not tabulate it here. We cover synthetic data generation and its results in Appendix A.1.1.

The standard goal in FL is to minimize amount of bits transferred. For this reason, we adopt the number of models transmitted to achieve a target accuracy as our metric in our comparisons. This metric is different than comparing communication rounds since not all methods communicate the same amount of information per round. FedDyn, FedAvg and FedProx transmit/receive the same amount of models for a fixed number of rounds whereas SCAFFOLD costs twice due to transmission of states. We compare algorithms for two different accuracy levels which we pick them to be close to performance obtained by centralizing data. Along with transmission costs of each method, we report the communication savings of FedDyn compared to each baseline in parenthesis. For methods that could not achieve aimed accuracy within the communication constraint, we append transmission cost with + sign. We observe FedDyn results in communication savings compared to all baselines to reach a target accuracy. We test FedDyn under the four characteristic properties of FL which are partial participation, large number of devices, heterogeneous data, and unbalanced data.

Moderate vs. Large Number of Devices. *FedDyn significantly outperforms competing methods in the practically relevant massively distributed scenario.* We report the performance of FedDyn on CIFAR-10 and CIFAR-100 with moderate and large number of devices in Table 2.1, while keeping the participation level constant (10%) and the data amounts balanced. Specifically, the moderately distributed setting has 100 devices with 500 images per device. The massively distributed setting has 1000 devices with 50 images per device for CIFAR-10, as well as 500 devices with 100 images per device for CIFAR-100. In each distributed setting, the data is partitioned in both IID and non-IID (Dirichlet 0.3) fashion. FedDyn leads to substantial transmission reduction in each of the regimes.

First, the communication saving in the massive setting is significantly larger relative to the moderate setting. Compared to SCAFFOLD, FedDyn leads to 4.8× and 2.9× gains respectively on CIFAR-10 IID setting. SCAFFOLD is not able to achieve 80% within 2000 rounds in the massive setting (shown in Figure A·4), thus actual saving is more than 4.8×. Similar trend is observed in the non-IID setting of CIFAR-10 and CIFAR-100. Second, all the methods require more communications to achieve a reasonable accuracy in the massive setting as the dataset is more decentralized. For instance, it takes FedDyn 637 rounds to achieve 84.5% with 100 devices, while it takes 840 rounds to achieve 80.0% with 1000 devices. Similar trend is observed for CIFAR-100 and other methods. FedDyn always achieves the target accuracy with fewer rounds and thus leads to significant saving. Third, a higher target accuracy may result in a greater saving. For instance, the saving relative to SCAFFOLD increases from 3× to 4.8× in the CIFAR-10 IID massive setting. We may attribute this to the fact that FedDyn aligns device functions to global loss and efficiently optimizes the problem.

Full vs. Partial Participation Levels. *FedDyn outperforms baseline methods* *across different device participation levels.* We consider different device participation levels with 100 devices and balanced data in Table 2.2 & 2.3 where part of CIFAR-10 and CIFAR-100 results are omitted since they are reported in moderate number of devices section of Table 2.1. The Shakespeare non-IID results are separately shown, since it has a natural non-IID split which does not conform with the Dirichlet distribution. The communication gain, with respect to best baseline, increases with greater participation levels from 2.9× to 9.4×; 4.0× to 12.8× and 4.2× to 7.9× for CIFAR-10 in different device distribution settings. We observe a similar performance increase in full participation for most of the datasets. This validates our hypothesis that FedDyn more efficiently incorporates information from all devices compared to other methods, and results in more savings in full participation. Similar to previous results, a greater target accuracy gives a greater savings in most of the settings. We also report results for 1% participation regime with different device distribution settings (See Table A.3 in Appendix A.1.1).

Balanced vs. Unbalanced Data. *FedDyn is more robust to unbalanced data than competing methods.* We fix number of devices (100) and participation level (10%) and consider effect of unbalanced data (Table A.2 (Appendix A.1.1)). FedDyn achieves 4.3× gains over the best competitor, SCAFFOLD to achieve the target accuracy. As before, gains increase with the target accuracy.

IID vs. non-IID Device Distribution. *FedDyn outperforms baseline methods across different device distribution levels.* We consider heterogeneous device distributions in the context of varying device numbers, participation levels and balanced-unbalanced settings in Table 2.1, 2.2, 2.3 and A.2 (Appendix A.1.1) respectively. Device distributions become more non-IID as we go from IID, Dirichlet .6 to Dirichlet .3 splits which makes global optimization problem harder. We see a clear effect of this change in Table 2.3 for 10% participation level and in Table A.2 for unbalanced setting. For instance, increasing non-IID level results in a greater communication saving such as from 2.9×, 4.0× to 4.2× in CIFAR-10 10% participation. Similar statement holds for MNIST, EMNIST-L and Shakespeare in Table 2.3 and for CIFAR-10 unbalanced setting in Table A.2. We do not observe a significant difference in savings for full participation setting in Table 2.2.

Summary. Overall, FedDyn consistently leads to substantial communication savings compared to baseline methods uniformly across various FL regimes of interest. We realize large gains in the practically relevant massively distributed data setting.

Table 2.2: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for 100% participation regimes in the IID, non-IID settings. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Dataset	Accuracy	FedDyn	SCAFFOLD	FedAvg	FedProx
IID					
CIFAR-10	85.0	198	1860(9.4×)	1000+(>5.1×)	1000+(>5.1×)
	81.4	67	320(4.8×)	754(11.3×)	655(9.8×)
CIFAR-100	51.0	259	1744(6.7×)	1000+(>3.9×)	1000+(>3.9×)
	39.4	55	172(3.1×)	1000+(>18.2×)	741(13.5×)
MNIST	98.2	38	72(1.9×)	194(5.1×)	445(11.7×)
	97.2	9	18(2.0×)	31(3.4×)	28(3.1×)
EMNIST-L	94.6	65	414(6.4×)	307(4.7×)	1000+(>15×)
	93.6	16	36(2.2×)	66(4.1×)	62(3.9×)
Shakespeare	46.4	33	74(2.2×)	96(2.9×)	113(3.4×)
	45.4	28	64(2.3×)	59(2.1×)	56(2.0×)
Dirichlet (.6)					
CIFAR-10	84.0	148	1890(12.8×)	1000+(>6.8×)	1000+(>6.8×)
	80.3	64	392(6.1×)	869(13.6×)	724(11.3×)
CIFAR-100	51.0	468	1838(3.9×)	1000+(>2.1×)	1000+(>2.1×)
	40.6	73	206(2.8×)	998(13.7×)	592(8.1×)
MNIST	98.1	39	108(2.8×)	157(4.0×)	416(10.7×)
	97.1	11	24(2.2×)	38(3.5×)	34(3.1×)
EMNIST-L	94.9	207	552(2.7×)	410(2.0×)	1000+(>4.8×)
	93.9	20	42(2.1×)	73(3.6×)	61(3.0×)
Dirichlet (.3)					
CIFAR-10	83.5	223	1762(7.9×)	1000+(>4.5×)	1000+(>4.5×)
	80.2	70	504(7.2×)	705(10.1×)	1000+(>14.3×)
CIFAR-100	50.5	405	1940(4.8×)	1000+(>2.5×)	1000+(>2.5×)
	41.0	80	224(2.8×)	911(11.4×)	1000+(>12.5×)
MNIST	98.1	35	76(2.2×)	313(8.9×)	458(13.1×)
	97.1	10	24(2.4×)	49(4.9×)	44(4.4×)
EMNIST-L	94.5	65	210(3.2×)	492(7.6×)	1000+(>15×)
	93.5	23	46(2.0×)	78(3.4×)	69(3.0×)
Non-IID					
Shakespeare	47.3	33	70(2.1×)	134(4.1×)	150+(>4.5×)
	46.3	28	62(2.2×)	53(1.9×)	64(2.3×)

Table 2.3: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for 10% participation regimes in the IID, non-IID settings. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Dataset	Accuracy	FedDyn	SCAFFOLD	FedAvg	FedProx
IID					
MNIST	98.2	100	142(1.4×)	588(5.9×)	362(3.6×)
	97.2	31	52(1.7×)	49(1.6×)	43(1.4×)
EMNIST-L	94.6	104	160(1.5×)	330(3.2×)	210(2.0×)
	93.6	58	84(1.4×)	69(1.2×)	65(1.1×)
Shakespeare	46.9	63	94(1.5×)	138(2.2×)	190(3.0×)
	45.9	56	76(1.4×)	96(1.7×)	75(1.3×)
Dirichlet (.6)					
CIFAR-10	83.5	403	1618(4.0×)	1000+(>2.5×)	1000+(>2.5×)
	81.3	189	486(2.6×)	977(5.2×)	943(5.0×)
CIFAR-100	51.0	521	1910(3.7×)	1000+(>1.9×)	1000+(>1.9×)
	41.6	170	302(1.8×)	931(5.5×)	748(4.4×)
MNIST	98.1	129	194(1.5×)	581(4.5×)	361(2.8×)
	97.1	37	60(1.6×)	57(1.5×)	57(1.5×)
EMNIST-L	94.9	192	296(1.5×)	306(1.6×)	1000+(>5.2×)
	93.9	55	102(1.9×)	95(1.7×)	86(1.6×)
Dirichlet (.3)					
MNIST	98.2	90	208(2.3×)	428(4.8×)	858(9.5×)
	97.2	37	68(1.8×)	76(2.1×)	61(1.6×)
EMNIST-L	94.4	107	178(1.7×)	804(7.5×)	1000+(>9.3×)
	93.4	58	100(1.7×)	81(1.4×)	86(1.5×)
Non-IID					
Shakespeare	47.6	63	102(1.6×)	169(2.7×)	133(2.1×)
	46.6	56	82(1.5×)	80(1.4×)	66(1.2×)

Chapter 3

Personalized Federated Learning

In this chapter, we consider the personalization of FL to the end user. In FL, data stored on the edge devices is statistically heterogeneous, namely, different devices/users have different data. As a result, naively fusing independently trained device models can result in significant bias and poor performance. While a number of previous works (see (Karimireddy et al., 2019)) have proposed methods to overcome the effects of statistical heterogeneity, the goal of these works have remained the same, namely, to realize a single model at termination time, which works as well as a learner with *centralized* test data. In particular, the performance is measured with respect to the combined test data of all of the devices.

While FL optimizes centralized performance, this metric may not be meaningful from the user's perspective. An end user, after all, will have personal objectives and interests. Therefore, a more suitable metric is to measure model performance against the user's custom test data[1]. Variability in user objectives include assigning increased importance to specific classes, variability in user tasks (word completion for native vs. foreign speakers), and requiring privacy/anonymity of class predictions.

We propose a novel federated training approach based on meta-learning, which allows for sample-efficient customization of a centralized model to suit an end user's objectives. That meta-learning approaches are well-suited for our customization scenario is not surprising, and has been leveraged in prior works (Chen et al., 2018;

[1] As a case in point consider two users, one whose experiences involve wild animals, while the other is primarily interested in domestic pets.

Jiang et al., 2019; Fallah et al., 2020b). Indeed, in our setup, each user is associated with a different task, and our goal, as in meta-learning, is to train a meta-model on a variety of tasks, such that this model can be rapidly re-purposed to solve new or existing tasks using only a few training examples.

Challenges. Personalized Federated Learning (PFL) presents two fundamental challenges. First, each client is associated with a personal task, and tasks across different clients exhibit significant statistical variability. In the conventional approach, such as meta-learning or multi-task learning, one leverages data across different tasks to build initial models (meta-model) that serve as a basis to refine and specialize to the presented task. However, since FL prohibits data sharing, this approach is no longer feasible. In this context, (Fallah et al., 2020b) propose to utilize federated averaging to overcome the no-data-sharing constraint, and utilize model-agnostic meta-learning (MAML) (Finn et al., 2017) to personalize to a specific user. Nevertheless, it is well-known that federated averaging performs poorly with statistically heterogeneous data, and results in significantly biased models in these contexts. To overcome these drawbacks, (Fallah et al., 2020b) impose statistical constraints on task and dataset variability across devices, which, in practice, maybe unrealistic.

In contrast, our proposed PFL approach allows for arbitrary variability in user tasks. To overcome task/dataset biases during meta-training, we propose a novel federated meta-learning method, which is based on dynamically modifying device loss functions in each round, so that the resulting meta-model is relatively unbiased towards any user. The proposed dynamic modification of loss is rooted in the rich theory of distributed optimization (Gabay and Mercier, 1976; Makhdoumi and Ozdaglar, 2017; Hong et al., 2016; Shamir et al., 2014), where one attempts to solve a distributed constrained problem through sequentially updated penalty functions. We focus on deep neural networks, and consider two meta-learning approaches: one based on MAML, which requires no additional parameters for customization, and the other based on ProtoNet (Snell et al., 2017), where the meta-model serves as a feature representation to train task customized classifiers. While MAML-based PFL performs well in most cases, ProtoNet-based PFL is particularly effective in cases that require generalization to new tasks, or cases that require anonymization of class names across devices. We derive convergence results for our algorithm, which is agnostic to task heterogeneity across devices in both full- and partial-participation settings. We also perform extensive experiments to empirically evaluate our method on real world datasets, and show that our method significantly outperforms prior works.

Contributions

- We propose a new algorithm, PFL, for personalized federated learning and show its convergence guarantees.
- We propose to extend Proto (Snell et al., 2017) meta adaptation in the personalized federated learning setup.
- We perform extensive empirical evaluations of PFL, as well as Proto adaptation and compare it to the baselines.
- During evaluation, we consider test performance of each user. Based on the individual end-user needs, we observe performance metrics such as average, best and worst device performance. We observe that PFL leads to significant communication savings.

This work is published in (Acar et al., 2021b).

3.1 Related Work

Meta learning. Meta learning is defined as 'learning to learn' (Thrun and Pratt, 2012). In this concept, the aim of the meta learner is to learn how to learn new tasks.

This paradigm is motivated from human learning where humans are able to transfer their experience from previous task instances into new tasks. Thus, meta learning is thought of as an important step towards future machine learning research (Lake et al., 2017). We refer to extensive surveys (Vanschoren, 2018; Hospedales et al., 2022) for a detailed discussion and we discuss closely related work here.

Conceptually, a meta learner learns a new algorithm for every task. There are many efforts to formulate the meta learning problem. One line of research proposes a non parametric meta adaptation (Vinyals et al., 2016; Snell et al., 2017). For instance, prototypical adaptation (Snell et al., 2017) is an extension of the non parametric k nearest neighbor method, where adaptation is based on the class clusters obtained using the available training dataset of the task. Another line of research views meta adaptation as a black box optimization and finds the adaptation based on the state of the meta learner (Santoro et al., 2016; Mishra et al., 2017; Ravi and Larochelle, 2017). For example, in (Ravi and Larochelle, 2017), the authors propose a meta algorithm in which the meta adaptation is obtained from the current state of a meta LSTM. Lastly, there are works in which the adaption is a fixed optimization procedure and the meta learning problem can be optimized using standard gradient descent techniques (Finn et al., 2017; Antoniou et al., 2018; Li et al., 2017). The most popular adaptation is MAML (Finn et al., 2017). In MAML, the meta model is customized by having one gradient descent update on the available task training data. Different from standard meta learning, personalized federated learning extends meta learning to the distributed learning scenario.

Personalized federated learning. We focus on customizing federated learning toward the end user objective, which can be termed federated meta learning (Chen et al., 2018) or personalized federated learning (Fallah et al., 2020b). This relatively new concept extends federated learning to the scenario where the server is required to find a good meta model in which a simple transformation using device data leads to a well performing personalized device model. For instance, Per-FedAvg (Fallah et al., 2020b) is proposed where the MAML meta transformation is used for personalization and the server model is optimized with FedAvg. Along the same lines (Jiang et al., 2019) proposes to use FedAvg, but use SGD with momentum for their updates. Recently, FedFomo (Zhang et al., 2021a) is proposed where there are n server meta models. Transmitting n models increases the communication costs of one round. Different from these methods, we propose PFL as a solution that has one server meta model, significantly reduces communication costs, and unlike FedAvg does not require strong constraints on statistical heterogeneity.

We further cover some of the new works published after our study. (Oh et al., 2022) propose a personalized federated learning method where clients share the same feature extractor and customize the classification layers based on their dataset. (Ma et al., 2022) develop a method that personalizes aggregation of device models in the server using one hypernetwork per device. (Achituve et al., 2021) propose to use Gaussian process for personalization through a common kernel function among the clients parametrized by a DNN.

3.2 Definition and Method

Our setting consists of one server and m devices. The server can send and receive models from devices without sharing data instances. In device i, there are N_i datapoints with features $\boldsymbol{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ which are drawn from a device specific distribution $(\boldsymbol{x}, y) \sim p_i$ denoted as $D_i = \{(\boldsymbol{x}_i^j, y_i^j)\}_{j=1}^{N_i}$. Each device customizes a device specific model from the server model using their dataset. This transformation for device i is modeled as $T_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$ where $\overline{\boldsymbol{w}}_i = T_i(\boldsymbol{w})$ corresponds to the personalized model transformed from meta model $\boldsymbol{w}$ for device i. We define our objective as,

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}\left[F(\boldsymbol{w})\triangleq\frac{1}{m}\sum_{i\in[m]}f_i\left(\overline{\boldsymbol{w}}_i\right)\right],\quad \overline{\boldsymbol{w}}_i=T_i(\boldsymbol{w}) \tag{OPT}$$

where $\boldsymbol{w}$ is the parameter set of the NN used, $\overline{\boldsymbol{w}}_i$ is the personalized model for device i, $f_i = E_{\{\boldsymbol{x},y\}\sim p_i}L\left((\boldsymbol{x},y);\boldsymbol{w}\right)$ is the loss obtained at device i and L is the loss function with respect to one data tuple.

Transformation function. Our objective is a generic objective and depends on the transformation function denoted as T_i for device i. There are different proposed transformation functions within meta learning area. For example, MAML transformation (Finn et al., 2017) is introduced as,

$$T_i(\boldsymbol{w})=\boldsymbol{w}-\eta\nabla\hat{f}_i(\boldsymbol{w})$$

where η is meta learning rate and $\hat{f}_i$ is the empirical loss function of the dateset as $\hat{f}_i(\boldsymbol{w}) = \frac{1}{N_i}\sum_{j=1}^{N_i}L\left((\boldsymbol{x}_i^j,y_i^j);\boldsymbol{w}\right)$. We can summarize MAML as doing one gradient descent update to personalize the meta model. Per-FedAvg (Fallah et al., 2020b) extends MAML transformation to personalized federated learning where it uses MAML meta transformation with FedAvg optimization (McMahan et al., 2017a).

We propose to use another meta transformation named as prototypical adaptation (Snell et al., 2017). Prototypical transformation constructs class representations using the meta model and the data on the device where a representation of a feature $\boldsymbol{x}$ with respect to model $\boldsymbol{w}$ is defined as $\boldsymbol{r}_{\boldsymbol{w}}(\boldsymbol{x})$. For each class label $k\in\mathcal{Y}$, the class representation is obtained by averaging the representations of this class instances as,

$$\boldsymbol{c}_{\boldsymbol{w}}^{i,k}=\frac{1}{|S_i^k|}\sum_{\boldsymbol{x}\in S_i^k}\boldsymbol{r}_{\boldsymbol{w}}(\boldsymbol{x})$$

where $S_i^k=\{(\boldsymbol{x},y):y=k,(\boldsymbol{x},y)\in D_i\}$ is the set of training data instances that are labeled as k in device i. Prototypical transformation is a non parametric adaptation and the customized device model labels the test data with the closest class representation as,

$$\underset{k \in \mathcal{Y}}{\operatorname{argmin}}\ d\left(\boldsymbol{c}_{\boldsymbol{w}}^{i,k}, \boldsymbol{r}_{\boldsymbol{w}}(\boldsymbol{x})\right)$$

where $\boldsymbol{w}$ is the meta model, $\boldsymbol{x}$ is a test point and $d(.,.)$ is a distance function.

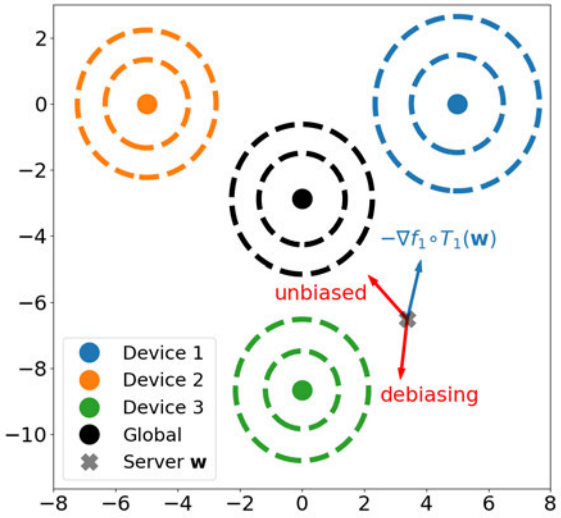


Figure 3·1: Toy example with three devices in a two dimensional parameter space. Only device 1 is active in the current round. In (Fallah et al., 2020b) algorithm, the model is pulled towards device 1's local minima due to the bias discrepancy. The debiasing and the correct unbiased directions are needed for convergence.

Device Bias in (Fallah et al., 2020b). (Fallah et al., 2020b) adapts MAML transformation and applies FedAvg algorithm on transformed local objectives. Namely, in each communication round, the server sends the current server meta model to the participating devices. Each active device runs SGD updates on the server model using gradient of the device specific meta loss, $\nabla f_i \circ T_i$ where T_i is MAML adaptation. Then, the device meta model is transmitted to the server and the received models are averaged.

Device Bias. As we noted in OPT, we are interested in solving the average meta loss functions of all devices. However, devices do not have access to the global loss, they have access to their meta loss function. This leads to a misalignment between the local meta losses and the global loss, because their optimal solutions are different, i.e., $\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) \neq \min_{\boldsymbol{w}} F(\boldsymbol{w})$. Consequently, local bias exhibited by device meta losses leads to global convergence issues. To avoid these issues, (Fallah et al., 2020b) proposes to limit the variability among device meta objectives.

A Toy Example. We visualize device biases for a three device example, where the loss functions are parameterized in a two dimensional space. Figure 3·1 shows contour plots of each device meta functions, $f_i \circ T_i$ as well as the global loss, F. The corresponding optimal meta models are shown with circles. We denote the current server model with × mark. Consider the case where only device 1 is active in the current round.

According to (Fallah et al., 2020b), the server sends the current model to device 1. The model is updated based on the local gradients, $\nabla f_1 \circ T_1$. As seen in the plot, the gradient pulls the server model in a different direction of the global minima. This is an example of the bias discrepancy as such the correct gradient information, shown as unbiased direction is different from the device gradient information. The discrepancy forces (Fallah et al., 2020b) to control the distance between device minima and the global minima for convergence.

We propose the concept of debiasing, where we explicitly debias the local objectives and orient the local loss towards the global objective as shown with red arrows in the figure 3·1.

Debiasing Local Objectives. We debias the local objective $f_i \circ T_i$ to the first-order, and introduce a quadratic regularizer. To build intuition, let us consider the following local objective,

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \langle \nabla f_i \circ T_i(\boldsymbol{w}^*), \boldsymbol{w} \rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^* \right\|^2 \tag{3.1}$$

where α is a hyperparameter and $\boldsymbol{w}^*$ is a stationary point of OPT. For a solution, $\boldsymbol{w}'$, we can write the first order condition as $\nabla f_i \circ T_i(\boldsymbol{w}') - \nabla f_i \circ T_i(\boldsymbol{w}^*) + \alpha (\boldsymbol{w}' - \boldsymbol{w}^*) = \boldsymbol{0}$. We see that $\boldsymbol{w}^*$ satisfies the condition. As such the objective is no longer biased towards the device minima. In particular, $\langle \nabla f_i \circ T_i(\boldsymbol{w}^*), \boldsymbol{w} \rangle$ term debiases the loss $f_i \circ T_i$ so that the gradient is not necessarily pointed towards the device minima. However, this is not strictly feasible, since this objective would require that the devices have access to the optimal solution $\boldsymbol{w}^*$.

A Feasible Surrogate. We consider the server meta model $\boldsymbol{w}^t$ in round communication t as a surrogate for $\boldsymbol{w}^*$, which results in the following objective,

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \left\langle \nabla f_i \circ T_i(\boldsymbol{w}^t), \boldsymbol{w} \right\rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2 .$$

We note that if the server model converges to a stationary point of OPT, then we recover the modified objective as in 3.1. Different from the previous objective, devices can construct the current objective since they receive the server model. However, the objective does not have the correct direction for the global loss because we observe that $\boldsymbol{w}^t$ is among one of the stationary points of the objective. This freezes the update so that device models are stuck at the server model. To circumvent this issue, we consider,

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \left\langle \nabla f_i \circ T_i(\boldsymbol{w}^t), \boldsymbol{w} \right\rangle + \left\langle \frac{1}{m} \sum_{j \in [m]} \nabla f_j \circ T_j(\boldsymbol{w}^t), \boldsymbol{w} \right\rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2 \tag{3.2}$$

where the gradient of the global loss, OPT, appears as a linear term. In Figure 3·1,

Algorithm 2 Personalized Federated Learning, PFL

```
**Input:** $T, \boldsymbol{w}^1, \boldsymbol{g}_i^1 = \boldsymbol{g}^1 = \mathbf{0}, K, \beta, \alpha$
for $t = 1, 2, \ldots T$ do
    Sample active device set $\mathcal{P}_t \subseteq [m]$ of $P$ devices,
    for $i \in \mathcal{P}_t$ do
        Receive $\boldsymbol{w}^t$ from server (& $\boldsymbol{g}^t$ in PFLScaf),
        $\mathcal{R}_i^t(\boldsymbol{w})$=Regularizer($\boldsymbol{w}, \boldsymbol{w}^t, \boldsymbol{g}_i^t$) (& $\boldsymbol{g}^t$ in PFLScaf),
        $\boldsymbol{w}_i^{t+1}, \boldsymbol{g}_i^{t+1}$ = Update($\boldsymbol{w}^t, K, \beta, D_i, \mathcal{R}_i^t$)
        Send $\boldsymbol{w}_i^{t+1}$ back to server (& $\boldsymbol{g}_i^{t+1}$ in PFLScaf),
    end for
    Freeze stale devices $\boldsymbol{w}_i^{t+1}$=$\boldsymbol{w}_i^t$, $\boldsymbol{g}_i^{t+1}$=$\boldsymbol{g}_i^t$ $\forall i \notin \mathcal{P}_t$
    Server
    PFLDyn: $\boldsymbol{w}^{t+1}$ = Update$_1$ $\left(\{\boldsymbol{w}_i^{t+1}\}_{i \in \mathcal{P}_t}\right)$,
    PFLScaf: $\boldsymbol{w}^{t+1}, \boldsymbol{g}^{t+1}$=Update$_2$ $\left(\{\boldsymbol{w}_i^{t+1}, \boldsymbol{g}_i^{t+1}\}_{i \in \mathcal{P}_t}\right)$.
end for
```

the direction of these two terms are shown in red arrows where the first linear term debiases the current loss and the second linear term results in the correct gradient direction.

Devices can construct the first linear term and the quadratic term with the server model. However, the second linear term depends on all device losses so that it is still not feasible. As a surrogate for this term, we transmit the gradient information of the current server meta model to the server. The server aggregates the gradient information from all devices and constructs the second term. Finally, devices need the server to send the average gradient information, $\frac{1}{m}\sum_{j \in [m]} \nabla f_j \circ T_j(\boldsymbol{w}^t)$, along with the server model. In summary, devices and the server communicates two models to construct this objective.

An Alternative Surrogate. Instead of using $\boldsymbol{w}^t$ model in the linear terms, we can as well use the recent device models, $\boldsymbol{w}_i^t$, to construct them. Then, the objective in

Eq. 3.2 becomes,

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \left\langle \nabla f_i \circ T_i(\boldsymbol{w}_i^t), \boldsymbol{w} \right\rangle + \left\langle \frac{1}{m} \sum_{j \in [m]} \nabla f_j {\circ} T_j(\boldsymbol{w}_j^t), \boldsymbol{w} \right\rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} {-} \boldsymbol{w}^t \right\|^2 . \tag{3.3}$$

We note that if the device models, $\boldsymbol{w}_i^t$s, converge to the stationary point of OPT, we still recover the proposed the modification as in Eq. 3.1. This seemingly subtle modification allows the server to transmit only one model instead of two models. This extension is further discussed in the subsequent section.

PFL Algorithms. We propose to use two recent algorithms as SCAFFOLD and FedDyn to debias the device level meta optimization.

Learning Structure. The general structure of PFL is given in Algorithm 2. In the beginning of each communication round, P devices are selected uniformly at random as active device set $\mathcal{P}_t$. The current server meta model, $\boldsymbol{w}^t$, is sent to each of these active devices. Devices construct the regularizer depending on the objectives as in Eq. 3.2 or 3.3 where $\boldsymbol{g}_i^t$ corresponds to the gradient of device level meta loss.

We continue device level optimization with a subroutine consisting of SGD steps and the constructed regularizer as described in Update method in Algorithm 3. First, we start from the server model $\boldsymbol{w}_{i,1}^{t+1} = \boldsymbol{w}^t$. To realize unbiased gradient estimates between the personalized model and the meta loss, we randomly draw two minibatches of data $(D_i^k, D_i^{k'})$ from device dataset D_i. We obtain a customized device model as $\overline{\boldsymbol{w}}_{i,k}^{t+1} = \tilde{T}_i(\boldsymbol{w}_{i,k}^{t+1}, D_i^k)$ where the transformation function personalizes the current meta model $\boldsymbol{w}_{i,k}^{t+1}$ using the minibatch of D_i^k. Then, we perform one step SGD update on the regularized empirical loss with respect to the second minibatch as,

$$\boldsymbol{w}_{i,k+1}^{t+1} {=} \boldsymbol{w}_{i,k}^{t+1} {-} \beta \left(\nabla \tilde{f}_i(\overline{\boldsymbol{w}}_{i,k}^{t+1}, D_i^{k'}) {+} \nabla \mathcal{R}_i^t(\boldsymbol{w}_{i,k}^{t+1}) \right)$$

where $\tilde{f}_i(\overline{\boldsymbol{w}}_{i,k}^{t+1}, D_i^{k'})$ is the empirical loss with the customized model $\overline{\boldsymbol{w}}^{t+1}$ on minibatch

$D_i^{k'}$ and β is the learning rate. After performing K SGD updates, the subroutine ends and we set the device meta model as $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_{i,K+1}^{t+1}$.

By definition, the regularizer, $\mathcal{R}_i^t(\boldsymbol{w})$, depends on the current server model and the current gradient information, $\boldsymbol{g}_i^t$. We update the gradient information for the next round depending on the objective as in Eq. 3.2 or 3.3.

After solving the device level optimization, active devices transmit the trained model $\boldsymbol{w}_i^{t+1}$ back to the server for aggregation. On the other hand, inactive devices do not receive the current model and freeze their local gradients and local models to be their past values. We note that we communicate the local gradient information along with the trained model if we use the objective in Eq. 3.2.

PFLDyn Algorithm. PFLDyn algorithm is based on the local objective as in Eq. 3.3 which is an extension of FedDyn to personalized federated learning.

The server integrates the global gradient information to the server model as explained in Update$_1$. Since the server meta model already has the global gradient information, devices do not need extra tranmsmission of the global gradient and they construct the regularizer as shown in Regularizer method in Algorithm 3. In PFLDyn, devices communicate only device models.

PFLScaf Algorithm. Using local objective as in Eq. 3.2, we can get to PFLScaf algorithm which is an extension of SCAFFOLD to personalized federated learning.

The server obtains the device models and device gradients from the active devices. It then calculates the global gradient and the server meta model as shown in Update$_2$. The server meta model and the global gradient are transmitted to the devices. Devices correct the biased gradient of the local loss with global gradient information as described in Regularizer method in Algorithm 3. Different from PFLDyn, PFLScaf communicates the models as well as gradient information.

Customized Transformations. In passing we point out that our proposed method allows for arbitrary transformations at the devices. This is important from the perspective that it allows for adaptation to new tasks as well as for scaling complexity of the customized classifier to the amount of available sample data.

Intuitive Justification. The optimal meta model that solves OPT satisfies the first order condition as,

$$\sum_{i\in[m]} \nabla f_i\left(\overline{\boldsymbol{w}}_i^*\right) = \mathbf{0}, \quad \overline{\boldsymbol{w}}_i^* = T_i(\boldsymbol{w}^*).$$

It is important to highlight the fact that the gradient of the individual device level meta objectives are not necessarily $\mathbf{0}$ .i.e $(\nabla f_i(\overline{\boldsymbol{w}}_i^*) \neq \mathbf{0})$. Indeed, were this to be the case, it would imply that fully optimizing device meta models would lead to device specific biases in the meta-model. PFL proposes to sequentially modify device empirical risk functions to eliminate such data-specific bias. The fact that this is possible is justified in the following Proposition 3.2.1.

Proposition 3.2.1 *For sufficiently large K, if the device meta models in Algorithm PFLDyn converge, they converge to the optimal meta model as,*

$$\lim_{t\to\infty} \boldsymbol{w}_i^t = \boldsymbol{w}_i^\infty \implies \boldsymbol{w}_i^\infty = \boldsymbol{w}^* \quad \forall i \in [m],$$

where $\sum_{i\in[m]} \nabla f_i\left(\overline{\boldsymbol{w}}_i^*\right) = \mathbf{0}, \quad \overline{\boldsymbol{w}}_i^* = T_i(\boldsymbol{w}^*)$.

3.2.1 Analysis of PFL

In this section, we present convergence guarantees for PFLDyn Algorithm for convex and nonconvex cases. Convergence rate analysis of PFLScaf Algorithm is mainly similar so we omit it here. In this context, we bound the number of communication rounds required to achieve ϵ error in OPT for convex functions and first-order stationarity condition for nonconvex functions. For simplicity we assume that the number of SGD steps, K, is sufficiently large such that, in each round, whenever a device is active, the solution returned is a stationary point of the operative customized loss at that time. In particular, say device, $i \in \mathcal{P}_t \subset [m]$ is active at time $t \in [T]$, then the operative customized loss is described by $f_i^t(\boldsymbol{w}) = f_i(\overline{\boldsymbol{w}}) + \mathcal{R}_i^t(\boldsymbol{w})$ where $\overline{\boldsymbol{w}} = T_i(\boldsymbol{w})$, and in this case we assume that K is sufficiently large to render, $\nabla f_i^t(\boldsymbol{w}_i^{t+1})$=$\mathbf{0}$. While this assumption may appear impractical, we note that, uniformly across all of our experiments, for small and large-scale datasets, we found that the norm of $\nabla f_i^t(\boldsymbol{w}_i^{t+1})$ is negligible relative to the size of $\boldsymbol{w}$ for $K \approx 50$. Furthermore, note that our proofs can be extended, and the resulting bounds suffer an additional variance term, which approaches zero with K.

Centralized Competitor. The centralized meta model $\boldsymbol{w}^*$ minimizes OPT with access to all device datasets. Since Algorithm 2 does not share data among devices, we characterize its performance with the number of communication rounds required to achieve ϵ difference relative to performance of centralized learner minimizing OPT, namely,

$$E\left[F(\boldsymbol{w}_{\text{Alg}}^T)\right] - F(\boldsymbol{w}^*) \leq \epsilon, \tag{3.4}$$

where $\boldsymbol{w}_{\text{Alg}}^T$ is the meta model Algorithm 2 finds after T communication rounds, $\boldsymbol{w}^*$ is the optimal meta model and expectation is with respect to randomness due to active device set at each round $(\mathcal{P}_t)$. While the statement 3.4 is tractable for convex $\{f_i \circ T_i\}$ functions, for nonconvex functions, following convention, we state the convergence as the number of communication rounds required to achieve a stationary point of OPT with ϵ error as,

$$E\left\|\nabla F\left(\boldsymbol{w}_{\text{Alg}}^T\right)\right\|^2 \leq \epsilon. \tag{3.5}$$

Based on convex and nonconvex convergence objectives, we state our theorem as,

Theorem 2 *For a suitably chosen $\alpha \in \mathbb{R}$, PFLDyn Algorithm, for sufficiently large K, returns an expected error less than ϵ in T communication rounds as,*

- *Convex and L smooth $\{f_i \circ T_i\}_{i\in[m]}$ functions,*

$$T = O\left(\frac{1}{\epsilon}\sqrt{\frac{m}{P}}\left(LD + \frac{1}{L}G\right)\right),$$

- *Nonconvex and L smooth $\{f_i \circ T_i\}_{i\in[m]}$ functions,*

$$T = O\left(\frac{1}{\epsilon}\left(L\frac{m}{P}\Delta_1 + L^2\Delta_2\right)\right),$$

where $D = \|\boldsymbol{w}^1 - \boldsymbol{w}^*\|^2$, $\boldsymbol{w}^* = \underset{\boldsymbol{w}}{arg\min}\ F(\boldsymbol{w})$, $G = \frac{1}{m}\sum_{i\in[m]}\|\nabla f_i\left(\overline{\boldsymbol{w}}_i^*\right)\|^2$, $\overline{\boldsymbol{w}}_i^* = T_i(\boldsymbol{w}^*)$, $\Delta_1 = F(\boldsymbol{w}^1) - F(\boldsymbol{w}^*)$, $\Delta_2 = \frac{1}{m}\sum_{i\in[m]}\|\boldsymbol{w}_i^1 - \boldsymbol{w}^1\|^2$.

The expected error is defined as in Eq. 3.4 and in Eq. 3.5 for convex and nonvonex functions respectively. The expectation is over the randomness of device participation.

Theorem 2 shows that with sufficient number of iterations, Algorithm 2 reaches to an expected ϵ error for convex and nonconvex device level meta function as in relations Eq. 3.4 and 3.5 respectively. We see the number of communication rounds to achieve expected ϵ error scales with $\frac{1}{\epsilon}$ for both convex and nonconvex settings. We present results for strongly convex functions in the supplementary section.

3.3 Experiments

Our goal in this section is to tabulate the performance of PFL and its variants against the state-of-art algorithms on benchmark datasets. We sample device data to synthesize various degrees of statistical data heterogeneity and task diversity among devices. We then report performance and tabulate our results against several metrics including oracle performance (centralized data), average customization performance, best and worst device customization. For each of these, we report the number of model transmissions required by PFL to achieve target accuracy of the competitor.

Methods. We evaluate PFL variants, namely, PFLDyn (Proto) and PFLScaf (Proto) that use Proto adaptation; PFLDyn (MAML) and PFLScaf (MAML) that use MAML adaptation and personalized FedAvg (Proto) that uses proto adaptation, P-Avg (Proto). We compare our methods to PerFedAvg (Fallah et al., 2020b) as well as their agnostic (no personalization) counterparts: FedAvg and no PFL variants. Observe that prior works in this context (Chen et al., 2018; Jiang et al., 2019) are essentially those that appear in (Fallah et al., 2020b), and for this reason we present (Fallah et al., 2020b)'s method and the vanilla FedAvg.

We first start with a summary datasets and models used in this section and we refer to Appendix A.1.4 for details of the empirical setup. We then continue how we construct diverse device datasets that reflects heterogeneity among devices. Then, we explain the metrics and measures we use to compare the methods. Finally, we present our findings.

Table 3.1: The number of model transmissions relative to one round of (Fallah et al., 2020b) required to reach the target test accuracy for the average level personalization performance in the Active Class Induced Diversity (ACID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of (Fallah et al., 2020b). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and the gain with respect to (Fallah et al., 2020b) method is shown.

Dataset	Accuracy	(Fallah et al., 2020b)	PFLDyn (Proto)	PFLDyn (MAML)	PFLScaf (Proto)	PFLScaf (MAML)	P-Avg (Proto)	Gain
3 Classes per Device								
CIFAR-10	92.2	>1000	**211**	455	618	>1000	797	>**4.7**×
	91.6	815	**152**	242	514	>1000	334	**5.4**×
CIFAR-100	90.6	>1000	**186**	376	480	>1000	838	>**5.4**×
	89.1	961	**133**	255	328	>1000	383	**7.2**×
5 Classes per Device								
CIFAR-10	88.8	>1000	**221**	392	666	>1000	783	>**4.5**×
	87.6	794	**163**	237	436	924	286	**4.9**×
CIFAR-100	86.5	860	**185**	223	478	>1000	>1000	**4.6**×
	86.3	718	**179**	203	466	>1000	954	**4.0**×
7 Classes per Device								
CIFAR-10	86.9	>1000	**202**	235	622	>1000	843	>**5.0**×
	85.8	925	**155**	177	452	734	365	**6.0**×
CIFAR-100	83.0	>1000	**200**	224	528	>1000	976	>**5.0**×
	82.7	998	**186**	207	494	>1000	772	**5.4**×

Datasets & Models. We use popular datasets with standard train/test splits such as CIFAR-10, CIFAR-100 (Krizhevsky, 2009). As models, we use a CNN architecture that has two convolutional layers with two max pooling layers followed by two fully connected layers and a final softmax layer. We implement methods in PyTorch framework (Paszke et al., 2019) and use Higer library (Grefenstette et al., 2019) for MAML adaptation.

Diverse FL datasets. Since heterogeneous device data distribution is a key challenge in customized federated learning, we perform our experiments with highly heterogeneous device settings. Following (Fallah et al., 2020b), we model task and data heterogeneity level of a federated setting in terms of distributional distance between device dataset joint distributions $p_i(\boldsymbol{x}, y)$ such as total variation (TV) distance. We propose two different ways of inducing divergent task dataset constructions across devices.

Active Class Induced Diversity (ACID). In this setting, we first assign a fixed sized class list to each device in which the size of the class list is small. After selecting the classes, train/test data of each device is randomly constructed from the actual train/test data splits without replacement according to the class lists. For instance, in CIFAR-100 dataset, we investigate a setting with 100 devices where we fix the number of classes in each device to be 5. Since there are overall 100 classes and we limit number of classes for each device to 5, we have many devices where each of them have strictly different classes. This results in large TV distance across devices since there is minimal class overlap. We also experiment with increasing and decreasing the number of classes per device, and these experiments point to how PFL handles different levels of task diversity.

Anonymous Label Induced Diversity (ALID). Similar to ACID, we again choose a fixed number of classes for each device and construct device datasets. For each device, we then randomly permute class indices, so that a class index in one device has no relationship with another device. Clearly the TV distance even when each device has all of the classes is large in this situation. Our motivation for this study stems from practical privacy concerns in federated learning, where devices may not wish to reveal class information but would still want to benefit from federated training.

Performance metrics. PFL minimizes the average performance (Eq. OPT) of the personalized models among devices.

Pointwise Metric. While average performance is important, an end-user is interested on her own dataset. It is important to tabulate pointwise device performance. For this reason we also report the best and worst performing devices.

Relative Target Accuracy. We compare our methods against competing methods in terms of the required number of transmitted models relative to one communication round of (Fallah et al., 2020b). This is equal to the number of communication rounds for all methods except PFLScaf variants. PFLScaf variants communicate two models in each communication round, so we report 2× of the communication rounds for PFLScaf variants. In all cases since one of the PFL variants dominates our competitors, we report the ratio of the number of transmitted models required between our method and the baseline as a measure of gain. If a method can not reach to the target in the allowed rounds, we mark the number with $>$ sign.

Oracle Accuracy. In Section 3.2.1 we derived convergence guarantees with respect to an oracle that has centralized data access and optimizes Eq. OPT accordingly. In this measure, we report the number of communication rounds required to get close to the target accuracy of an oracle.

Partial Participation. Following federated learning settings, we tested the methods with 100 devices where 10% of them are active at each round. For CIFAR-10 and CIFAR-100, we considered both ACID and ALID with $3, 5$ and 7 classes per device schemes. We refer to Appendix A.1.4 for additional experimental details.

Analysis and Discussions. Table 3.1 shows the gain compared to (Fallah et al., 2020b) method for various settings of 3, 5 and 7 classes per devices for both datasets in the ACID setting. Similarly, Table 3.2 demonstrates the device performances for the ALID scenario. The convergence curves as well as the lowest and the highest level personalization results are given Appendix A.1.4.

Table 3.2: The number of model transmissions relative to one round of (Fallah et al., 2020b) required to reach the target test accuracy for the average level personalization performance in the Anonymous Label Induced Diversity (ALID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of (Fallah et al., 2020b). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and gain with respect to (Fallah et al., 2020b) method is shown.

Dataset	Accuracy	(Fallah et al., 2020b)	PFLDyn	PFLDyn	PFLScaf	PFLScaf	P-Avg	Gain
			(Proto)	(MAML)	(Proto)	(MAML)	(Proto)	
3 Classes per Device								
CIFAR-10	90.1	>1000	**109**	970	290	>1000	169	>**9.2**×
	87.9	792	**73**	323	184	520	95	**10.8**×
CIFAR-100	89.8	>1000	**156**	849	406	>1000	390	>**6.4**×
	83.7	985	**53**	229	118	656	83	**18.6**×
5 Classes per Device								
CIFAR-10	87.5	>1000	**161**	846	452	>1000	341	>**6.2**×
	79.6	514	**54**	88	126	296	64	**9.5**×
CIFAR-100	83.9	>1000	**203**	520	296	>1000	233	>**4.9**×
	78.7	983	100	177	146	680	**87**	**11.3**×
7 Classes per Device								
CIFAR-10	86.7	>1000	**224**	911	574	>1000	455	>**4.5**×
	76.3	966	**57**	86	120	292	59	**16.9**×
CIFAR-100	77.5	>1000	**158**	568	228	>1000	170	>**6.3**×
	68.3	960	**58**	151	82	432	58	**16.6**×

Personalization is needed in both ACID and ALID scenarios. As seen in Figure A·16 and A·17, no personalization baselines converge to substantially lower average test accuracy in the ACID scenario. Furthermore, these methods predict the classes randomly in the ALID scenario due to the label anonymity, which in turn indicates the need of personalization. Thus, the results of PFL and FedAvg without personalization are not tabulated in tables due to their non-comparable performance.

PFLDyn (Proto) leads to significant savings in both ACID and ALID scenarios. We observe that PFLDyn using Proto adaptation reaches the target accuracy faster than all the other methods on the average device performance metric in Table 3.1 and 3.2. For instance, in CIFAR-10, ACID 7 classes per device setting, PFLDyn (Proto) leads to more than 5× communication savings to achieve the same level of performance compared to (Fallah et al., 2020b) method. This effect is also seen in Figure A·14 where PFLDyn (Proto) converges faster and to a higher point than (Fallah et al., 2020b).

PFL based optimization outperforms FedAvg based optimization regardless of adaption function (MAML or Proto). As shown in Table 3.1 and 3.2, the PFL based methods converge to the target accuracy with fewer number of model transmissions in nearly all the experiments. As seen in Figure A·14 and A·15, PFL methods achieve a higher test accuracy compared to (Fallah et al., 2020b). We can infer that PFL is capable of debiasing meta-model updates at the server allowing for superior device personalization. Thus, PFL leads to a faster and robust convergence than FedAvg regardless of the adaption function.

PFL improves the performance of the lowest level personalization. In addition to the average test accuracy among all devices, PFL based methods converges faster than (Fallah et al., 2020b) even for the lowest level performing devices in both ACID and ALID scenarios as shown in Table A.5 and A.6. PFL improves the performance by finding a better meta model for personalization among all devices. In particularly, the savings of PFL on the average device performance doesn't sacrifice the performance of a subset of devices.

PFL achieves centralized performance. The centralized performance for CIFAR-10, 5 classes per device ACID and ALID using Proto adaptation is 89.8% and 90%

respectively. Based on Table 3.1 and 3.2 we see that PFLDyn (Proto) achieves near the centralized performance around 300 communication rounds. Similarly, the centralized performance for CIFAR-100, 5 classes per device ACID setting using Proto adaptation is 89.7% . PFLDyn (Proto) achieves near centralized performance within 400 communication rounds without actually sharing device data.

Proto adaptation is robust to label anonymity. The Proto-based adaptation demonstrates similar convergence performance in the ALID scenario with anonymous labels compared to the ACID scenario. In contrast, the performance of the MAML-based adaption (PFLDyn, PFLScaf and (Fallah et al., 2020b)) degrades significantly in the ALID scenario. According to Figure A·14 and A·15, the convergence curves of methods using the Proto adaption are similar in the ACID and ALID scenarios. However, the MAML-based methods converge to significantly lower average test accuracy. In addition, the same observation can be found in Table 3.1 and 3.2. For instance, in the CIFAR-10 5 classes setting, the Proto-based models (PFLDyn, PFLScaf and P-Avg) require 161, 452 and 341 communication rounds respectively to achieve the average test accuracy 87.5% in the ALID scenario, while similarly require 163, 436 and 286 rounds to achieve 87.6% in the ACID scenario. But (Fallah et al., 2020b) can no longer achieve such an accuracy level within 1000 rounds in the ALID scenario as in the ACID scenario. Thus, the Proto-based adaption is more robust than the MAML-based adaption in more strictly privacy-preserving scenario when the labels are anonymous among devices.

Algorithm 3 PFL Subroutines

```
function Regularizer($\boldsymbol{w}, \boldsymbol{w}^t, \boldsymbol{g}_i^t$) (& $\boldsymbol{g}^t$ in PFLScaf):
    PFLDyn: $\mathcal{R}_i^t(\boldsymbol{w}) = -\langle \boldsymbol{w}, \boldsymbol{g}_i^t \rangle + \frac{\alpha}{2}\|\boldsymbol{w}-\boldsymbol{w}^t\|^2$,
    PFLScaf: $\mathcal{R}_i^t(\boldsymbol{w}) = \langle \boldsymbol{w}, -\boldsymbol{g}_i^t + \boldsymbol{g}^t\rangle$,
    Return $\mathcal{R}_i^t(\boldsymbol{w})$
end function
function Update($\boldsymbol{w}^t, K, \beta, D_i, \mathcal{R}_i^t$):
    Set $\boldsymbol{w}_{i,1}^{t+1} = \boldsymbol{w}^t$,
    for$k = 1, 2, \dots K$ do
        Get two minibatches $D_i^k, D_i^{k'}$ randomly from $D_i$,
        Set customized model $\overline{\boldsymbol{w}}_{i,k}^{t+1} = \tilde{T}_i(\boldsymbol{w}_{i,k}^{t+1}, D_i^k)$,
        Update meta model
        $\boldsymbol{w}_{i,k+1}^{t+1}=\boldsymbol{w}_{i,k}^{t+1}-\beta\left(\nabla\tilde{f}_i(\overline{\boldsymbol{w}}_{i,k}^{t+1}, D_i^{k'})+\nabla\mathcal{R}_i^t(\boldsymbol{w}_{i,k}^{t+1})\right)$
    end for
    Set $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_{i,K+1}^{t+1}$,
    PFLDyn:$\boldsymbol{g}_i^{t+1}=\boldsymbol{g}_i^t-\alpha\left(\boldsymbol{w}_i^{t+1}-\boldsymbol{w}^t\right)\approx\nabla f_i\circ T_i(\boldsymbol{w}_i^{t+1})$,
    PFLScaf:$\boldsymbol{g}_i^{t+1}=\boldsymbol{g}_i^t-\boldsymbol{g}^t-\frac{1}{K\beta}\left(\boldsymbol{w}_i^{t+1}-\boldsymbol{w}^t\right)\approx\nabla f_i\circ T_i(\boldsymbol{w}^t)$,
    Return $\boldsymbol{w}_i^{t+1}, \boldsymbol{g}_i^{t+1}$
end function
function Update$_1$ $\left(\{\boldsymbol{w}_i^{t+1}\}_{i\in\mathcal{P}_t}\right)$:
    $\boldsymbol{g}^{t+1} = \boldsymbol{g}^t - \alpha\frac{1}{m}\left(\sum_{i\in\mathcal{P}_t}\boldsymbol{w}_i^{t+1} - \boldsymbol{w}^t\right)$,
    $\boldsymbol{w}^{t+1} = \left(\frac{1}{|\mathcal{P}_t|}\sum_{i\in\mathcal{P}_t}\boldsymbol{w}_i^{t+1}\right) - \frac{1}{\alpha}\boldsymbol{g}^{t+1}$
    Return $\boldsymbol{w}^{t+1}$
end function
function Update$_2$ $\left(\{\boldsymbol{w}_i^{t+1}, \boldsymbol{g}_i^{t+1}\}_{i\in\mathcal{P}_t}\right)$:
    $\boldsymbol{g}^{t+1} = \boldsymbol{g}^t + \frac{1}{m}\left(\sum_{i\in\mathcal{P}_t}\boldsymbol{g}_i^{t+1} - \boldsymbol{g}_i^t\right)$,
    $\boldsymbol{w}^{t+1} = \left(\frac{1}{|\mathcal{P}_t|}\sum_{i\in\mathcal{P}_t}\boldsymbol{w}_i^{t+1}\right)$
    Return $\boldsymbol{w}^{t+1}, \boldsymbol{g}^{t+1}$
end function
```

Chapter 4

Federated Learning in Heterogeneous Networks

This chapter targets a more practical FL problem where users support different model architectures based on the device's capacity. For instance, consider a FL setting with many cellphone users where we want to distributively train a model on all user data (McMahan et al., 2017b). Some users might have the latest release of a cellphone whereas the rest use old versions. The users with the latest releases would prefer a different, potentially more complex, model than the user with old cellphones. Conventional FL as in (McMahan et al., 2017a) fails in this case because devices have different model architectures.

We investigate the above practical FL problem where we allow devices to have different architectures based on their capacities. We propose FedHeN that modifies device training by introducing a novel side objective to the devices with complex models. The side objective allows FedHeN to jointly train complex and simple architectures.

We test our method in real-world datasets of CIFAR10 and CIFAR100 and compare it to a naive baseline and the current state-of-the-art method. We show that FedHeN achieves significant communication savings as well as better performance compared to the competitors.

Contributions

- We present FedHeN to jointly train a server model with different architectures based on device capacities.

- We empirically show that FedHeN achieves significant communication savings compared to the baselines.

This work is presented in (Acar and Saligrama, 2022).

4.1 Related Work

HeteroFL (Diao et al., 2021) introduces FL with heterogeneous networks problem. HeteroFL considers a setting where simple architecture is mapped to a subset of the complex architecture. The server constructs new models by averaging model weights of all devices based on the mapping. Different from HeteroFL, FedHeN introduces novel side objectives for complex device training.

Early Exit. Due to their size, big DNNs consume more energy and they are slow to operate. (Hubara et al., 2016; Yang et al., 2019) propose to quantize/binarize the weights of DNNs to improve memory and computation costs. (Bolukbasi et al., 2017) propose to train a big DNN so that the network adaptively early exits in 'easy' examples to decrease inference costs. (Kaya et al., 2019) propose to add a side objective on DNN to prevent 'overthinking' of DNNs for easy examples. Different from these works, we are interested in FL. FedHeN introduces side objectives to jointly train models in FL with different architectures.

We further cover some of the new works published after our study. (Kim et al., 2023) propose a depth-based simple-complex architecture differentiation and use a self-distillation loss to improve local training further. (Jiang et al., 2023) employ an adaptive model pruning strategy where devices train models based on their capacities. (Zhong et al., 2022) propose the Semi-HFL method, using semi-supervised techniques and an early exit strategy.

Algorithm 4 FL in Heterogeneous Networks - FedHeN

```
Input: $T$, $E$, $\eta$, N, $\{\mathcal{D}_i\}_{i\in[N]}$, initial models $\boldsymbol{w}_s^1, \boldsymbol{w}_c^1$,
for $t = 1 \ldots T$ do
    Randomly sample active devices, $\mathcal{Z} \subset [N]$,
    Divide $\mathcal{Z}$ into simple and complex devices, $\mathcal{Z}_s, \mathcal{Z}_c$
    # Client Optimization
    for $i \in \mathcal{Z}_s$ do
        Receive the server simple model, $\boldsymbol{w}_s^t$,
        $\boldsymbol{w}_{s,i}^{t+1} = \text{ClientTraining}\left(\boldsymbol{w}_s^t, \mathcal{D}_i, E, \eta\right)$
        Transmit $\boldsymbol{w}_{s,i}^{t+1}$ back to the server.
    end for
    for $j \in \mathcal{Z}_c$ do
        Receive the server complex model, $\boldsymbol{w}_c^t$,
        $\boldsymbol{w}_{c,j}^{t+1} = \text{ClientTrainingSideObj}\left(\boldsymbol{w}_c^t, \mathcal{D}_j, E, \eta\right)$
        Transmit $\boldsymbol{w}_{c,j}^{t+1}$ back to the server.
    end for
    # Server Optimization
    Set $\boldsymbol{w}_s^{t+1}$ using weights from all active devices,
    $\boldsymbol{w}_s^{t+1} = \frac{1}{|\mathcal{Z}|}\left(\sum_{i\in\mathcal{Z}_s} \boldsymbol{w}_{s,i}^{t+1} + \sum_{j\in\mathcal{Z}_c}\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}}\right)$
    Set $\boldsymbol{w}_c^{t+1}$'s sub-net as the updated simple model,
    $[\boldsymbol{w}_c^{t+1}]_{\mathcal{M}} = \boldsymbol{w}_s^{t+1}$
    Set rest of the $\boldsymbol{w}_c^{t+1}$ using complex active devices,
    $[\boldsymbol{w}_c^{t+1}]_{\mathcal{M}'} = \frac{1}{|\mathcal{Z}|_c}\sum_{j\in\mathcal{Z}_c}\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}'}$
end for
```

4.2 Definition and Method

FL setting consists of one server node and N device nodes. Each device i has a different dataset $\mathcal{D}_i$. Let $f_i : \mathcal{W} \to \mathcal{R}$ be the loss of using a model on device i's dataset. FL solves,

$$\min_{\boldsymbol{w}\in\mathcal{W}} \frac{1}{N}\sum_{i\in[N]} f_i\left(\boldsymbol{w}\right)$$

where $\boldsymbol{w}$ is the NN parameters.

Different from the conventional FL, we are interested in having different architectures in devices. For simplicity, consider a setting where we have a simple architecture,

Algorithm 5 FedHeN Device Optimizations

```
function ClientTraining ($\boldsymbol{w}_s, \mathcal{D}_i, E, \eta$):
    Start from $\boldsymbol{w}_i$=$\boldsymbol{w}_s$, train $E$ epochs,
    for $E$ epochs, batch $B \subset \mathcal{D}_i$ do
        Compute batch gradient, $\hat{\nabla} f_i(\boldsymbol{w}_i)$,
        Update, $\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i - \eta \hat{\nabla} f_i(\boldsymbol{w}_i)$,
    Return trained model $\boldsymbol{w}_i$
end function
function ClientTrainingSideObj ($\boldsymbol{w}_c, \mathcal{D}_j, E, \eta$):
    Start from $\boldsymbol{w}_j$=$\boldsymbol{w}_c$, train $E$ epochs with side obj.,
    for $E$ epochs, batch $B \subset \mathcal{D}_j$ do
        Compute batch gradient along with side obj.,
        $\hat{\nabla} f_j(\boldsymbol{w}_j)$, $\hat{\nabla} f_j\left([\boldsymbol{w}_j]_{\mathcal{M}}\right)$,
        Update, $\boldsymbol{w}_j \leftarrow \boldsymbol{w}_j - \eta \left(\hat{\nabla} f_j(\boldsymbol{w}_j) + \hat{\nabla} f_j\left([\boldsymbol{w}_j]_{\mathcal{M}}\right)\right)$,
    Return trained model $\boldsymbol{w}_j$
end function
```

$\boldsymbol{w}_s \in \mathcal{W}_s$, and a complex architecture $\boldsymbol{w}_c \in \mathcal{W}_c$. Let $\mathcal{S} \subset [N]$ and $\mathcal{C} = [N] - S$ be the devices that have simple and complex architectures respectively. We reformulate our problem as,

$$\min_{\substack{\boldsymbol{w}_s \in \mathcal{W}_s \\ \boldsymbol{w}_c \in \mathcal{W}_c}} \frac{1}{|\mathcal{S}|} \sum_{i \in S} f_i\left(\boldsymbol{w}_s\right) + \frac{1}{|\mathcal{C}|} \sum_{j \in C} f_j\left(\boldsymbol{w}_c\right)$$

$$\text{such that } \mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = 0 \tag{4.1}$$

where $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c)$ captures the relationship between simple and complex architectures.

We note that Eq. 4.1 is an ERM objective. If there is no condition (no $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = 0$ constraint), one could minimize the objective by separating complex and simple losses. However, we are not interested in training data, we would like to train models that perform well on test data. Hence, we introduce $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c)$ as a way of regularizing the models.

To relate simple and complex architectures, we assume,

Assumption 1 *Simple architecture is a sub-network of the complex architecture. There exists a set of indices, $\mathcal{M}$, of complex architecture such that* $\mathcal{W}_s = \{[\boldsymbol{w}_c]_{\mathcal{M}} | \boldsymbol{w}_c \in$

Table 4.1: IID split, 50 simple and 50 complex devices, 10% participation rate. The number of communication rounds required to achieve the target test performance for different methods. The gain in using FedHeN compared to best baseline method is given.

Dataset	Accuracy	FedHeN	Decouple	NoSide	Gain
Simple Model					
CIFAR-10	84.4	289	943	805	**2.8**×
	83.4	249	731	669	**2.7**×
CIFAR-100	46.4	296	864	984	**2.9**×
	45.4	250	588	807	**2.4**×
Complex Model					
CIFAR-10	88.5	649	991	941	**1.4**×
	87.5	456	739	669	**1.5**×
CIFAR-100	46.8	468	963	614	**1.3**×
	45.8	376	752	472	**1.3**×

$\mathcal{W}_c\}$ *where* $[\boldsymbol{w}_c]_{\mathcal{M}}$ *selects the weights of* $\boldsymbol{w}_c$ *based on the index set* $\mathcal{M}$*.*

We encourage weight sharing between simple architecture and the corresponding sub-network of the complex architecture as in Assumption 1. We let $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = \|\boldsymbol{w}_s - [\boldsymbol{w}_c]_{\mathcal{M}}\|^2$.

To further regularize models, we add a side objective to the complex device training. Complex devices minimize their losses along with the corresponding sub-network of simple architecture as,

$$\min_{\substack{\boldsymbol{w}_s \in \mathcal{W}_s \\ \boldsymbol{w}_c \in \mathcal{W}_c}} \frac{1}{|\mathcal{S}|} \sum_{i \in S} f_i\left(\boldsymbol{w}_s\right) + \frac{1}{|\mathcal{C}|} \left(\sum_{j \in C} f_j\left(\boldsymbol{w}_c\right) + f_j\left([\boldsymbol{w}_c]_{\mathcal{M}}\right) \right)$$

$$\text{such that } \mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = 0 \tag{4.2}$$

We would like highlight some properties of Eq. 4.2,

- Simple architecture is trained on all datapoints instead of on the datapoints only from the simple devices. Hence, the generalization of simple architecture is improved.

- $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c)$ correlates simple and complex architecture. A better simple model leads to a better complex model through $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = 0$ condition.

FedHeN Algorithm. FedHeN solves Eq. 4.2 with the steps summarized in Algorithm 4.

In each round, a random subset of devices become active, $\mathcal{Z}$. We divide set $\mathcal{Z}$ into simple active and complex active device sets as $\mathcal{Z}_s$ and $\mathcal{Z}_c$ respectively.

Simple active devices receive the server simple model, $\boldsymbol{w}_s^t$. We compute a local model $\boldsymbol{w}_{s,i}^{t+1}$ by starting from $\boldsymbol{w}_s^t$ and training it for E epochs on local dataset $\mathcal{D}_i$ displayed as 'ClientTraining' method (Alg. 5). The trained model is transmitted back to the server.

Complex active devices receive the server complex model, $\boldsymbol{w}_c^t$. We train a local model starting from $\boldsymbol{w}_c^t$ and training it for E epochs using their local dataset, $\mathcal{D}_j$ with gradients of the complex and the simple model shown as 'ClientTrainingSideObj' method (Alg. 5). Namely, we update the model with summation of batch gradient of complex model, $\hat{\nabla} f_j(\boldsymbol{w}_j)$, and batch gradient of the corresponding sub-network (simple) model, $\hat{\nabla} f_j\left([\boldsymbol{w}_j]_{\mathcal{M}}\right)$. The trained model is transmitted back to the server.

The server collects models from participating devices. The server simple model is constructed by averaging weights from all active devices, .i.e the simple devices $\{\boldsymbol{w}_{s,i}^{t+1}\}_{i\in\mathcal{Z}_s}$ as well as the common sub-net of the complex devices $\left\{\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}}\right\}_{j\in\mathcal{Z}_c}$, ln. 18 in Alg. 4. The common sub-network of the server complex model is set equal to the constructed server simple model, ln. 20 in Alg. 4. Finally, the rest of the server complex model is constructed by averaging the corresponding weights of the active complex models, .i.e $\left\{\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}'}\right\}_{j\in\mathcal{Z}_c}$ where $\mathcal{M}'$ corresponds to the sub-network that is not common with the simple architecture, ln. 22 in Alg. 4.

This completes one round of training of FedHeN. We iterate the same process for

Table 4.2: Non-IID split, 50 simple and 50 complex devices, 10% participation rate. The number of communication rounds required to achieve the target test performance for different methods. The gain in using FedHeN compared to best baseline method is given.

Dataset	Accuracy	FedHeN	Decouple	NoSide	Gain
Simple Model					
CIFAR-10	79.4	295	986	810	**2.7**×
	78.4	256	816	676	**2.6**×
CIFAR-100	43.8	278	978	914	**3.3**×
	42.8	239	813	762	**3.2**×
Complex Model					
CIFAR-10	84.2	596	1000	857	**1.4**×
	83.2	519	887	751	**1.4**×
CIFAR-100	44.8	450	997	498	**1.1**×
	43.8	372	754	456	**1.2**×

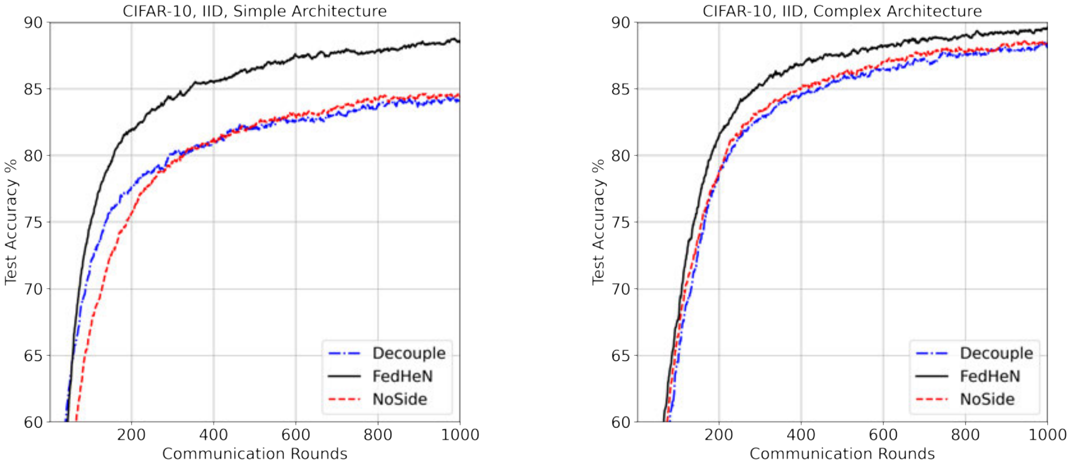
T communication rounds.

Cost of side objective. In passing, we note that side objective adds minimal cost to the complex devices. Firstly, it is a light weight operation. Complex devices calculate gradients of the full model, $\boldsymbol{w}_c$. Calculating the gradient with respect to the simple model, $\boldsymbol{w}_s$, requires less computation. Secondly, the main energy consumption occurs during transmission of models (Halgamuge et al., 2009).

4.3 Experiments

In this section, we compare FedHeN method to baselines in real-world dataset settings. We refer to Appendix A.1.3 for a description of the hyperparameters.

FL dataset. We test our method using CIFAR-10 and CIFAR-10 (Krizhevsky, 2009). We split the datasets into 100 clients and randomly activate 10 clients in each round. We consider both IID and non-IID splits in our experiments. IID split is constructed by randomly partitioning data into clients. Non-IID split is constructed using a Dirichlet prior on the labels as in (Yurochkin et al., 2019).



(a)

(b)

Figure 4·1: Test accuracy vs. communication rounds on CIFAR-10 IID split. **a**: Simple, **b**: Complex.

We assume the first 50 devices have simple architecture and the last 50 devices have complex architecture in all experiments.

Models. We use PreActResNet18[1] (He et al., 2016b) for the complex architecture. PreActResNet18 has 4 residual blocks and total of 11.1M parameters.

If we centralize all client datapoints, the complex architecture gets 93% and 73.5% performance for CIFAR-10 and CIFAR-100 datasets respectively. If we centralize half of the datapoints as in 50 devices, the complex architecture gets 90.5% and 62.6% performance for CIFAR-10 and CIFAR-100 datasets respectively.

As a simple architecture, we consider the first 2 residual blocks of PreActResNet18. Then, we add a mix pooling layer (Lee et al., 2016) that learns a weighted combination of avg pooling and max pooling layers as in (Kaya et al., 2019). The simple architecture has overall 0.7M parameters.

If we centralize all client datapoints, the simple architecture gets 86% and 63.2%

[1] BatchNorm layers store data statistics which results in privacy leakage. We use GroupNorm layers (Wu and He, 2018) instead in all ResNet models.

performance for CIFAR-10 and CIFAR-100 datasets respectively. If we centralize half of the datapoints as in 50 devices, simple model gets 84.5%, 55.7% performance for CIFAR-10 and CIFAR-100 respectively.

Methods. We compare FedHeN to two baselines as,

- *Naive Decouple.* Decouple minimizes Eq. 4.1 without any $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c) = 0$ constraint. It decouples complex and simple device training. It separately trains a complex and a simple model using FedAvg. Decouple is summarized in Algorithm 8.

- *NoSide*[2] *(Diao et al., 2021).* NoSide is motivated from HeteroFL (Diao et al., 2021). It minimizes Eq. 4.1 with the same $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c)$ as FedHeN. The key difference is that it does not use side objective in the complex architecture training. NoSide is summarized in Algorithm 9.

Evaluation Metric. We fix a target test accuracy for server simple and server complex models. We compare the number of communications rounds to achieve the target test accuracy for all methods.

Results. Table 4.1 & 4.2 show the number of communication rounds to get target accuracies in all methods. We highlight the gain of using FedHeN compared to the best competitor in the last column. We present convergence curves vs. communication rounds in Figure 4·1, A·18 & A·19 (Appendix A.1.3).

FedHeN leads to significant communication savings. We observe that FedHeN trains better models uniformly in all the experiments shown in the gain columns of Table 4.1 & 4.2. For instance, the same simple performance of 43.8% is obtained using 3.3× less communication with FedHeN in CIFAR-100 non-IID setting. The

[2] HeteroFL is proposed in a setting where simple model is obtained by shrinking CNN channels of the complex model different from our setting. Except from the simple model definition, HeteroFL uses the same $\mathcal{R}(\boldsymbol{w}_s, \boldsymbol{w}_c)$ as FedHeN and it does not add side objective. We name HeteroFL in our setting as NoSide.

communication savings ranges from 1.1× to 3.3× in our experiments.

Decouple vs NoSide. Decouple is a naive algorithm. However, it performs close to NoSide algorithm in CIFAR-10 IID and non-IID settings as shown in Figure 4·1 & A·18. For instance, the same test accuracy for complex model is achieved in 991 and 941 rounds for Decouple and NoSide models in CIFAR-10 IID setting as shown in Table 4.1.

Simple model in FedHeN achieves similar to centralized accuracy. FedHeN achieves better simple performance because simple architecture is trained on all datapoints due to the side objective in complex devices. Moreover, weight sharing between complex and simple architecture improves simple model's generalization. For instance, FedHeN's simple model in CIFAR-10 achieves 88.6% test accuracy in IID split within 1000 communication rounds as shown in Figure 4·1 which is higher than the centralized accuracy of simple model. Differently, NoSide and Decouple gets worse simple performance compared to FedHeN and the centralized model as shown in Figure 4·1, A·18 & A·19.

Complex model in FedHeN achieves better performance compared to competitors. Training with side objective improves complex model performance in FedHeN. For instance, in CIFAR-10 IID setting, FedHeN's complex model achieves 89.6% test performance within 1000 communication rounds which is >1% better compared to the competitors. This is also reflected in the gain values. FedHeN leads to 1.5× communication savings for complex model.

Chapter 5

Memory Efficient Online Meta-Learning

In this chapter, we develop a novel method for online meta-learning overcoming drawbacks of (Finn et al., 2019). Like (Finn et al., 2019) we focus on training deep neural network models, and consider the setting where task instances are revealed to the learner episodically. We also attempt to train the model's initial parameters so that, on any new task, the model parameters can be rapidly adapted with a small amount of data by means of gradient descent. Subsequently, the learner then updates its underlying task-agnostic parameters based on solving the new task.

Linear Memory Scaling. (Finn et al., 2019)'s follow-the-meta-learner (FTML) leverages the well-known follow the regularized leader (FTRL) method in online learning. While learning task-agnostic model parameters (meta-learning step), FTML recalls all task instances that have heretofore appeared. This is undesirable and impractical, and as such leads to linear increase in memory with the number of observed tasks. One option is to update the meta model only using the current task, but this leads to significant current task bias. Another possibility is to leverage an online gradient descent (OGD) algorithm, based on linearization of loss-functions for prior tasks, but the linearization would be around stale model parameters associated with the previous tasks. While practical, empirically these options do not lead to meaningful meta-learning performance.

Memory Efficient Online Meta-Learning (MOML). To overcome memory scaling, we introduce a fixed-size state-vector, which is dynamically updated after completion of an episodic task. The state-vector, which serves the purpose of encoding past task experiences, parameterizes the regularizer penalty for next episode. As a result, model parameters retain prior task experience, and utilize this experience to solve new tasks. Our MOML scheme is not only memory efficient, but is more effective than FTML. We compare MOML with FTML on current tasks as well as past tasks (to evaluate catastrophic forgetting), and show that MOML dominates FTML on several benchmark datasets. We also analyze MOML theoretically and show that for T tasks, we achieve sub-linear $O(\sqrt{T})$ regret on convex losses, and $O(\sqrt{T})$ local regret for non-convex loss functions.

Contributions

- We present, MOML, a new family of online learning algorithms that do not explicitly store loss functions from previous rounds.
- We show that MOML has $O\left(\sqrt{T}\right)$ regret guarantees,
- We empirically show that MOML achieves significantly improved memory footprint with no perceptible degradation in performance over existing baselines.

This work is published in (Acar et al., 2021c).

5.1 Related Work

Online learning. In vanilla online learning (Shalev-Shwartz, 2007), (possible adversarial) loss functions are sequentially revealed and the learner is trained as well as tested at each round. The agent aims to minimize cummulative regret that measures how well the algorithm performs compared to the best possible fixed model in hindsight. Online learning is a well established field and we refer to extensive studies for more information (Hazan, 2019; Shalev-Shwartz, 2012). Online gradient descent (OGD) (Zinkevich, 2003) proposes to take a gradient descent step in each round using the current loss. Follow the Regularized Learner (FTRL) (Abernethy et al., 2008) minimizes a regularized version of all seen loss functions. Different from meta learning, the learner is expected to minimize the loss functions, and does not leverage insights from past experiences. In contrast, our focus is on online meta learning problem where the agent is expected to meta learn the revealed task instances.

Continual learning, (a.k.a lifelong learning) (Thrun and Pratt, 2012) is related to online learning, with particular focus on catastrophic forgetting. In this context, the agent is expected to do well on the seen tasks as well as efficiently learn new task instances (Chen and Liu, 2018). For instance, Learn-to-Grow framework (Li et al., 2019b) avoids forgetting previous tasks by expanding the network architecture of the learner with upcoming tasks. Variational continual learning (Nguyen et al., 2018) is a method using variational inference on a set which has representative datapoints from the seen tasks. In contrast, our goal is to reduce the memory footprint, by allowing for the learner to delete *all* data instances for past tasks. Additionally, our goal is to derive regret guarantees as in online learning.

Online meta learning (Finn et al., 2019) proposes to fuse meta learning with online learning. In online meta learning, an agent is expected to meta learn tasks where the tasks are sequentially revealed. Prior works have proposed to leverage OGD and FTRL to the online meta learning problem (Zhuang et al., 2020; Finn et al., 2017). In the FTML (Finn et al., 2019) approach the goal is to learn initializations of model parameters (meta-model) so as to allow for quick adaptation to all of the previously viewed task instances based on taking a few gradient steps from the meta-model. For this reason, FTML must store data from all seen tasks to update its meta model. This means that the memory complexity of FTML linearly grows as new tasks arrive which is impractical. We propose a memory efficient approach, which does not require storing past past instances.

Subsequent works such as OSML (Yao et al., 2020) and FTML-VS (Yu et al., 2020) propose extensions to FTML. OSML is a pipeline that has multiple so called meta blocks to facilitate the learning of new tasks. FTML-VS aims to decrease the number of datapoints used during meta training as tasks arrive. Nevertheless, these methods still require storing datapoints for all seen tasks. Different from these works, our goal is to bypass storing datapoints corresponding to seen tasks.

We further cover some of the new works published after our study. (Ho et al., 2023) propose a dynamic-prototype memory replay method, PMR, to decrease memory complexity by storing a few examples of seen tasks. (VS et al., 2022) consider an object detection online meta-learning problem and focus on empirical performance using unsupervised domain adaptation methods. (Zhang et al., 2021b) propose a Bayesian approach where meta-parameters are constructed using Gaussian mixture models.

5.2 Definition and Method

The learner's goal is to train a meta model, chosen from a parameterized model with parameters $\boldsymbol{w} \in \mathbb{R}^d$ for the setting where new task instances are sequentially revealed at each round. Each task has a specific joint distribution denoted as $\mathcal{P}_t$ in which task features $\boldsymbol{x} \in \mathcal{X}$ and the corresponding labels $y \in \mathcal{Y}$ are drawn from it $(\boldsymbol{x}, y) \sim \mathcal{P}_t$. The agent has access to a limited supervised dataset $D_t = \{(\boldsymbol{x}_t^i, y_t^i)\}_{i=1}^{N_t}$ for each task in order to obtain a task specific model. We define task loss as the expected loss with respect to $\mathcal{P}_t$ as $f^t(\boldsymbol{w}) = E_{\{\boldsymbol{x},y\}\sim P_t} L\left((\boldsymbol{x}, y); \boldsymbol{w}\right)$ where L is the loss function that the model incurs on the data tuple $(\boldsymbol{x}, y)$. Our objective is to get a sublinear rate of the following regret statement,

$$R_T = \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^t) - \min_{\boldsymbol{w}\in\mathbb{R}^d} \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}) \tag{5.1}$$

where U^t is the meta adaptation function that transforms the meta model into a task specific model using the limited supervised dataset D_t and the agent is compared against the best fixed meta learner that has access to all loss functions $\{f^t\}_{t=1}^T$ in hindsight.

Adaptation function. The regret statement depends on the transformation function U^t for task t. There are many transformations proposed in meta learning field to obtain a task specific model out of the meta model. In this work, we focus on MAML (Finn et al., 2017) adaptation. MAML transformation is proposed as,

$$U^t(\boldsymbol{w}) = \boldsymbol{w} - \eta \frac{1}{|D_t|} \sum_{(\boldsymbol{x},y)\in D_t} \nabla L\left((\boldsymbol{x}, y); \boldsymbol{w}\right)$$

and corresponds to updating the meta model with a step gradient descent using a meta learning rate η.

MOML Intuition. Before describing the algorithm, we build intuition for our solution by considering two conventional online learning algorithms: FTRL and OGD. Let us assume that we have $\{\ell^t\}_{t=1}^T$ losses in an online setting.

FTRL Algorithm. FTRL updates its model using all seen losses and a regularizer, namely,

$$\boldsymbol{w}^{t+1} = \operatorname*{argmin}_{\boldsymbol{w}} \; \frac{\mu}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{t} \ell^i(\boldsymbol{w}), \tag{5.2}$$

where μ is the coefficient on the quadratic regularizer. FTRL must store the history of all the past observed loss functions. Since the optimal competitor minimizes the sum of these losses (i.e. $\sum_{t=1}^T \ell^t(\boldsymbol{w})$), we can view FTRL, with proper regularization, converging to the optimal competitors risk at the expense of storing all seen task information.

OGD Algorithm. Different from FTRL, OGD does not store the seen losses. It applies one gradient descent step using the currently revealed loss as,

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \beta \nabla \ell^t \left(\boldsymbol{w}^t \right) \tag{5.3}$$

where β is the learning rate. Even though it gets to desired regret guarantees, it is hard to compare OGD to the best competitor that minimizes the sum of losses (i.e. $\sum_{t=1}^{T} \ell^t(\boldsymbol{w})$).

The Bias Problem. OGD updates the model with gradients arising from the loss revealed at that time. Since, the best competitor seeks to optimize the sum of losses, one could learn the best competitor, by running SGD on the average of all losses as

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \beta \frac{1}{T} \sum_{t=1}^{T} \nabla \ell^t \left(\boldsymbol{w}^k \right). \tag{5.4}$$

This iteration has the property that it converges to the optimal solution in hindsight for convex losses and a suitable choice of β.

The difference between OGD update in 5.3 and the competitor update in 5.4 is that the gradient directions are not aligned. More explicitly, the minima of the current loss is not the same as the competitor, $\min \ell^t(\boldsymbol{w}) \neq \min \sum_{s=1}^{T} \ell^s(\boldsymbol{w})$. As such the gradients have different directions. We propose a solution based on debiasing the gradient of the current loss, in the hope of taking a step towards the global gradient, while bypassing the need for recalling past seen instances.

A Toy Example. We illustrate the bias issue for an online learning setting with $T = 6$ losses where the parameter space is two dimensional. Figure 5·1 shows the contour plots (for quadratic loss functions) and the corresponding local optimal solutions for the 5 seen losses; the current revealed loss $\ell^6(\boldsymbol{w})$; and the sum of all losses $\sum_{t=1}^{6} \ell^t(\boldsymbol{w})$. The learner's parameters at this time is depicted as ×.

Using the current loss, we can only go towards minima of the cuurent loss where

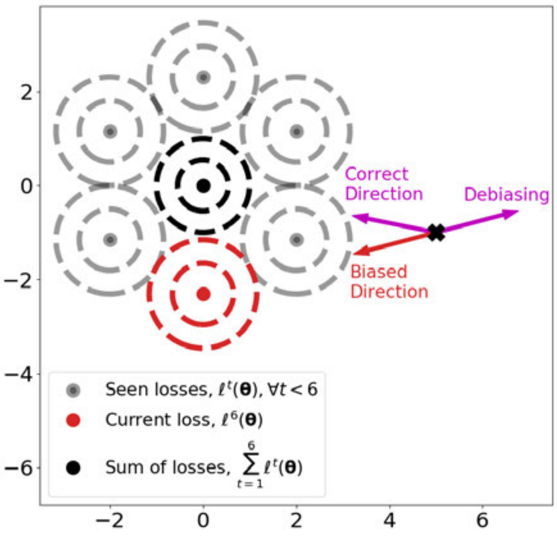


Figure 5·1: A two dimensional online learning setting with $T = 6$ losses. The current loss pulls the model towards its minima. However, the sum of losses have a different optimum point. Hence, the model is biased towards the current loss minima. We propose to debias the gradient so that the model is correctly updated.

the direction is shown with a red arrow, 'biased direction'. However, we would like to move towards the global minima denoted as the 'correct direction' since it points to the best competitor. The biased direction and the correct directions are not aligned which we refer as the bias problem.

The Debiasing Concept. Unfortunately, at a round $t < T$, all of the losses have not yet been revealed, and as such we can debias based only on the past losses. We propose to debias the current loss noted as 'debiasing' direction in Figure 5·1 by leveraging the past seen empirical loss functions. Then, we substitute a surrogate direction with the goal of correcting the biased direction. Our objective is to bypass the need to recall previous seen instances in computing this correction.

Let us denote the current model as $\boldsymbol{w}^t$. We start with the current model, $\boldsymbol{w}_1^{t+1} =$

$\boldsymbol{w}^t$ and apply K corrected gradient descent steps as,

$$\boldsymbol{w}_{k+1}^{t+1} = \boldsymbol{w}_{k}^{t+1} - \beta \left(\nabla \ell^t \left(\boldsymbol{w}_{k}^{t+1} \right) - \boldsymbol{d}^t + \boldsymbol{c}^t \right) \tag{5.5}$$

where $k = 1, 2, \ldots K$, $\boldsymbol{d}^t$ debiases the current loss and $\boldsymbol{c}^t$ encourages the correct direction. We note that our proposed solution does not require to store the losses as such $\boldsymbol{d}^t$ and $\boldsymbol{c}^t$ terms are updated using only the current loss.

MOML Algorithm. Our proposed method is presented in Algorithm 6. In each round, the current loss (f^t) along with adaptation function (U^t) is modified using a quadratic regularization and it is revealed to the algorithm. A task specific model is obtained using the transformation and the current meta model $\boldsymbol{w}^t$ as $\overline{\boldsymbol{w}}^t = U^t(\boldsymbol{w}^t)$. Then, the performance of the task specific model is recorded as $f^t(\overline{\boldsymbol{w}}^t) = f^t \circ U^t(\boldsymbol{w}^t)$.

We first update the meta model by optimizing using a quadratic penalty:

$$\mathcal{R}^t(\boldsymbol{w}) = -\langle \nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t), \boldsymbol{w} \rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2, \tag{5.6}$$

where $\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t)$ and $\boldsymbol{w}^t$ are the states the model stores. The linear term debiases the current loss and the second term corrects the direction as in $\boldsymbol{d}^t$ and $\boldsymbol{c}^t$ terms defined in update Eq. 5.5.

Subsequently, the algorithm iteratively optimizes the loss function with gradient corrections as,

$$\boldsymbol{w}_{k+1}^{t+1} = \boldsymbol{w}_{k}^{t+1} - \beta \left(\nabla f^t \circ U^t(\boldsymbol{w}_{k}^{t+1}) + \nabla \mathcal{R}^t(\boldsymbol{w}_{k}^{t+1}) \right). \tag{5.7}$$

After K gradient descent updates, the new meta model is obtained $\boldsymbol{w}^{t+1} = \boldsymbol{w}_{K+1}^{t+1}$.

MOML explicitly stores states $(\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t), \boldsymbol{w})$ by way of summarization of previous task instances, and as such incorporates this information in the constructed regularizer.

Algorithm 6 Memory Efficient Online Meta Learning- MOML

```
Input: $T, \nabla f^0 \circ U^0(\boldsymbol{w}^1) = \boldsymbol{w}^1 = \boldsymbol{w}^1 = \mathbf{0}, \alpha, K, \beta$
for $t = 1, 2, \dots T$ do
    Output $\boldsymbol{w}^t$, reveal $f^t$ and $U^t$, suffer $f^t \circ U^t(\boldsymbol{w}^t)$,
    $\mathcal{R}^t(\boldsymbol{w}) = -\langle \nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t), \boldsymbol{w} \rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2$,
    $\boldsymbol{w}_1^{t+1} = \boldsymbol{w}^t$,
    for $k = 1, 2, \dots K$ do
        $\boldsymbol{w}_{k+1}^{t+1} = \boldsymbol{w}_k^{t+1} - \beta \left( \nabla f^t \circ U^t(\boldsymbol{w}_k^{t+1}) + \nabla \mathcal{R}^t(\boldsymbol{w}_k^{t+1}) \right)$
    end for
    $\boldsymbol{w}^{t+1} = \boldsymbol{w}_{K+1}^{t+1}$,
    $\boldsymbol{w}^{t+1} = \frac{1}{2} \left( \boldsymbol{w}^t + \boldsymbol{w}^{t+1} - \frac{1}{\alpha} \nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) \right)$,
end for
```

$\boldsymbol{w}$ state is recursively updated as,

$$\boldsymbol{w}^{t+1} = \frac{1}{2} \left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1} - \frac{1}{\alpha} \nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) \right). \tag{5.8}$$

This completes one round of update mechanism for MOML.

Buffered-MOML (B-MOML). MOML can be extended to the situation where a fixed size buffer of previous task instances are also used. In this embodiment, we are allowed to store the latest B losses. For this scenario, the update rule for $\boldsymbol{w}$ is:

$$\boldsymbol{w}_{k+1}^{t+1} = \boldsymbol{w}_k^{t+1} - \beta \left(\nabla L_B^t(\boldsymbol{w}_k^{t+1}) + \nabla \mathcal{R}_B^t(\boldsymbol{w}_k^{t+1}) \right).$$

where $L_B^t(\boldsymbol{w}) = \frac{1}{B} \sum_{i=0}^{B-1} f^{t-i} \circ U^{t-i}(\boldsymbol{w})$ is the sum of last B losses and

$$\mathcal{R}_B^t(\boldsymbol{w}) = -\left\langle \frac{1}{B} \sum_{i=0}^{B-1} \nabla f^{t-i-1} \circ U^{t-i-1}(\boldsymbol{w}^{t-i}), \boldsymbol{w} \right\rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2$$

is the adapted regularizer. We utilize B-MOML in deriving regret bounds for non-convex (adversarial) setting.

Random Task Buffer. In experiments we also consider storing random tasks instances in our buffer. To do so, we consider a buffer as first-in-first-out (FIFO) queue. For each new task, we sample a biased coin with parameter p. We accept the new task in the buffer and put it at end of our queue, and then delete the first task.

Memory footprint of MOML does not grow over time. Our proposed method does not require storing all seen task instances. Instead, it accumulates previous task information in the auxiliary model ($\boldsymbol{w}$). Different from MOML, FTML needs to increase its memory usage in each round since it explicitly stores previous task information. We note that this leads to significant savings in terms of memory requirement.

MOML can leverage any meta adaptation function (U^t). We can use any adaptation function such as MAML, or other objectives such as prototypical adaptation, etc. MOML leads to a new family algorithms that can be applied to any online learning setup.

5.2.1 Analysis of MOML

MOML minimizes the following risk in K gradient steps,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} f^t \circ U^t(\boldsymbol{w}) + \mathcal{R}^t(\boldsymbol{w}) \tag{5.9}$$

To simplify the convergence analysis, we assume that MOML reaches a stationary point of Eq. 5.9, namely,

$$\nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) - \nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t) + \alpha \left(\boldsymbol{w}^{t+1} - \boldsymbol{w}^t\right) = \mathbf{0}. \tag{5.10}$$

This assumption is not unrealistic. Indeed, due to our quadratic penalty, in experiments, we have found that MOML is essentially reaches a stationary point (i.e., Eq. 5.10) within a small number of gradient steps. In particular, we find that the residual noise is significantly smaller than the model parameters, and can be ignored for the purpose of analysis. Nevertheless, we point out that it is possible to extend our results to include this additional residual noise.

MOML proposes to reach a similar solution without explicitly storing losses from previous rounds. We present an intuitive justification for our viewpoint based on the assumption that MOML is a stable[1] algorithm and the sequential updates, $\boldsymbol{w}^t$, converge, namely, $\lim_{t\to\infty} \boldsymbol{w}^t = \boldsymbol{w}'$. Nevertheless, with this assumption in place, it follows that MOML, if it converges, converges to a stationary point of the desired loss function. We state this as a proposition.

Proposition 5.2.1 *Suppose $f^t \circ U^t(\cdot)$ are a sequence of smooth functions with uniformly bounded Lipshitz constant. Furthermore, suppose $\boldsymbol{w}^t$ is a bounded convergent sequence approaching $\boldsymbol{w}'$. Then, $\boldsymbol{w}'$ is also a stationary point of the competitor defined as in Eq. 5.1.*

We sketch the proof below. Using Cesaro[2] mean argument, if models converge, the mean model does as well: $\frac{1}{t}\sum_{s\in[t]} \boldsymbol{w}^s \underset{t\to\infty}{\longrightarrow} \boldsymbol{w}'$. If we average Eq. 5.10 over time, we observe that $\nabla f^t \circ U^t(\boldsymbol{w}^{t+1})$ terms telescope and we get $\frac{1}{t}\sum_{s\in[t]} \boldsymbol{w}^{s+1} - \frac{1}{t}\sum_{s\in[t]} \boldsymbol{w}^s = -\frac{1}{\alpha t}\nabla f^t \circ U^t(\boldsymbol{w}^{t+1})$. If the gradients are bounded we can assume $-\frac{1}{\alpha t}\nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) \underset{t\to\infty}{\longrightarrow} \mathbf{0}$. Consequently we see that mean model and mean $\boldsymbol{w}$ state converge to the same model as $\frac{1}{t}\sum_{s\in[t]} \boldsymbol{w}^s \underset{t\to\infty}{\longrightarrow} \boldsymbol{w}'$. If we average update rule of $\boldsymbol{w}$ in Eq. 5.8 over time and plug these relations we get $\frac{1}{t}\sum_{s\in[t]} \nabla f^{s-1} \circ U^{s-1}(\boldsymbol{w}^s) \underset{t\to\infty}{\longrightarrow} \mathbf{0}$. Since we assume $\boldsymbol{w}^t \underset{t\to\infty}{\longrightarrow} \boldsymbol{w}'$, for sufficiently large t, $\frac{1}{t}\sum_{s\in[t]} \nabla f^{s-1} \circ U^{s-1}(\boldsymbol{w}') \underset{t\to\infty}{\longrightarrow} \mathbf{0}$. This is the stationary point relation of the accumulated losses. Therefore, MOML can be close to the optimal competitor without actually storing the losses from previous rounds.

As in standard online learning, we assume the losses to have a bounded gradient $\|\nabla f^t \circ U^t(\boldsymbol{w})\| \le G$. We first give a regret statement for convex losses.

Theorem 3 *For convex possibly adversarial $\{f^t \circ U^t\}_{t=1}^T$ functions and $\alpha = O\left(\sqrt{T}\right)$, Algorithm 6 satisfies,*

$$\sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^t) - \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^*) = O\left(\sqrt{T}\left(\|\boldsymbol{w}^*\|^2 + G^2\right)\right),$$

[1] Validating this assumption requires additional proof, such as showing the map $\boldsymbol{w}^t \to \boldsymbol{w}^{t+1}$ is contractive. In online meta learning, $\boldsymbol{w}^t$ convergence is not required since the losses are non-stochastic. The convergence happens in stochastic loss settings.

[2] If a sequence $\{a_i\}_{i=1}$ converges $a_i \underset{i\to\infty}{\longrightarrow} a'$, mean also converges $\frac{1}{i}\sum_{s=1}^{i} a_s \underset{i\to\infty}{\longrightarrow} a'$.

where $\boldsymbol{w}^{*}{=}arg\min\limits_{\boldsymbol{w}\in\mathbb{R}^d}\ \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w})$.

Theorem 3 gives sub-linear, $O\left(\sqrt{T}\right)$, regret rate for convex functions with bounded gradients. The dependency on T is optimal since there exists a setting where any algorithm suffers $\Omega\left(\sqrt{T}\right)$ regret for convex adversarial losses (Hazan et al., 2007).

Nonconvex Adversarial. It is well-known (see (Hazan et al., 2017)) that for adversarial nonconvex losses, sub-linear regret for the formulation in Eq. 5.1 is generally difficult to achieve. One option is to instead evaluate $\sum_{t=1}^{T}\left\|\nabla \ell^t(\boldsymbol{w}^t)\right\|^2$, but this is not meaningful, since we want to reach a stationary point for the sum of the losses, and not each individual loss. As a tractable regret notion, (Hazan et al., 2017) propose to use

$$\sum_{t=1}^{T}\left\|\frac{1}{B}\sum_{i=0}^{B-1}\nabla \ell^{t-i}(\boldsymbol{w}^t)\right\|^2, \tag{5.11}$$

where we consider a B sized window of the losses. This introduced time window smoothens the gradient of loss suffered and leads to sub-linear regret. Similar to (Hazan et al., 2017) algorithm, we use B-MOML to tackle this type of regret notion. B-MOML gets to $O\left(\frac{T}{B^2}G^2\right)$ regret. (Hazan et al., 2017) show a lower bound where a set of losses of losses is constsructed in a way that any algorithm suffers $\Omega\left(\frac{T}{B^2}\right)$. Based on this result, our regret rate is optimal in terms of T and B dependencies. We give formal statement and the proof in Appendix A.2.2.

Stochastic Setting with Nonconvex Losses. As another extension, we propose a different regret notion for nonconvex losses where we do not need to consider a time window either in the regret statement or in the algorithm. In this notion, at each time, we assume the losses are chosen uniformly at random without replacement from a predetermined set of losses, $i_t \sim [K]$, $\{\ell^j\}_{j=1}^{K}$ and we aim to minimize

$$\sum_{t=1}^{T} E\left\|\frac{1}{K}\sum_{k=1}^{K}\nabla \ell^{k}(\boldsymbol{w}^t)\right\|^2,$$

where the expectation is with respect to the random index i_t of the losses.

Theorem 4 *Suppose $\mathcal{T}$ is a collection of tasks, and for each $\tau \in \mathcal{T}$, $f^\tau \circ U^\tau$ is nonconvex L smooth. We choose tasks i_t from some task distribution, $P_\mathcal{T}$ in an IID fashion. Then, for $\alpha = O\left(\sqrt{T}\right)$, Algorithm 6 satisfies,*

$$\sum_{t=1}^{T} E \left\| E_\tau[\nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)] \right\|^2 = O\left(\sqrt{T}\left(\Delta + G^2 L\right)\right)$$

where $\Delta = E_\tau[f^\tau \circ U^\tau(\boldsymbol{w}^1)] - \min_{\boldsymbol{w}} E_\tau[f^\tau \circ U^\tau(\boldsymbol{w})]$.

Theorem 4 gives sub-linear regret rate for a nonconvex losses. For the stochastic setting, our regret rate is $O\left(\sqrt{T}\right)$.

Stochastic Setting with Nonconvex Polyak- Lojasiewicz (PL) Losses. The regret statement in Theorem 4 is motivated from the first order condition of stationary points. A better statement with respect to the optimum competitor can be obtained for PL nonconvex losses in the stochastic setting.

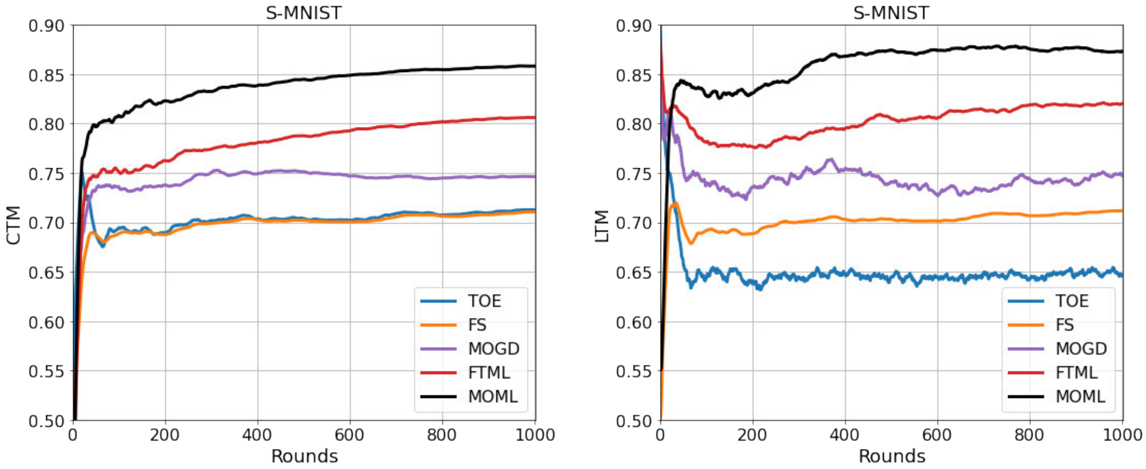
Corollary 1 *If each loss in Theorem 4 satisfies the (PL) condition, with parameter μ, it follows that,*

$$\sum_{t=1}^{T} E\left[E_\tau[f^\tau \circ U^\tau(\boldsymbol{w}^t)] - \min_{\boldsymbol{w}} E_\tau[f^\tau \circ U^\tau(\boldsymbol{w})]\right] = O\left(\sqrt{T}\frac{1}{\mu}\left(\Delta + G^2 L\right)\right).$$

Corollary 1 gives sub-linear rate with respect to the best competitor. We note that PL condition allows us to get a similar regret statement as in the convex setting even for nonconvex losses.

5.3 Experiments

This section displays our empirical comparison of MOML against competing baselines on standard datasets. We highlight main advantages of our method under various settings. We use PyTorch framework (Paszke et al., 2019) to train and evaluate our models. MAML meta training is implemented with Higher (Grefenstette et al., 2019)



(a)

(b)

Figure 5·2: Experiment results on S-MNIST. **(a)**: CTM versus rounds plot. MOML adapts well to the unseen tasks compared to the baselines. **(b)**: Smoothed LTM versus rounds. MOML is robust to catastrophic forgetting.

library. Hyperparameter tuning in an online setting poses challenges, since unlike the batch setting, we typically do not have a validation set. To overcome this issue we leverage the Hedge algorithm (Freund and Schapire, 1997) for hyperparameter tuning. We start by explaining the datasets, models and the baselines used for evaluations. We refer to Appendix A.1.4 for details of our setup.

Datasets. We evaluate the performance of our approach on three benchmark datasets: *MNIST* (LeCun et al., 1998), *CIFAR-100* (Krizhevsky, 2009) and *miniImageNet* (Vinyals et al., 2016). We state the task generation process for each of the datasets below.

Sequential MNIST (S-MNIST): Similar to Rainbow MNIST (Finn et al., 2019), we construct S-MNIST dataset consisting of diverse MNIST classification tasks. We generate a large-scale dataset consisting of 1000 tasks where train/test set of each task include transformations such as rotations, axes flip, cropping and scaling. These transformations are applied to class instances of each task. We note that unlike

Rainbow MNIST (Finn et al., 2019) where a specific transformation is used in each task, we allow a more diverse tasks by including different transformations among classes within each task.

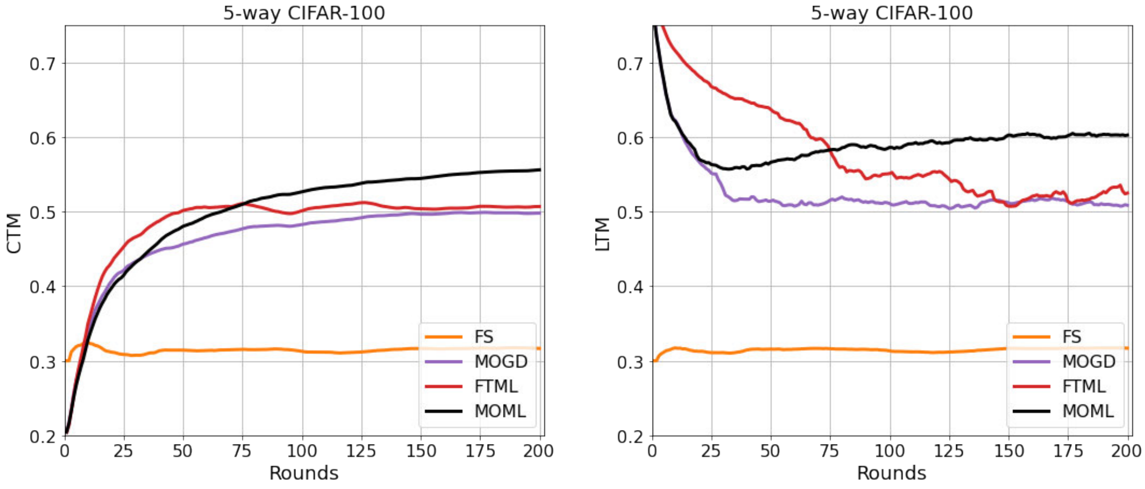
5 way-CIFAR-100: Similar to (Finn et al., 2019), we construct a sequence of 5-way classification tasks within CIFAR-100 dataset. There are overall 200 tasks and train/test set of each task contains a 5-class combination from the 100 classes. Since only 5 classes are chosen out of 100 classes for each task, this generation ensures that the tasks are diverse, and particularly challenging for the online meta learning setting.

5 way-miniImagenet: miniImageNet dataset is collected as a subset ILSVRC-2012 (Deng et al., 2009) where there are a total of 100 classes with 600 images in each class. miniImageNet has realistic RGB images and it is harder compared to CIFAR100 dataset. Similar to 5 way-CIFAR-100, in 5 way-miniImageNet, we generate 100 tasks where each task has 5 classes from miniImageNet dataset.

Realistic Test-Bed. We note that S-MNIST and 5-way CIFAR-100 tasks are harder than the corresponding ones in (Finn et al., 2019). Firstly, our setting is large scale where there are 1000 and 200 tasks for S-MNIST and 5 way-CIFAR-100 respectively compared to 60 and 50 tasks in prior work. Secondly, we have fewer number of training data per task in our setting, and in particular, 60 and 250 training datapoints for S-MNIST and 5 way-CIFAR-100 datasets respectively. In contrast, (Finn et al., 2019) has 900 and 2000 datapoints. We note that in online meta learning, we must leverage common knowledge across different tasks, since we do not have sufficient data for any one task. Our task generation resembles a more realistic and large-scale experimental test-bed. As a result, FTML performance is significantly lower than that reported in prior work.

Models. We use fully connected network architecture for S-MNIST experiments.

The model takes flattened version of the image and passes through one hidden layer of size 200 neurons with ReLU non linearity followed by the softmax layer. For 5 way-CIFAR-100, we use a CNN architecture consisting of two convolutional layers with 64 5×5 filters, two max pooling layers, two fully connected layers with hidden sizes as 384 and 192 and a final output layer. In 5 way-miniImageNet, we use a similar CNN architecture where we have three convolutional blocks and max pooling layers followed by two fully connected layers of size 400 and 100 and a final softmax layer.



(a) (b)

Figure 5·3: Experiment results on 5 way-CIFAR-100. **(a)**: CTM versus rounds plot. MOML adapts well to the unseen tasks compared to the baselines. **(b)**: Smoothed LTM versus rounds plot. MOML is robust to catastrophic forgetting.

Methods. We compare MOML algorithm against two types of baselines. First set of baselines meta learns tasks such as FTML (Finn et al., 2019) and Meta OGD (MOGD). As described, FTML is an extension of FTRL in online meta learning setting where the algorithm stores all seen task instances. Similarly, we define Meta OGD (MOGD) where the algorithm extends OGD (Zinkevich, 2003) using the meta losses at each iteration. Different from FTRL, MOGD does not store losses. Our second set of baselines follows (Finn et al., 2019) and includes train on everything

(TOE) and train from scratch (FS). In TOE baseline, we do not meta learn a model, instead we store all tasks and learn one predictive model. During inference first fine tune TOE baseline using MAML and then record its performance. In FS baseline, we adapt a random model to each task using the limited data.

Performance Metrics. We report performance on three different metrics. These are Current Task Metric, Long-Term Task Metric and Task Learning Efficiency Metric.

Current Task Metric (CTM). We evaluate our models with respect to the current revealed task instance. We first allow the meta model to adapt to the current task using MAML adaptation and record its performance on task test data. We note that meta model is adapted directly before being trained on the task similar to the regret statement (Eq. 5.1).

Long-Term Task Metric (LTM). Different from CTM, we also look at the performance with respect to the previous tasks. At each round, we adapt the current meta model to each of the previous tasks using the limited data and record the performances with task test data. Then, we consider the average performance among the seen tasks. This metric measures *catastrophic forgetting*, and our goal is to ensure that past experiences are not forgotten. We note that the meta model is first adapted to the old tasks using associated training data in evaluating catastrophic forgetting and LTM.

Task Learning Efficiency Metric. Similar to (Finn et al., 2019), we record the number of datapoints required to achieve a sufficient performance on the current task instance.

Analysis and Discussions. We report average CTM as well as LTM accuracy for all methods in Table 5.1. We present CTM and LTM versus rounds plots in Figure 5·2, 5·3 and 5·4 for S-MNIST, 5 way-CIFAR-100 and 5 way-miniImageNet settings respectively. We further test the effect of number of classes in each task on CIFAR-100 dataset and report the performances of MOML and FTML for the setting where each task has $3, 4$ or 5 classes in Table 5.2.

MOML adapts well to unseen tasks, improving upon competing methods. MOML gets to similar or higher accuracy in CTM compared to the baselines in Table 5.1 and 5.2. For instance, in 5 way-CIFAR-100 setting, MOML reaches 5% higher accuracy than FTML. As another example, Figure 5·2 shows that MOML strictly dominates all baselines after 100 rounds. These findings show that MOML can easily adapt new task instances.

MOML is task efficient. MOML achieves 80% test accuracy on new task instances by using less amount of limited data compared to FTML in S-MNIST as shown in Appendix A.1.4. The findings show that MOML is more task efficient and it can easily adapt to new task instances.

MOML is robust to catastrophic forgetting. MOML implicitly encodes past experience without explicitly storing it. This is evident in Table 5.1 and 5.2. For instance, in S-MNIST setting, MOML improves 5% LTM over FTML baseline. This shows that unlike competitors, MOML has superior performance on previously observed tasks.

MOML decreases memory complexity. Among the meta learning methods, MOML and MOGD do not store seen task instances. Different from these methods, FTML remembers all seen task information and minimizes the accumulated sum of losses. Not storing task instances for MOGD leads to performance degradation in comparison to FTML (Table 5.1). On the other hand, MOML outperforms both methods without storing previous data.

Meta learning is necessary for good generalization. TOE baseline learns one predictive model for all tasks. Since tasks are diverse, its performance is strictly

[3] For miniImageNet, MOML performances reported in Table 5.1 is B-MOML with buffer of 10 tasks.

Table 5.1: Performances for S-MNIST, 5way-CIFAR-100 and 5way-miniImageNet. TOE is not tabulated for 5way-CIFAR-100 and 5way-miniImageNet due to its poor performance.

	CTM	**LTM**
S-MNIST		
TOE	71.22	63.73
FS	71.15	71.14
MOGD	74.63	74.07
FTML	80.62	82.49
MOML	85.82	87.49
5 way-CIFAR-100		
FS	31.58	31.60
MOGD	49.92	50.21
FTML	50.68	54.50
MOML	55.83	60.78
5 way-miniImageNet		
FS	20.90	20.81
MOGD	49.08	56.86
FTML	56.77	63.12
MOML[3]	56.23	64.27

lower than the methods that meta learns tasks even though we allow fine tuning during inference time. Similarly, FS does not use meta learning and directly adapts the model for each task. it performs poorly on both CTM and LTM (see Table 5.1).

Ablation study on B-MOML. B-MOML, a variant of MOML algorithm, described in Section 5.2, allows for storing B tasks in a buffer. We test B-MOML to see the effect of buffer size. Table 5.3 displays B-MOML performance with buffer sizes as $0, 5, 10$. We note that $B = 0$ corresponds to original MOML algorithm where we do not store previous tasks.

MOML and B-MOML performances are comparable. We observe that storing seen task instances improves performance on new task and reduces the catastrophic for-

Table 5.2: MOML and FTML performances for $3, 4$ and 5 way-CIFAR-100 settings.

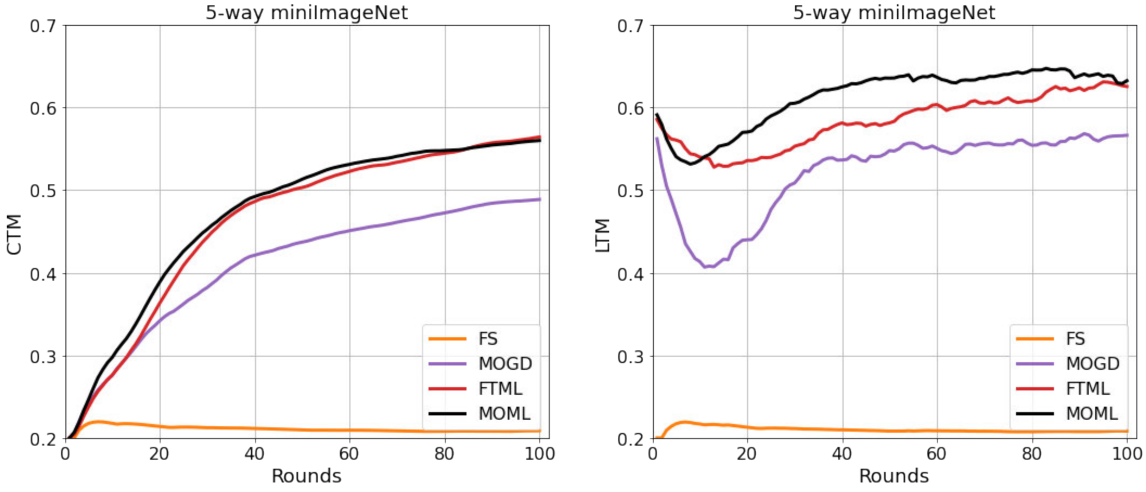
	K-way		
	3	**4**	**5**
		CTM	
FTML	67.07	58.19	50.68
MOML	69.10	60.85	55.83
		LTM	
FTML	61.84	54.70	54.50
MOML	72.15	62.79	60.78

Table 5.3: Ablative study of B-MOML with $0, 5$ and 10 size buffers where $B = 0$ is original MOML.

	Buffer Size, B		
	0	**5**	**10**
	S-MNIST		
CTM	85.82	84.21	84.33
LTM	87.49	87.54	87.77
	5 way-CIFAR-100		
CTM	55.83	56.37	56.70
LTM	60.78	63.76	65.24

getting in Table 5.3. In particular, LTM performances are improved with B-MOML. However, the change in CTM performance is marginal. For instance, increasing buffer from 0 to 10 increases CTM by only 1% in CIFAR-100 . We can infer that MOML effectively summarizes past task information.

Latest B-buffer is more effective than random B-buffer. As described in Section 5.2, we consider B-MOML variants where we allow buffers with the latest-B and a random variant of a fixed size. Table 5.4 shows that latest-B Buffer is somewhat more effective than the random case for MOML.



(a)

(b)

Figure 5·4: Experiment results on 5 way-miniImageNet. **(a)**: CTM versus rounds plot. MOML adapts well to the unseen tasks compared to the baselines. **(b)**: Smoothed LTM versus rounds plot. MOML is robust to catastrophic forgetting.

Table 5.4: Ablative study of buffer storing schemes of B-MOML.

	Buffer Size, B	
	5	**10**
S-MNIST		
CTM		
Random	83.30	84.07
Last	84.21	84.33
LTM		
Random	86.09	86.97
Last	87.54	87.77
5 way-CIFAR-100		
CTM		
Random	56.40	55.07
Last	56.37	56.70
LTM		
Random	64.37	64.37
Last	63.76	65.24

Chapter 6

Scaffolding a Student to Instill Knowledge

In this chapter, we focus on carefully guiding a powerful model's predictions to a smaller model to improve the generalization of the small model. One of the primary issues in KD (Hinton et al., 2015), where soft predictions of the student and a powerful teacher model is matched, is that the loss function is somewhat blind to the student's capacity to interpolate. In particular, when the student's capacity is significantly lower than the teacher's, we expect the student to follow the teacher only on those inputs realizable by the student.

We are led to the following question: *What can the teacher provide by way of predictive hints for each input, so that the student can leverage this information to learn to its full capacity?*

Our Proposal: Scaffolding a Student to Distill Knowledge (DiSK). To address this question, we propose that the teacher, during training, not only set a predictive target, $\mathbf{t}(\mathbf{x})$, but also provide hints on hard to learn inputs. Specifically, the teacher utilizes its model to output a guide function, $g(\mathbf{x})$, such that the student can selectively focus only on those examples that it can learn.

- *if* $g(\mathbf{x}) \approx 1$*,* teacher discounts loss incurred by the student on the input $\mathbf{x}$.

- *if* $g(\mathbf{x}) \approx 0$*,* teacher signals the input $\mathbf{x}$ as learnable by student.

With this in mind we modify the KL distance in the KD objective and consider, $D_{KL}(\mathbf{t}(\mathbf{x}), \phi(\mathbf{s}(\mathbf{x}), g(\mathbf{x})))$, where $\phi(\mathbf{s}, g)$, which will be defined later, is such that,

$\phi(\mathbf{s}, 0) = \mathbf{s}$ if g offers no scaffolding. We must impose constraints on the guide function g to ensure that only hard-to-learn examples are scaffolded. In the absence of such constraints, the guide can declare all examples to be hard, and the student would no longer learn. We propose to do so by means of a budget constraint $B(\mathbf{s}, g) \leq \delta$ to ensure that the guide can only help on a small fraction of examples. While more details are described in Sec. 6.3, we note that, in summary, our proposed problem is to take the empirical linear combination of the aforementioned KL distance and a cross-entropy term as the objective, and minimize it under the empirical budget constraint.

We emphasize that $g(\mathbf{x})$ is used only during training . The inference logic for the student remains the same as there is no need for $g(\mathbf{x})$ during inference. The guide function supported student training has three principal benefits. The benefits are explored in Sec. 6.1.

- **Censoring Mechanism.** Our guide function censors examples that are hard to learn for the student. In particular, when there is a large capacity gap, it is obvious that the student cannot fully follow the teacher. For this reason, the teacher must not only *set an expectation* for the student to predict, but also selectively gather examples that the student has the *ability to predict.*

- **Smoothen the Loss landscape.** We also notice in our synthetic experiments that whenever scaffolding is powerful, and can correct student's mistakes, the loss landscape undergoes a dramatic transformation. In particular, we notice fewer local minima in the loss viewed by the guided student.

- **Good Generalization.** The solution to our constrained optimization problem in cases where the guide function is powerful can ensure good student generalization. Specifically, we can bound the statistical error in terms of student complexity and not suffer additional complexity due to the teacher.

Contributions. We summarize our main results.

- We develop a novel approach to KD that exploits teacher representations to adjust the predictive target of the student by scaffolding hard-to-learn points. This novel scaffolding principle has wider applicability across other KD variants, and is of independent interest.

- We design a novel response-matching KD method (Gou et al., 2021) which is particularly relevant in the challenging regime of large student-teacher capacity mismatch. We propose an efficient constrained optimization approach that produces powerful training scaffolds to learn guide functions.

- Using synthetic experiments, we explicitly illustrate the structural benefits of scaffolding. In particular, we show that under our approach, guides learn to censor difficult input points, thus smoothening the student's loss-landscape and often eliminating suboptimal local minima in it.

- Through extensive empirical evaluation, we demonstrate that the proposed DiSK method;

  - yields large and consistent accuracy gains over vanilla KD under large student-teacher capacity mismatch (upto 5% and 2% on CIFAR-100 and Tiny-Imagenet).

  - produces student models that can get near-teacher accuracy with significantly smaller model complexity (e.g. 8× computation reduction with $\sim 2\%$ accuracy loss on CIFAR-100).

  - improves upon KD even under small student-teacher capacity mismatch, and is even competitive with modern feature matching approaches.

This work is published in (Kag et al., 2023).

6.1 Illustrative Examples

We present two synthetic examples to illustrate the structural phenomena of the censoring mechanism and smoothening of student's loss landscape enabled by the scaffolding approach DiSK, which lead to globally optimal test errors. We defer exact specification of the algorithm to Sec.6.3.

Example 1 (1D Intervals). Consider a toy dataset with one dimensional features $x \in [0, 9]$ and binary class labels $y \in \{\text{Red}, \text{Blue}\}$ as shown in Figure 6·1. There are two Blue labelled clusters as in $[2, 3]$ and in $[5, 7]$. The remaining points are labelled as Red. We sample 1000 i.i.d. data points as the training set and 100 data points as the test set with balanced data from both classes. We describe the details of the experiment setup such as models and learning procedure in Appx. A.1.5.

Teacher T belongs to the 2-interval function class, and the capacity-constrained student S belongs to the 1-interval function class. Since teacher capacity is sufficient to separate the two classes without error, it learns the correct classifier (see Figure 6·1). In contrast, the best possible student hypothesis cannot correctly separate the two classes. Hence, the student will have to settle onto one of the many local minima. We show these minima and the contour plot for the student in Figure 6·1. We present the results of training student models with different initializations in Table 6.1.

KD suffers from bad local minima. KD loss landscape contains many local minima (see Figure 6·1). Due to a big gap between student and teacher capacity, it is unable to help the student discern between these minima. Hence, KD fails to distinguish between the different minima (see Table 6.1).

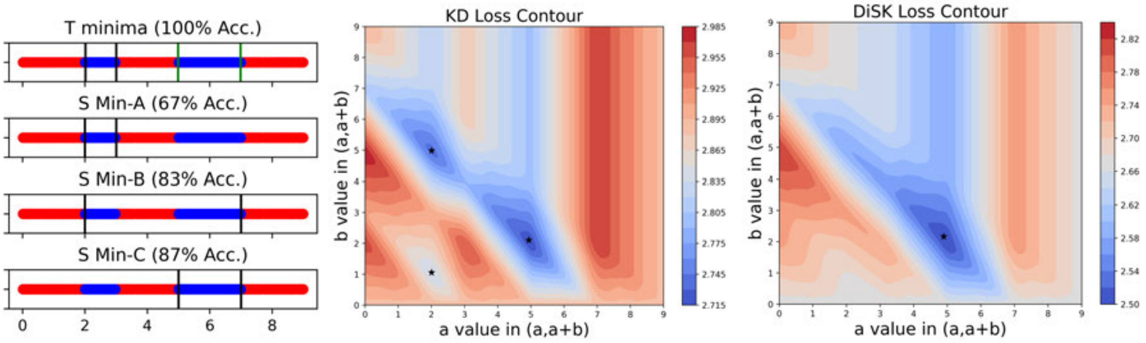
DiSK censors interval $[2, 3]$ *and in addition focuses training on learnable datapoints.* If we analyze the guide function at the end of the training, we see that it covers (censors) the first Blue cluster. Indeed, both clusters are not simultaneously

Table 6.1: The number of times each method lands on various local minima in two toy problems for 100 runs.

Dataset	**1D Intervals**			**2D Gaussians**			
Minima	**A**	**B**	**C (Global)**	**A**	**B**	**C**	**D (Global)**
Accuracy	**67%**	**83%**	**87%**	**70%**	**80%**	**90%**	**100%**
Cross-Entropy	35	64	1	73	12	9	6
KD	30	67	3	1	11	31	57
DiSK	9	1	90	0	0	3	97

*l*earnable with the available student capacity. Once we censor the interval $[2, 3]$, then the problem becomes realizable for the student model. The guide function thus captures the excess capacity of the data.

DiSK smoothens loss landscape. The guide function and the budget constraint enable our method to have a smooth loss landscape thanks to the guide-function censoring points, which eliminate the local minima. Hence, DiSK solution lands in the global minimum with high probability.



(a) (b) (c)

Figure 6·1: **(a)**: 1D Intervals. Data distribution on x-axis $[0, 9]$. Teacher T learns the correct decision boundary with 2-intervals and it is the global minima for this binary classification task. Student S has many bad local minima, and one global minima that best describes the decision boundary with 1-interval. **(b)**: KD training. Loss contour plot shows the various local minima exist. **(c)**: DiSK training. Loss contour plot shows the bad local minima no longer exist.

Example 2 (2D Gaussians). Consider another toy dataset with two dimensional features $\mathbf{x} \in \mathbb{R}^2$ and three class labels $y \in \{\text{Red}, \text{Green}, \text{Blue}\}$. Here we wish to show that DiSK can allow for *globally optimal solutions reaching 100% accuracy, which appears unachievable with cross-entropy minimization regardless of data size.*

Figure 6·2.a shows the labelled data. There are six cluster centers, two with each class label. Data points are drawn using Gaussian balls around the cluster centers with small radii. We sample 1000 i.i.d. data points as the training set and 1000 data points as the test set with equal representation from all three classes. We provide details (hypothesis classes, learning procedure, etc.) in Appx. A.1.5.

The teacher is a 3-layer neural network with 8, 16, and 3 neurons. The student is a 2-layer neural network with 2 and 3 neurons. We point out that the teacher being an over-parameterized network in this feature space, easily learns the correct decision boundary. While the student being severely constrained network suffers in learning the task. Different training runs lead to different local minima. We show teacher solution and student local minima in Figure 6·2.a. The contour plots for the student models under KD loss and DiSK loss are shown in Figure 6·2.b-6·2.c using (Li et al., 2018).

The results are similar to the 1D example - KD converges to a poor local minimum with at least 43% of the initializations, while in contrast, DiSK escapes these by focusing on the learnable part of the input space (Fig. 6·2.c), converging to the global minimum nearly always (Table 6.1).

To conceptualize our findings in these examples, let us attempt to intuitively infer the example-censoring, landscape-smoothening, and good generalization, by utilizing the following conditions that appear to be satisfied for these synthetic examples.

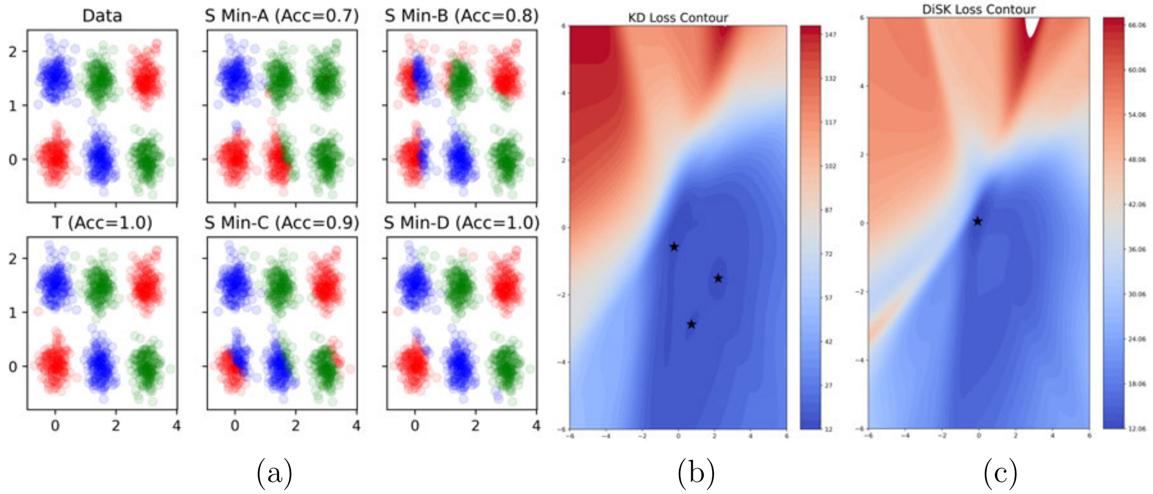
Realizability. Suppose we are in a situation where the guide function $g \in \mathcal{G}$ is sufficiently powerful that there is a student and guide function capable of interpolation, i.e., predictions supported by the guide function, $\phi(\mathbf{s}, g)$, interpolates to mimic the labels.

Example: For instance, consider a binary classification problem with the labels $y \in \{-1, 1\}$. Let $\phi(s, g) = y(s + g)$ with $s(\boldsymbol{w}) \in [-1, 1]$. Our realizability condition is that we always satisfy $y(s(\boldsymbol{w}) + g(\boldsymbol{w})) > 0$. As such, this leads to the condition that if $ys(\boldsymbol{w}) \leq 0$, then $yg(\boldsymbol{w}) > 0$. Therefore, $\mathbb{E}[\mathbb{1}_{[ys(\boldsymbol{w})<0]}] \leq \mathbb{E}[\mathbb{1}_{[ys(\boldsymbol{w})<0, yg(\boldsymbol{w})>0]}] \leq \mathbb{E}[\mathbb{1}_{[yg(\boldsymbol{w})>0]}]$.

Small Guide Function Capacity. In addition to realizability suppose the class of guide functions $g \in G$ has a small capacity (for instance, small VC dimension). For our case this condition is satisfied because our guidance function is obtained by using an MLP on teacher's last layer features.

Example: Continuing with the example above, say we now have m training instances, $(\boldsymbol{w}_i, y_i)$, $i \in [m]$, $\hat{g}(\boldsymbol{w})$ is guide function output of DiSK. We can infer by standard statistical learning results (Shalev-Shwartz and Ben-David, 2014) that, for the estimated function $\hat{g} \in G$, it follows with probability greater than $1 - \eta$ that $\mathbb{E}[\mathbb{1}_{[y\hat{g}(\boldsymbol{w})>0]}] \leq \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[y_i\hat{g}(\boldsymbol{w}_i)>0]} + \sqrt{\frac{VC(G)+\log\frac{1}{\eta}}{m}}$. As a result, we can say that if there is a student, $s(\boldsymbol{w})$ (not necessarily that output by DiSK), which complements $\hat{g}(\boldsymbol{w})$ and satisfies realizability, then with probability greater than $1 - \eta$: $\mathbb{E}[\mathbb{1}_{[ys(\boldsymbol{w})<0]}] \leq \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[y_i\hat{g}(\boldsymbol{w}_i)>0]} + \mathcal{O}\left(\sqrt{\frac{VC(G)+\log\frac{1}{\eta}}{m}}\right)$.

Remarks. The key point is that the student capacity is considerably larger since we typically train an entire DNN, and student complexity-based bound can be vacuous. While the guidance function does bound the student generalization error in terms of guide function complexity, there are strong caveats— we require the strong assumption of realizability on the entire domain, and additionally, while the guide function can witness student error, we are not in a position to precisely estimate it without additional training data. Furthermore, the RHS is a relaxed bound on the student training error. This motivates having a budget constraint to ensure that



(a) (b) (c)

Figure 6·2: (a): 2D Gaussians. Data distribution on $\mathbb{R}^2$. Teacher T learns the correct decision boundary with 3 layer NN and it is the global minima for this three-way classification task. While student S has many bad local minima, and one global minima that best describes the decision boundary with 2 layer NN. **(b)**: KD training. Loss contour plot shows the various local minima in the loss landscape. **(c)**: DiSK training. Loss contour plot shows the bad local minima no longer exist (wider minima, join two adjust minima, remove bad local minima).

student learns with small training error.

6.2 Related Work

We refer the reader to (Gou et al., 2021) for a comprehensive survey on knowledge distillation.

Response Matching. (Zeng and Martinez, 2000; Bucila et al., 2006) distill the response of an ensemble of classifiers into a single neural network by creating a pseudo-labeled dataset using the ensemble of classifiers. (Ba and Caruana, 2014) extend this to the setting with the neural network as the teacher. (Hinton et al., 2015) propose vanilla KD that distills knowledge from an ensemble of neural networks into a single network by matching their output logits. This work provided a simple recipe for aligning the teacher and student predictive distributions using the Kullback-Leibler

(KL) divergence. Recently, (Beyer et al., 2022) modify the KD procedure to include patient and consistent teacher resulting in substantial gains. Knowledge consistency is enforced by using the same aggressive data augmentation and image views in the student as in the teacher. Patience is promoted using a very long training schedule. This results in a computationally very expensive training process.

(Stanton et al., 2021) analyze response matching KD and suggests that difficulty in optimization leads to poor knowledge distillation. Thus, the teacher and student predictions do not always match, even on the training data. (Cho and Hariharan, 2019) study vanilla KD through the lens of mismatched student and teacher capacities. They show that small students are unable to mimic complex teachers. They propose early stopping teacher training as to remedy to achieve a student-learnable teacher. The above works fail miserably when the gap between student and teacher complexities is large. Specifically, the student cannot learn the complex teacher decision boundaries primarily due to the small student capacity. It becomes imperative to selectively choose only easy-to-learn data points and transfer the teacher knowledge from these points and ignore the hard-to-learn data points during distillation. Thus, our proposal targets the problem of severe capacity gaps between student and teacher models. Additionally, our experiments with standard student and teacher configurations show that DiSK is still competitive when the capacity difference is small.

Feature Matching. In response matching, teacher supervision is limited to its logits. We can enforce intermediate layer feature matching for refined teacher supervision. FitNets (Romero et al., 2015) extend the KD by including the feature matching in the middle layers. (Zagoruyko and Komodakis, 2017) use feature map attention as the teacher supervision. (Tung and Mori, 2019) preserve pairwise similarity in feature maps amongst data points during the distillation. (Chen et al., 2022) modify the student by projecting the student features onto the teacher feature space and by reusing the teacher classifier. While our work focuses primarily on selective distillation in vanilla KD for simplicity. We can easily extend the proposed framework to incorporate it into feature-matching distillation.

Privileged Information. (Vapnik and Izmailov, 2015) propose the 'learning under privileged information' (LUPI) framework wherein a support vector machine is trained using privileged information unavailable during the inference stage. Later, (Lopez-Paz et al., 2016) unify LUPI and vanilla KD into generalized distillation, wherein the teacher is learned using the privileged information. Next, the student is trained using the ground truth and the teacher labels. These works rely on privileged information in the application domain and are shown to work on toy setups. Since our guide function, g is available only during training, it can be thought of as privileged information from a teacher.

Curriculum Learning & Hard Instance Mining. Curriculum Learning (CL) (Bengio et al., 2009; Hacohen and Weinshall, 2019; Graves et al., 2017) sorts the data based on their hardness as measured by some scoring function (ex., predictive entropy, softmax margin score, etc.). It presents the data points during training in the order of increasing hardness. Similarly, Hard Instance Mining (HIM) (Zhou et al., 2020) reduces the weight of the easy example and increases the weight on hard inputs to promote hard-example learning. We point out that our method is only conceptually related to these works via input hardness. Our method helps the student with hard examples by providing explicitly discounted help g. We learn the helper function g (through teacher representation) that decides whether the student needs help on a given input. Thus, we do not prioritize learning hard examples keeping in mind the fact that student capacity is much smaller than the teacher.

6.3 Definition and Method

Notation. Let $\mathcal{X}$ and $\mathcal{Y} = \{1, \ldots, C\}$ be the feature and label spaces respectively, focusing on a C-class classification task. We assume that we have a training set of N i.i.d. data points $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We use symbols S and T to denote the student and teacher models respectively. Let $\mathbf{l}_S(\mathbf{x}) \in \mathcal{R}^{|\mathcal{Y}|}$ and $\mathbf{l}_T(\mathbf{x}) \in \mathcal{R}^{|\mathcal{Y}|}$ be the score vector, logits, predicted by S and T on input $\mathbf{x}$. We use τ as the temperature used to soften the probability distribution. We write the resulting softened student and teacher probabilities as $\mathbf{s}^\tau(\mathbf{x})$ and $\mathbf{t}^\tau(\mathbf{x})$, i.e.,

$$\mathbf{s}^\tau(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{l}_S(\mathbf{x})}{\tau}\right); \quad \mathbf{t}^\tau(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{l}_T(\mathbf{x})}{\tau}\right)$$

The standard prediction probabilities correspond to $\mathbf{s}^1(\mathbf{x})$ and $\mathbf{t}^1(\mathbf{x})$. We will use $\mathbf{s}_y^\tau(\mathbf{x})$ to denote the yth coordinate in $\mathbf{s}^\tau(\mathbf{x})$, and similarly $\mathbf{t}_y^\tau(\mathbf{x})$. The hard prediction of the student is $p_S(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \mathbf{s}_y^1(\mathbf{x})$, and similarly $p_T(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \mathbf{t}_y^1(\mathbf{x})$ for the teacher.

We use $g(\mathbf{x}) \in [0, 1]$ to denote the helper guide function for the student and teacher pair (S, T). Guide takes input $\mathbf{x}$ and any other feature processed by (S, T) pair and decides whether or not the student needs help on the input $\mathbf{x}$. Finally, we define ReLU activation as $(\cdot)_+ = \max(0, \cdot)$.

Vanilla Knowledge Distillation KD relaxes the $0-1$ error between the student predictions and the true labels y using the cross-entropy loss $\mathcal{L}_{CE}$. Then, KD denotes the distance between the student and teacher softened probability distributions using the KL divergence. We summarize the corresponding losses as,

$$\mathcal{L}_{CE}(\mathbf{s}) = -\frac{1}{N}\sum_{i=1}^{N} \log \mathbf{s}_{y_i}^1(\mathbf{x}_i); \quad \mathcal{L}_{KL}^\tau(\mathbf{s}) = -\frac{1}{N}\tau^2 \sum_{i=1}^{N}\sum_{y} \mathbf{t}_y^\tau(\mathbf{x}_i) \log \frac{\mathbf{s}_y^\tau(\mathbf{x}_i)}{\mathbf{t}_y^\tau(\mathbf{x}_i)}$$

For hyperparameters $\alpha \in [0, 1], \tau > 0$, KD minimizes a mixture of the above losses, as shown below

$$\mathcal{L}_{KD}^{\tau,\alpha}(\mathbf{s}) = \alpha\mathcal{L}_{CE}(\mathbf{s}) + (1-\alpha)\mathcal{L}_{KL}^{\tau}(\mathbf{s}). \tag{6.1}$$

Selective Knowledge Distillation. KD attempts to transfer the knowledge from the teacher to the student on all training data points, which is a sub-optimal objective when there is a capacity mismatch between the student and the teacher. Instead, we propose distilling selective knowledge (DiSK) to allow the student to selectively ignore some hard-to-learn data points during training, transferring the teacher's knowledge only on easy-to-learn inputs, and matching the learning to student capacity. Our objective is to minimize

$$\min_{\mathbf{s},g,\delta} \underbrace{\frac{1}{N}\sum_{i=1}^{N} \text{distance}(\mathbf{t}(\mathbf{x}_i);\phi(\mathbf{s},g)(\mathbf{x}_i))}_{\text{Distance between } T \text{ and } S \text{ with help of } g} \text{ subject to } \underbrace{\frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}_i)\mathbb{1}_{\{y_i \neq \arg\max_y \mathbf{s}_y(\mathbf{x}_i)\}} \leq \delta}_{\text{Support budget constraint on } g} \tag{6.2}$$

where ϕ interpolates student predictions based on the guide's help. The divergence term helps in minimizing the distributional distance between the teacher and student probabilities after the guide g is included. While the budget term in the optimization constrains the helper g to provide help only when necessary, the amount of help given to the student should be within the budget $\delta \in [0,1]$.

Function g Construction. As previously stated, we use the teacher's last layer features and soft predictions as input to the guide g. The guide is structured as a light-weight three-layer neural network with these inputs, with a sigmoid activation at the last layer. We re-emphasise that g is not used at inference time, and only aids training. More details are left to Appx.A.1.5.

Relaxed Losses, Lagrangian & Optimization Algorithm. We relax Eq. 6.2 and construct a Lagrangian by integrating the constraint into the minimization.

Budget constraint relaxation. We relax the indicator loss in the budget to a cross-entropy, and treat δ as a hyperparameter to get

$$\mathcal{L}^{\delta}_{budget}(\mathbf{s}, g) = \left[-\frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}_i)\log s^{1}_{y_i}(\mathbf{x}_i) - \delta\right]_{+} \tag{6.3}$$

We view the scaffold as a way for the student to interpolate the uncensored data. It suggests that a good initialization for the budget is the error of cross-entropy trained model when the student does not have the teacher supervision. Thus, we scan the budget in a small interval around this initialization.

Distillation objective. Motivated from KL loss, we construct a distance loss with guide function as,

$$\mathcal{L}^{\tau,K}_{dist}(\mathbf{s}, g) = -\frac{1}{N}\tau\tau_{\mathbf{t},\mathbf{s},\mathcal{D}}\sum_{i=1}^{N}\sum_{y} t^{\tau}_{y}(\mathbf{x}_i)\log\left(s^{\tau_{\mathbf{t},\mathbf{s},\mathcal{D}}}_{y}(\mathbf{x}_i) + 1_{y\in \text{top}_K(t^{\tau}(\mathbf{x}_i))}g(\mathbf{x}_i)\right) \tag{6.4}$$

We point out two modifications in the distillation loss. First, $\mathcal{L}_{dist}$ explicitly adds guide value to softened student probabilities in selected class indices. The class indices guide function adds value are picked as top K classes based on the teacher probabilities for any input $\mathbf{x}_i$ where K is a hyperparameter of our method. The rest of the class indices do not get any value from g. Second, we use different temperature parameters for teacher and student. Temperature parameter for teacher, τ, is a hyperparameter. The student temperature is found by minimizing the KL loss between teacher softened probabilities and the student softened probabilities over the training dataset, .i.e $\tau_{\mathbf{t},\mathbf{s},\mathcal{D}} = \arg\min_{\tau'}\sum_i KL(\mathbf{t}^{\tau}(\mathbf{x}_i), \mathbf{s}^{\tau'}(\mathbf{x}_i))$.

Similar to KD, we incorporate standard cross entropy loss between student model predictions and the ground truth labels for stability. We construct our Lagrangian by combining Eq. 6.3 and 6.4 as,

$$\mathcal{L}^{\tau,K,\delta,\alpha}_{DiSK}(\mathbf{s}, g, \lambda) = \alpha\mathcal{L}_{CE}(\mathbf{s}) + (1-\alpha)\mathcal{L}^{\tau,K}_{dist}(\mathbf{s}, g) + \lambda\mathcal{L}^{\delta}_{budget}(\mathbf{s}, g) \tag{6.5}$$

Algorithm 7 DiSK: Distilling Selective Knowledge.

```
Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, Teacher $\mathbf{t}$,
Parameters: $\tau$, $K$, $\alpha$, $\lambda_{\min}$, $\lambda_{\max}$, Number of iterations $R$, $\lambda_T$ cosine period, Budget $\delta$,
Initialize: $\mathbf{s}$, randomly initialize $g$, $\lambda = \lambda_{\min}$,
for $r = 1$ to $R$ do
    Randomly Shuffle Dataset $\mathcal{D}$
    $g \leftarrow \arg\min_g \mathcal{L}_{DiSK}^{\tau,K,\delta,\alpha}(\mathbf{s}, g, \lambda)$
    $\mathbf{s} \leftarrow \arg\min_{\mathbf{s}} \alpha\mathcal{L}_{CE}(\mathbf{s}) + (1-\alpha)\mathcal{L}_{dist}^{\tau,K}(\mathbf{s}, g)$
    $\lambda \leftarrow \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \times \frac{(1-\cos\frac{r \mod \lambda_T}{\lambda_T}\pi)}{2}$
end for
Return : $\mathbf{s}$
```

where α is a hyper-parameter and λ is the dual parameter of DiSK.

We optimize Obj. 6.5 using a primal dual update scheme as explained in Algorithm 7.

Primal Parameter Updates ($\mathbf{s}, g$*).* We learn the student $\mathbf{s}$ and the guide function g using alternating minimization. We approximate arg min with running SGD for a small number of epochs on the full dataset. In each iteration, we first learn the guide function g to select the data partition from which the knowledge needs to be distilled. Next, given the function g, we learn the student using the help g. We empirically found that not optimizing the student model on budget loss gives more stable results. Hence, we minimize the student model only on the distillation and cross-entropy losses.

Dual Parameter Update (λ*) Intuition.* Although it is tempting to optimize the above via a dual ascent and primal descent scheme (wherein the dual parameter λ is increased by residual term in the constraint until constraint satisfaction), recent work, (Sun and Sun, 2021), has proposed to decrease the λ in the non-convex regime. Inspired by this, we update the dual parameter by a fixed schedule between $[\lambda_{\min}, \lambda_{\max}]$. $\lambda_{\min}$ encourages exploration and allows student model to distill knowledge from all

Table 6.2: Model Statistics. We compute the storage (number of parameters) and computational requirements (number of multiply-addition operations) of the models.

Architecture		CIFAR-100		Tiny-Imagenet		Architecture	CIFAR-100	
		MACs	**Params**	**MACs**	**Params**		**MACs**	**Params**
Teacher	ResNet10-ℓ	64M	1.25M	255M	1.28M			
	ResNet10	253M	4.92M	1013M	5M	ResNet32x4	1083M	7.4M
	ResNet18	555M	11.22M	2221M	11.27M	Wide-ResNet-40-2	327M	2.25M
	ResNet34	1159M	21.32M	4637M	21.38M			
Student	ResNet10-xxs	2M	13K	8M	15K	ResNet8x4	177M	1.2M
	ResNet10-xs	3M	28K	12M	31K	ShuffleNetV2	44.5M	1.4M
	ResNet10-s	4M	84K	16M	90K	Wide-ResNet-16-2	101M	700K
	ResNet10-m	16M	320K	64M	333K	Wide-ResNet-40-1	83M	570K
						MobileNetV2x2	22M	2.4M

points. On the other hand, $\lambda_{\max}$ enforces the constraint and forces the student model to learn on uncensored inputs. We choose $R \approx 4\lambda_T$, so that the algorithm is exposed to a few exploratory periods. For the final period, we increase λ monotonically so that budget is more strictly enforced at termination.

Computational Efficiency. Algorithm 7 trains both student and guide networks. The guide network being small (three-layer MLP) relative to the student (CNN model), the additional cost in training the guide is relatively insignificant, and as such DiSK efficiency is similar to KD.

6.4 Experiments

We evaluate DiSK in various capacity mismatch scenarios on benchmark datasets.

Datasets. We use publicly available CIFAR-100 (Krizhevsky, 2009), Tiny-Imagenet (Le and Yang, 2015) datasets. CIFAR-100 contains 50K training and 10K test images from 100 classes with size $32 \times 32 \times 3$. While Tiny-Imagenet contains 100K training and 10K test images from 200 classes with size $64 \times 64 \times 3$. We provide the dataset setup and data augmentations used in detail in Appx. A.1.5.

Models. We evaluate standard convolutional models on these datasets including ResNet(He et al., 2016), Wide-ResNet(Zagoruyko and Komodakis, 2016), MobileNet(Sandler et al., 2018), and ShuffleNet(Ma et al., 2018). Table 6.2 shows the storage and computational requirements of all the models used in this work. We provide explicit model configurations in Appx. A.1.5, including the tiny models we generate from the ResNet architectures.

Methods. We study performance against standard cross-entropy (CE) based learning and the vanilla KD methods. For each method, we train models for 200 epochs using SGD as the optimizer with 0.9 momentum and 0.1 learning rate. See Appx. A.1.5 for more training details. We have recorded the mean in our results as the variance of 3 trials in our experiments is not larger than 0.1 in most cases.

We perform evaluations in different settings. Below, we explain individual setups. We cover more ablative experiments in Appx. A due to page limit.

Large Capacity Mismatch Setting. We distill knowledge from a teacher model into a student model where the student has much less capacity compared to the teacher model. We use four large capacity ResNet teachers and five tiny ResNet students and train these students using CE, KD, and DiSK methods. Performances, and the gains of DiSK are reported in Table 6.3.

Small Capacity Mismatch Setting. While DiSK has been designed for the scenario when student capacity is very low, we further evaluate it in the setting where teacher and student capacities are similar, to probe how far the power of the method extends. The model classes used are the standard choice for this scenario (Chen et al., 2022; Tung and Mori, 2019). Performance is reported in Table 6.4.

Table 6.4 further reports the results of the feature matching distillation methods: FitNets (Romero et al., 2015), SemCKD (Chen et al., 2021), and SimKD (Chen et al., 2022). Such methods can often outperform response matching KD on large students, due to student representations that are more aligned with the teacher, but typically at an increased training cost. While feature matching methods are not the main focus of our work (and in principle scaffolding idea can be extended to them), we observe that DiSK often improves upon their performance without any direct feature matching.

Experiment results. Below, we highlight salient features of DiSK based on empirical data.

DiSK outperforms the baselines uniformly across all datasets and student sizes. As shown by Table 6.3, DiSK significantly improves the student performance in CIFAR-100 and the (more challenging) Tiny-Imagenet dataset, respectively showing accuracy gains of up to 5% and 2% compared to KD. These gains are consistent across a wide range of student and teacher capacities.

DiSK achieves better performance with worse teachers than KD does with even the best teachers. In Table 6.3, we point out that the student performance increases for KD as the teacher complexity is increased for a given student. But note that for the same student, DiSK achieves much better performance with even worse teacher. For instance, for the ‘ResNet10-m‘ student, KD accuracy increases from 66.96% to 68.09% by using high capacity teachers. But ‘ResNet10-m‘ trained with even the worst teacher (‘ResNet10-ℓ‘) achieves 70.03% accuracy. This saves a lot of resources in any application as large teacher requires more training time, and larger compute resources.

DiSK is competitive even in small capacity difference setting. As shown by Table 6.4, DiSK does not loose its competitive edge over the KD even when the student is relatively similar sized as teachers, and shows gains of up to 2.5% relative to KD. We conjecture that the observed gains arise from the fact that DiSK provides scaffolding for hard points to the student in initial training stages, which promotes the student to learn easy examples first. As training progresses, DiSK removes the discounted help from hard inputs. As a result, the student evolves from simpler hypothesis to the ones consistent with both easy and hard inputs. This justifies our dual parameter (λ) update in Algorithm 7, wherein we periodically increase and decrease λ to enforce and relax the budget constraint.

DiSK students achieve near teacher accuracy while saving up to 8× *MACs &* 5× *Params.* As reported in Table 6.3, student ('ResNet10-m') trained with the teacher ('ResNet10-ℓ') achieves close to the teacher accuracy of 71.99%. In this process, it saves 4× compute and requires 4× less parameters. Similarly, student ('ShuffleNetV2') trained with the teacher ('ResNet32x4') achieves close to the teacher accuracy of 81.45%. In this process, it saves 24× compute and requires 5× less parameters.

DiSK cleverly selects a subset of datapoints and smoothens the loss landscape. As illustrated in Figure 6·1 and 6·2, DiSK judiciously selects a subset of hard-to-learn data points for the students and provide discounted help to the student focus on easily learnable inputs. As a result, it eliminates some bad local minima in the student loss-landscape, and smoothens tihs surface.

DiSK enables the student to reach saturation capacity. In Table 6.3, the performance of KD often suffers as the teacher size is increased, e.g., student ('ResNet10-s') accuracy decreases substantially with the teacher 'ResNet34' versus the teacher 'ResNet18'. In contrast, DiSK saturates the student performance across different teachers. For instance, student ('ResNet10-m') accuracy is $\approx 70\%$ for all the teachers. Thus, we point out that DiSK enables the student to reach saturation. This may be due to the fact that guide g identifies the same set of 'easy' points across different teachers.

Table 6.3: DiSK performance under large capacity mismatch on CIFAR-100 & Tiny-Imagenet: We draw mismatched teachers and students from the ResNet family, and report accuracy of CE trained teachers and students, performance of students distilled using KD and DiSK, and gains of the latter relative to KD.

Architecture		**CIFAR-100**					**Tiny-Imagenet**				
		Accuracy (%)					**Accuracy (%)**				
Teacher	**Student**	**Teacher**	**CE**	**KD**	**DiSK**	**Gain**	**Teacher**	**CE**	**KD**	**DiSK**	**Gain**
ResNet10-ℓ	ResNet10-xxs	71.99	32.05	32.64	**37.56**	**4.92**	52.14	17.44	17.59	**18.62**	**1.03**
	ResNet10-s		52.16	54.92	**58.14**	**3.22**		34.65	35.77	**37.43**	**1.66**
	ResNet10-m		65.24	66.96	**70.03**	**3.07**		44.74	46.01	**48.03**	**2.02**
ResNet10	ResNet10-xxs	75.25	32.05	34.25	**37.84**	**3.59**	56.04	17.44	17.96	**18.55**	**0.59**
	ResNet10-s		52.16	54.95	**58.36**	**3.41**		34.65	36.11	**37.37**	**1.26**
	ResNet10-m		65.24	67.27	**70.15**	**2.88**		44.74	46.08	**48.19**	**2.11**
ResNet18	ResNet10-xxs	76.56	32.05	34.16	**37.8**	**3.64**	62.48	17.44	17.47	**18.53**	**1.06**
	ResNet10-s		52.16	55.76	**58.11**	**2.35**		34.65	35.59	**37.5**	**1.91**
	ResNet10-m		65.24	68.09	**69.86**	**1.77**		44.74	45.91	**47.7**	**1.79**
ResNet34	ResNet10-xxs	80.46	32.05	33.93	**37.78**	**3.85**	63.06	17.44	17.67	**18.91**	**1.24**
	ResNet10-s		52.16	54.19	**58.02**	**3.83**		34.65	35.43	**37.68**	**2.25**
	ResNet10-m		65.24	66.78	**69.89**	**3.11**		44.74	45.89	**47.6**	**1.71**

Table 6.4: DiSK performance with small capacity mismatch on CIFAR-100. We pick standard student and teacher configurations used in the KD literature, and report accuracies and gains similarly to Table 6.3. Feature matching KD baselines are due to (Chen et al., 2022).

Architecture		**CIFAR-100**							
		Response Matching KD					**Feature Matching KD**		
		Accuracy (%)					**Accuracy (%)**		
Teacher	**Student**	**Teacher**	**CE**	**KD**	**DiSK**	**Gain**	**FitNet**	**SemCKD**	**SimKD***
ResNet32x4	ResNet8x4	81.45	73.89	76.25	**76.92**	**0.67**	74.32	76.23	78.08
	ShuffleNetV2		73.74	79.13	**80.23**	**1.1**	75.82	77.62	78.39
	Wide-ResNet-16-2		74.26	76.28	**77.67**	**1.39**	74.70	75.65	77.17
	MobileNetV2x2		69.24	76.05	**77.24**	**1.19**	73.09	73.98	75.43
Wide-ResNet-40-2	ResNet8x4	78.41	73.89	75.15	**76.05**	**0.9**	75.02	75.85	76.75
	ShuffleNetV2		73.74	75.81	**78.33**	**2.52**	-	-	-
	Wide-ResNet-40-1		72.81	74.44	**75.92**	**1.48**	74.17	74.4	75.56
	MobileNetV2x2		69.24	73.92	**76.32**	**2.40**	-	-	-

*SimKD accuracy is not emphasized as it employs additional layers beyond the given student architecture and thus not directly comparable to other methods.

Chapter 7

Future Work

This chapter gives some future directions in limited data settings. We mainly focus on federated learning and provide preliminary results in improving performance.

7.1 Synthetic data generation

Synthetic data generation can be an essential technique to increase the data amount in limited training data settings. Different synthetic data-generating methods include generative adversarial networks (GAN) (Goodfellow et al., 2020), diffusion models (Sohl-Dickstein et al., 2015), and stable diffusion (Rombach et al., 2022). GANs train a generator and discriminator model simultaneously, where the discriminator aims to detect synthetically generated examples. Diffusion models impose hierarchical autoencoder models to have an iterative process. Recently, stable diffusion further advances diffusion models and tackles high-resolution images. We refer to surveys for a detailed investigation (Nikolenko, 2019; Lu et al., 2023).

Potential benefits.

- Currently, stable diffusion generates very real-looking images. Using real-looking synthetic images might boost the performance.
- The same idea can be extended to non-federated learning settings where the data is limited.

Potential issues.

- Data generation is usually slow for stable diffusion models. We want to investigate more on the fast inference of data generation techniques to be used in federated learning.

- Generating images based on client data might lead to privacy problems. For instance, (Carlini et al., 2023) can generate an image almost identical to the training data. Such information leakage would raise privacy concerns in federated learning.

7.2 Improving generalization with side losses

The data in each client is limited in federated learning, and a DNN easily overfits if proper precautions are not taken. We prose to prevent overfitting by utilizing already in-use side objectives from different disciplines. We plan on incorporating two different losses explained in the following paragraphs.

Robustness. We plan to add a robustness loss similar to the ones in self-supervised learning methods (Chen and He, 2021; Grill et al., 2020). Robustness loss aims to align the predictions of two different data augmentations of the same image, allowing the model to be robust to the augmentations that promote the generalization.

How to do it? While we can use sophisticated constrained optimization techniques, we preliminary show results on a simple fixed Lagrangian setting by adding the robustness loss to the cross-entropy loss for each device. We leave the investigation as future work.

Distillation. We slightly change the problem and assume that a more powerful teacher model is available to the clients during training. We add distillation loss in the cross-entropy training. We motivate this problem in the following paragraphs.

Would clients have resources for the teacher model? Clients would only forward propagate data using the teacher model, which costs less than backward propagation. Hence, clients can support a more complex architecture if the teacher requires similar resources as backpropagating the client model.

Is this a practical problem? Consider a setting where we would like to customize models for each client using a small number of gradient steps, as in MAML (Finn et al., 2017). In such a setting, an already trained teacher model can not be used for the clients since fine-tuning the teacher model is not supported due to its size. Therefore, we need to train a simpler client model with the help of the teacher model.

How to do it? We test the naive idea of using the vanilla KD distillation loss (Hinton et al., 2015) during the local training instead of cross-entropy loss. We can use novel ideas like DiSK (Kag et al., 2023) or SimKD (Chen et al., 2022). We leave the investigation as future work.

Finally, we can combine the robustness and the distillation to improve the performance further. The modifications explained modify the local cross-entropy loss. Utilizing advanced federated learning algorithms such as FedDyn (Acar et al., 2021a) lead to better models.

7.2.1 Definition and Loss Formulation

We follow a similar notation as in Chapter 2. We target the following objective,

$$\arg\min_{\boldsymbol{w}\in\mathbb{R}^d}\left[\frac{1}{m}\sum_{k\in[m]}L_k(\boldsymbol{w})\right]; \quad L_k(\boldsymbol{w})=\frac{1}{N_k}\sum_{i\in[N_k]}\ell(\boldsymbol{w};(\boldsymbol{x}_k^i,y_k^i)) \tag{7.1}$$

where, m is number of devices, $\boldsymbol{w}$ is the parameter vector of the neural network model, ℓ is cross-entropy loss, $\{\boldsymbol{x}_k^i, y_k^i\}_{i\in[N_k]}$ is the local dataset at device k and $L_k(\boldsymbol{w})$ is the kth device's empirical loss. Chapter 2 introduces FedDyn as a communication-efficient way to solve the federated learning problem with provable convergence guarantees.

This chapter changes the local losses using robustness and distillation. We summarize the local loss changes in the following.

Robustness. We add a loss that penalizes different model outputs for two augmentations of the same data point. Briefly, during training, we change the local loss at device k to,

$$\begin{aligned}&\alpha\left[\frac{1}{N_k}\sum_{i\in[N_k]}\ell(\boldsymbol{w};(\boldsymbol{x}_k^i,y_k^i))\right]\\&+(1-\alpha)\left[\frac{1}{N_k}\sum_{i\in[N_k]}E_{\hat{\boldsymbol{x}}_k^i,\dot{\boldsymbol{x}}_k^i\sim\mathcal{T}(\boldsymbol{x}_k^i)}\tau^2KL\left(\boldsymbol{w}^\tau\left(\dot{\boldsymbol{x}}_k^i\right),\boldsymbol{w}^\tau\left(\hat{\boldsymbol{x}}_k^i\right)\right)\right]\end{aligned}\tag{7.2}$$

where, τ is temperature parameter, $\hat{\boldsymbol{x}}_k^i$ and $\dot{\boldsymbol{x}}_k^i$ are different augmentations of the same data point $\boldsymbol{x}_k^i$, $\boldsymbol{w}^\tau(\boldsymbol{x})$ is the softened model prediction probability vector for datapoint $\boldsymbol{x}$, the first term is the standard cross-entropy loss, the second term is the robustness loss that aligns model outputs for different augmentations of the same input, and $\alpha\in[0,1]$ is a hyperparameter that trades offs between two terms.

Clarification for the cross-entropy loss. We also draw an augmentation of the data points in constructing the local cross entropy loss in Eq. 7.1. Since data augmentation is a common technique, it is not emphasized in the algorithm details in Chapter 2. We explicitly state the augmentation for robustness loss since the construction uses different augmentations by definition, but we keep the cross-entropy one as it is for simplicity.

Distillation. Similar to robustness, we add vanilla distillation loss to prevent overfitting. During training, we change the local loss at device k to,

$$\alpha\left[\frac{1}{N_k}\sum_{i\in[N_k]}\ell(\boldsymbol{w};(\boldsymbol{x}_k^i,y_k^i))\right]+(1-\alpha)\left[\frac{1}{N_k}\sum_{i\in[N_k]}\tau^2KL\left(\overline{\boldsymbol{w}}^\tau\left(\boldsymbol{x}_k^i\right),\boldsymbol{w}^\tau\left(\boldsymbol{x}_k^i\right)\right)\right]\tag{7.3}$$

where, τ is temperature parameter, $\overline{\boldsymbol{w}}$ is the teacher model, $\boldsymbol{w}^\tau\left(\boldsymbol{x}\right)$ is the softened output probability vector of the model on data point $\boldsymbol{x}$, the first term is the standard cross-entropy loss, the second term is the distillation loss that encourages the model to follow softened teacher outputs, and $\alpha \in [0, 1]$ is a hyperparameter that trades offs between two terms.

Robustness and Distillation. In experiments, we first separately test the robustness and the distillation losses. We further combine both losses and try the merged version. We give the integrated version as well for the sake of completeness. During training, we change the local loss at device k to,

$$\begin{aligned} &\alpha \left[\frac{1}{N_k} \sum_{i \in [N_k]} \ell(\boldsymbol{w}; (\boldsymbol{x}_k^i, y_k^i)) \right] + (1-\alpha) \left[\frac{1}{N_k} \sum_{i \in [N_k]} \tau^2 KL \left(\overline{\boldsymbol{w}}^{\tau} \left(\boldsymbol{x}_k^i \right), \boldsymbol{w}^{\tau} \left(\boldsymbol{x}_k^i \right) \right) \right] \\ &+ (1-\alpha) \left[\frac{1}{N_k} \sum_{i \in [N_k]} E_{\hat{\boldsymbol{x}}_k^i, \dot{\boldsymbol{x}}_k^i \sim \mathcal{T}(\boldsymbol{x}_k^i)} \tau^2 KL \left(\boldsymbol{w}^{\tau} \left(\dot{\boldsymbol{x}}_k^i \right), \boldsymbol{w}^{\tau} \left(\hat{\boldsymbol{x}}_k^i \right) \right) \right] \end{aligned} \tag{7.4}$$

How to optimize the losses? We introduced some modifications in the local losses for training. We can use any federated optimization method to train a server model, such as FedAvg (McMahan et al., 2017a), or FedDyn (Acar et al., 2021a).

7.2.2 Experiments

We test the proposed variations using a real-world dataset. We give the details of the experiment in the following paragraphs.

Dataset. We use CIFAR100 (Krizhevsky, 2009) as our dataset in the experiments. We split the dataset identically and independently into 100 devices, where each device gets a random 5 data points per class in a total of 500. We activate 10% of the devices during federated learning training in each participation round.

Model architecture. We use PreActResNet (He et al., 2016b) as our client model. ResNet family operates on batch norm layers by definition, which store running mean and std estimates. Sharing running estimates might lead to privacy concerns, so we replace all batch norm layers with group norm (Wu and He, 2018) layers that do not store running estimates. We use PreActResNet-18 ($\approx$ 11M parameters) as a client model. For distillation loss, we use PreActResNet-34 ($\approx$ 21M parameters) as a teacher model.

Methods. We test different variants of the optimization algorithms and the losses. We report performances of:

- the standard FedAvg baseline with the standard cross-entropy loss,
- the FedDyn method with the standard cross-entropy loss,
- the FedAvg method with the robustness idea as the local losses in Eq. 7.2,
- the FedAvg method with the distillation idea as the local losses in Eq. 7.3, and
- combination of all ideas as the FedDyn method with the robustness and the distillation idea as the local losses in Eq. 7.4.

Parameters. We use a batch size of 50, local epochs of 5, weight decay of $5 \times 1e - 4$, SGD optimizer, an initial learning rate of 0.1, learning rate decay of 0.998 for each round a temperature parameter of 3.5 for robustness and distillation variants. α parameter in the FedDyn method is set to 0.1. α trade-off parameter of 0.1 is used for Eq. 7.3 and Eq. 7.4. α trade-off parameter of 0.5 is used for Eq. 7.2. We use extensive augmentation techinques as RandomCrop, ‘RandomHorizontalFlip‘, AutoAugment (Cubuk et al., 2019), Cutout (DeVries and Taylor, 2017) with a window size of 16, and Mean-Std-Normalization. We compare the federated models to the centralized training. In centralized training, we use a batch size of 128, weight decay of $5 \times 1e - 4$, a cosine scheduler learning rate where the initial learning rate is 0.1, and train the model for 300 epochs.

Evaluation. Decreasing communication costs is the ultimate goal of federated learning. To compare methods, we fix a common target test accuracy. Then, we

Table 7.1: CIFAR100, 100 devices, 10% participation rate, IID split. Comparison of methods. The number of communication rounds for each method to get the same accuracy is reported. The gain to FedAvg with cross-entropy baseline is given in parentheses.

Accuracy	FedAvg CE	FedDyn CE	FedAvg Distillation	FedAvg Robustness	FedDyn Distillation Robustness
67.0	1634 (1.00)	419 (3.90)	403 (4.05)	808 (2.02)	**250 (6.54)**
66.0	1071 (1.00)	399 (2.68)	381 (2.81)	742 (1.44)	**241 (4.44)**

report the number of communication rounds to achieve the target accuracy for all methods.

Results. We plot the convergence curves of the methods in Figure 7·1. We report the corresponding number of communication rounds to achieve a common target accuracy in Table 7.1.

Centralized training. If we centralize the dataset, the client model (ResNet18 with group normalization layers) gets to 74.2% accuracy with cross-entropy training. The model achieves 81.2% and 77.8% if we train using the distillation loss (Eq. 7.3) and the robustness loss (Eq. 7.2), respectively.

FedDyn-based optimization gets to close centralized performance. FedDyn with CE loss training achieves 71.8% accuracy at 1200 communication rounds, whereas FedAvg with CE training can only get 66.4%. Similarly, FedDyn with distillation and robustness losses training gets to 81.2% accuracy. FedDyn de-biases the local training objective, leading to high communication savings to achieve near-centralized performance levels.

Changing local loss from cross-entropy to distillation or robustness improves training. Keeping the optimization method as FedAvg and only changing cross-entropy loss to distillation or robustness leads to high communication savings and better final models. For instance, as reported in Table 7.1, FedAvg with the robustness loss leads

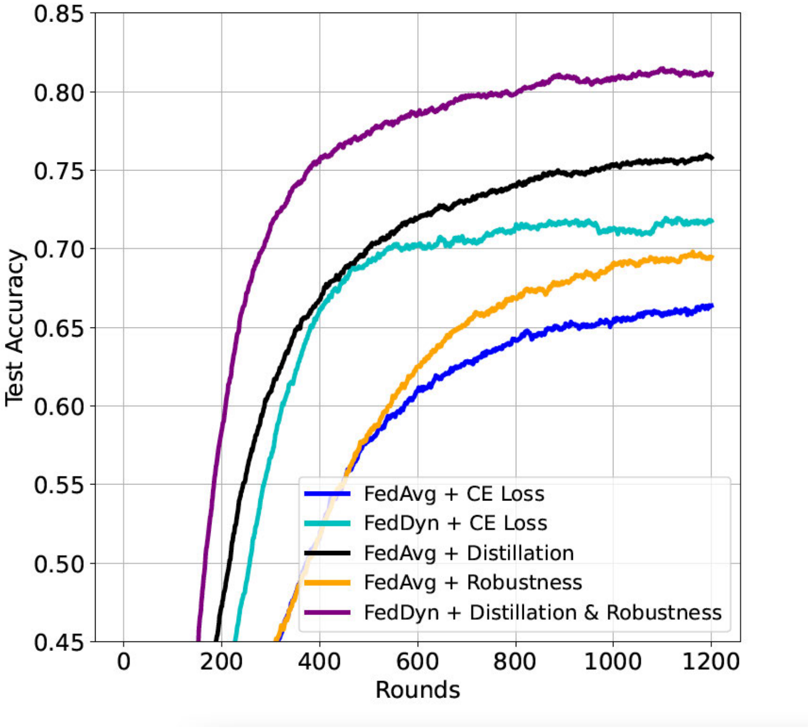


Figure 7·1: CIFAR100, 100 devices, 10% participation rate, IID split. Convergence curves.

to 2× gain over FedAvg with cross-entropy loss to achieve test accuracy of 67%. Figure 7·1 shows that FedAvg with the distillation loss achieves 75.8% accuracy, whereas FedAvg with the cross-entropy loss gets to 66.4% accuracy.

Distillation leads to a better model compared to robustness. Table 7.1 and Figure 7·1 show that distillation is more powerful than robustness. One explanation is that we can access a powerful teacher model for distillation-based training, whereas robustness-based training only uses extra realizations of the augmentations. For instance, Table 7.1 shows that FedAvg with distillation loss leads to 2× gain over FedAvg with robustness loss to achieve 67% test accuracy.

FedDyn-based optimization with both losses gives the best model. Combining all innovations leads to the best model in training. Table 7.1 reports that FedDyn with both distillation and robustness losses leads to a 6.5× gain over the standard baseline of FedAvg with cross-entropy loss. Moreover, Figure 7·1 shows that FedDyn with both distillation and robustness losses achieves as much as 81.2%, significantly higher than other methods within the 1200 communication rounds.

Next Steps. We list potential next steps in the following.

- The experiments are conducted using naive variants of distillation and fixed Lagrangian ideas. Using more sophisticated ideas might improve the performance.
- Only the IID split is tested in the experiments. Comparing methods in more challenging non-IID splits would make the differences between the methods more straightforward.
- We use only the robustness idea to improve generalization without a powerful teacher. Testing with different loss formulations, such as stability, might improve performance.

Chapter 8

Conclusions

This thesis focuses on the algorithms in limited data settings. Each chapter targets a different problem and proposes solutions that avoid overfitting by training generalizable models. We test the proposed algorithms with real-world datasets and show significant gains compared to the competitors. We highlight the benefits of our approaches in the following.

- *Chapter 2.* Our proposed method, FedDyn, focuses on the data heterogeneity problem in federated learning. As a solution, FedDyn adds a novel dynamic regularizer to local loss functions for each device without any extra transmission costs. We further give convergence guarantees of our method by being agnostic to the heterogeneity levels.

- *Chapter 3.* Our method, PFL, targets the personalization problem in federated learning. Our solution has provable convergence guarantees for arbitrary heterogeneous settings and further increases the users' privacy by supporting anonymization of the local label set.

- *Chapter 4.* We tackle the supporting different architectures problem in federated learning and propose FedHen as a solution. FedHen places a hierarchy between different architectures and trains simple architectures as a subset of the complex ones in each device without increasing communication costs.

- *Chapter 5.* Our method, MOML, targets the memory footprint complexity problem in online meta-learning. MOML successfully summarizes the experiments from earlier task rounds so that it discards the data of the earlier rounds, preventing the memory footprint from growing linearly with more rounds. We give a sub-linear regret analysis of our method.

- *Chapter 6.* We focus on the distillation problem where a powerful, already-trained teacher model helps train a student model. Our proposed method, DiSK, selectively distills knowledge from the teacher model on the learnable training points and ignores the over-capacity information on the hard-to-learn data points.

Finally, we mention some interesting future directions with preliminary results. Chapter 7.1 is about improving training using the recent data generation techniques with an application in federated learning. Chapter 7.2 proposes to change local losses in federated learning and conducts a preliminary empirical investigation where our approaches lead to significant communication gains.

Appendix A

Appendix

A.1 Experiment Details and More Ablations

A.1.1 FedDyn

Synthetic Data

Dataset. We introduce a synthetic dataset to reflect different properties of FL by using a similar process as in (Li et al., 2020a). The datapoints $(\boldsymbol{x}_j, y_j)$ of device i are generated based on $y_j = \arg\max(\boldsymbol{\theta}_i^* \boldsymbol{x}_j + \boldsymbol{b}_i^*)$ where $\boldsymbol{x}_j \in \mathbb{R}^{30\times1}$, $y_j \in \{1, 2, \ldots 5\}$, $\boldsymbol{\theta}_i^* \in \mathbb{R}^{5\times30}$, and $\boldsymbol{b}_i^* \in \mathbb{R}^{5\times1}$. $(\boldsymbol{\theta}_i^*, \boldsymbol{b}_i^*)$ tuple represents the optimal parameter set for device i and each element of these tuples are randomly drawn from $\mathcal{N}(\mu_i, 1)$ where $\mu_i \sim \mathcal{N}(0, \gamma_1)$. The features of datapoints are modeled as $(\boldsymbol{x}_j \sim \mathcal{N}(\nu_i, \sigma))$ where σ is a diagonal covariance matrix with elements $\sigma_{k,k} = k^{-1.2}$ and each element of ν_i is drawn from $\mathcal{N}(\beta_i, 1)$ where $\beta_i \sim \mathcal{N}(0, \gamma_2)$. The number of datapoints in device i follows a lognormal distribution with variance γ_3. In this generation procees, γ_1, γ_2 and γ_3 regulate the relation of the optimal models for each device, the distribution of the features for each device and the amount of datapoints per device respectively.

We simulate different settings by allowing only one type of heterogeneity at a time and disabling the randomness from the other two. For instance, if we want to disable type 1 heterogeneity, we draw one single set of optimal parameters $(\boldsymbol{\theta}^*, \boldsymbol{b}^*) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$ and use it to generate datapoints for all devices. Similarly, ν_i is set to 0 to disable type 2 heterogeneity and γ_3 is set to 0 to disable type 3 heterogeneity. We consider four settings in total, including type 1, 2, and 3 heterogeneous as well as a homogeneous setting. The number of devices is set to 20 and the number of datapoints per device is on average 200 in the generation process.

Models. We test FedDyn, SCAFFOLD, FedAvg and FedProx using a multiclass logistic classification model with cross entropy loss. We keep batch size to be 10, weight decay to be 10^{-5}.

We test learning rates in $[1, .1]$ and epochs in $[1, 10, 50]$ for all three algorithms. α parameter of FedDyn is chosen among $[.1, .01, .001]$; K parameter of SCAFFOLD is searched in $[20, 200, 1000]$ which corresponds to the same amount of computation using above epoch list; and μ regularization hyperparameter of FedProx in $[0.01, .0001]$.

Table A.4 reports the number models transmitted relative to one round of FedAvg to achieve the target training loss for best hyperparameter selection in various settings with 10% device participation. As shown, FedDyn leads to communication savings in each of the settings in range 1.1× to 7.6×.

Real Data

Datasets. MNIST, EMNIST-L, CIFAR-10 and CIFAR-100 are used for image classification tasks and Shakespeare dataset is used for a next character prediction task. The image size is $(1\times28\times28)$ in MNIST and EMNIST; $(3\times32\times32)$ in CIFAR-10 and CIFAR-100 with overall 10 classes in MNIST and CIFAR-10; 62 classes in EMNIST; and 100 classes in CIFAR-100. We choose the first 10 letters from the letter section of EMNIST (named it as EMNIST-L) similar to (Li et al., 2020a) work. Features in Shakespeare dataset consists of 80 characters and labels are the following characters. Overall, there are 80 different labels for datapoints.

We use the usual train and test splits for MNIST, EMNIST-L, CIFAR-10 and CIFAR-100. The number of training and test samples of the benchmark datasets are summarized in Table A.1.

To generate IID splits, we randomly divide training datapoints and assign them to devices. For non-IID splits, we utilize the Dirichlet distribution as in (Yurochkin et al., 2019). Firstly, a vector of size equal to the number of classes are drawn using Dirichlet distribution for each device. These vectors correspond to class priors per devices. Then one label is sampled based on these vectors for each device and an image is sampled without replacement based on the label. This process is repeated until all datapoints are assigned to devices. The procedure allows the label ratios of each device to follow a Dirichlet distribution. The hyperparameter of Dirichlet distribution corresponds to statistical heterogeneity level in the device datapoints. Overall, for a 100 device experiment, each device has 600, 480, 500 and 500 datapoints in MNIST, EMNIST-L, CIFAR-10 and CIFAR-100 respectively. For these datasets, three different federated settings are generated including an IID and two non-IID Dirichlet settings with .6 and .3 priors. Figure A·3 shows the heterogeneity levels for MNIST dataset in these different settings. The amount of most occurred class labels that consume 40%, 60% and 80% of device data are shown in the histogram plots. For example, every class label is equally represented in IID setting hence 4, 6 and 8 classes occupy 40%, 60%, and 80% of the local datapoints for each device. If we consider non-IID settings, we see 80% of local data belongs to mostly 4 or 5 different classes for Dirichlet .6; and 3 or 4 different classes for Dirichlet .3 settings.

To generate unbalanced data, we sample datapoint amounts from a lognormal distribution. Controlling the variance of lognormal distribution gives unbalanced data per devices. For instance, in CIFAR-10, balanced and unbalanced data settings have standard deviation of data amounts among devices as 0 and 0.3 respectively.

LEAF (Caldas et al., 2018) is used to generate the Shakespeare dataset used in this work. The LEAF framework allows to generate IID as well as non-IID federated settings. The non-IID dataset is the natural split of Shakespeare where each device corresponds to a role and the local dataset contains this role's sentences. The IID dataset is generated by combining the sentences from all roles and randomly dividing them into devices. In this work, we consider 100 devices and restrict number of datapoints per device to 2000.

Models. We use fully connected neural network architectures for MNIST and EMNIST-L. Both models take input images as a vector of 784 dimensions followed by 2 hidden layers and a final softmax layer. The number of neurons in the hidden layers are 200 and 100 for MNIST and EMNIST-L respectively. These models achieve 98.4% and 95.0% test accuracy in MNIST and EMNIST-L if trained on datapoints from all devices. The model considered for MNIST is the same model used in original FedAvg work (McMahan et al., 2017a).

For CIFAR-10 and CIFAR-100, we use a CNN consisting of two convolutional layers with 64 5×5 filters, two 2×2 max pooling layers, two fully connected layers with 394 and 192 neurons, and finally a softmax layer. The models achieve 85.2% and 55.3% test accuracy in CIFAR-10 and CIFAR-100 respectively. Our CNN model is similar to the used for CIFAR-10 in the original FedAvg work (McMahan et al., 2017a), except that we don't use Batch Normalization layers.

For the next character prediction task (Shakespeare), we use an LSTM. The model converts an 80 character long input sequence to a 80×8 sequence using an embedding. This sequence is fed to a two layer LSTM with hidden size of 100 units. The output of stacked LSTM is passed to a softmax layer. Overall, this architecture achieves a test accuracy of 50.8% and 51.2% in IID and non-IID settings, respectively, if trained on data from all devices. We report both IID and non-IID performance here because the datasets are randomly regenerated out of the whole Shakespeare writing hence train and test split is different for both cases. This Neural Network model is the same model used in the original FedProx study (Li et al., 2020a).

In passing, we note here that, we are not after state of the art model performances for these datasets, our aim is to compare the performances of these models in federated setting using FedDyn and other baselines.

Hyperparameters. We consider different hyperparameter configurations for different setups and datasets. For all the experiments, we fix batch size as 50 for MNIST, CIFAR-10, CIFAR-100 and EMNIST-L datasets and as 100 for Shakespeare dataset.

We note here that μ, α and K hyperparameters are used only in FedProx, FedDyn and SCAFFOLD respectively. K is the equivalent of epoch for SCAFFOLD algorithm and we searched K values to have the same amount of local computation as in other methods. For example, if each device has 500 datapoints, batch size is 50 and epoch is 10, local devices apply 100 SGD steps which is equivalent to K being 100.

MNIST. As for the 100 devices, balanced data, full participation setup, hyperparameters are searched for all algorithms in all IID and Dirichlet settings for a fixed 100 communication rounds. The search space consists of learning rates in $[.1, .01]$, epochs in $[10, 20, 50]$, Ks in $[120, 240, 600]$, μs in $[1, .01, .0001]$ and αs in $[.001, .01, .03, .1]$. Weight decay of 10^{-4} is applied to prevent overfitting and no learning rate decay across communications rounds is used. The selected configuration for FedAvg is .1 learning rate and 20 epoch; for FedProx is .1 learning rate and .0001 μ; for FedDyn is .1 learning rate, 50 epoch and .01 α; and for SCAFFOLD is .1 learning rate and 600 K for all IID and Dirichlet settings. These configurations are fixed and their performances are obtained for 500 communication rounds.

For the partial participation, 100 devices, balanced data setup, the selected configuration for FedAvg is .1 learning rate and 10 epoch; for FedProx is .1 learning rate and .0001 μ; for FedDyn is .1 learning rate, 50 epoch and .01 α; and for SCAFFOLD is .1 learning rate and 600 K for all IID and Dirichlet settings except that α is chosen to be .03 for 10% IID setting. 0.998 learning rate decay per communication round is used and weight decay of 10^{-4} is applied to prevent overfitting for all methods.

For the centralized model, we choose learning rate as .1, epoch as 150 and learning rate is halved in every 50 epochs.

EMNIST-L. We used similar hyperparameters as in MNIST dataset. The configuration for FedAvg is .1 learning rate and 20 epoch; for FedProx is .1 learning rate and 10^{-4} μ; for FedDyn is .1 learning rate, 50 epoch and 0.005 α; and for SCAFFOLD is .1 learning rate and 500 K for all IID and Dirichlet full participation settings.

The selected configuration for FedAvg is .1 learning rate and 10 epoch; for FedProx is .1 learning rate and .0001 μ; for FedDyn is .1 learning rate, 50 epoch; and for SCAFFOLD is .1 learning rate and 500 K for all IID and Dirichlet partial settings. α is chosen to be .003 for 10% and 1% IID; .005 for 10% Dirichlet .6 and 1% Dirichlet .3 ; .001 for 1% Dirichlet .6 and .01 for 10% Dirichlet .3 settings. 0.998 learning rate decay per communication round is used and weight decay of 10^{-4} is applied to prevent overfitting for all methods.

For the centralized model, we choose learning rate as .1, epoch as 150 and learning rate is halved in every 50 epochs.

CIFAR-10. The same hyperparameters are applied to all the CIFAR-10 experiments, including: 0.1 for learning rate, 5 for epochs, and 10^{-3} for weight decay. The learning rate decay is selected from the range of $[0.992, 0.998, 1.0]$. The α value is selected from the range of $[10^{-3}, 10^{-2}, 10^{-1}]$ for FedDyn. The μs value is selected from the range of $[10^{-2}, 10^{-3}, 10^{-4}]$.

For the centralized model, we choose learning rate as .1, epoch as 500 and learning rate decay as .992.

CIFAR-100. The same hyperparameters are applied to the CIFAR-100 experiments with 100 devices. including: 0.1 for learning rate, 5 for epochs, and 10^{-3} for weight decay. The learning rate decay is selected from the range of $[0.992, 0.998, 1.0]$.

The α value is selected from the range of $[10^{-3}, 10^{-2}, 10^{-1}]$ for FedDyn. The μs value is selected from the range of $[10^{-2}, 10^{-3}, 10^{-4}]$.

As for 500 device, balanced data, 10% participation, IID setup, .1 learning rate, .0001 μ, 10^{-3} weight decay applied. Epochs in $[2, 5]$ and corresponding Ks in [4,10] searched. αs in $[.1, .01, .001]$ are considered for FedDyn. Epoch of 2 is selected for FedDyn, FedAvg and FedProx, K of 4 is selected for SCAFFOLD. .01 α value is selected for FedDyn. The same parameters are chosen for 500 device, balanced data, 10% participation, Dirichlet .3 setup.

As for 100 device, unbalanced data, 10% participation, IID and Dirichlet .3 settings, epoch of 2 is selected for FedDyn, FedAvg and FedProx, K of 20 is selected for SCAFFOLD. .1 α value is applied for FedDyn. .0001 μ is used in FedProx.

For the centralized model, we choose learning rate as .1, epoch as 500 and learning rate decay as .992.

Shakespeare. As for 100 devices, balanced data, full participation setup, the hyperparameters are searched with all combinations of learning rate in [1], epochs in $[1, 5]$, Ks in $[20, 100]$, μs in $[.01, .0001]$ and αs in $[.001, .009, .01, .015]$. Weight decay of 10^{-4} is applied to prevent overfitting and no learning rate decay across communications rounds is used. The selected configuration for FedAvg is 1 learning rate and 5 epoch; for FedProx is 1 learning rate, 5 epoch and .0001 μ; for FedDyn is 1 learning rate, 5 epoch and .009 α; and for SCAFFOLD is 1 learning rate and 100 K in IID and non IID settings.

For the partial participation, 100 devices, balanced data setup, we choose 1 learning rate and 5 epoch for FedAvg; 1 learning rate, 5 epoch and .0001 μ for FedProx; 1 learning rate and 100 K for SCAFFOLD; and 1 learning rate and 5 epoch for FedDyn in all cases. α is .015 and .001 for 10% and 1% settings respectively. No learning rate decay is applied for 10% settings and a decay of .998 is applied for 1% settings.

Weight decay of 10^{-4} is applied to prevent overfitting.

For the centralized model, we choose learning rate as 1, epoch as 150 and learning rate is halved in every 50 epochs.

Additionally, we performed gradient clipping to prevent overflow in weights for all methods. We found out that, this increases stability of algorithms.

Convergence Plots. We give convergence plots of experiments. The convergence plots of moderate and large number of devices in different device distributions are shown in Figure A·4 and A·5 for CIFAR-10 and CIFAR-100 datasets. Similarly, convergence curves of different participation levels and distributions are plotted in Figure A·6, A·7, A·8, A·9 and A·10 for all datasets. Finally, Figure A·12 and A·13 show convergence plots for balanced data and unbalance data in different device distributions.

We emphasize that convergence curves show accuracy achieved with respect to rounds communicated. However, the metric we want to minimize, the amount of information transmitted, is not the same as number of communication rounds. For instance, SCAFFOLD transmits two models including state of devices per communication round. This difference is accounted in the tables.

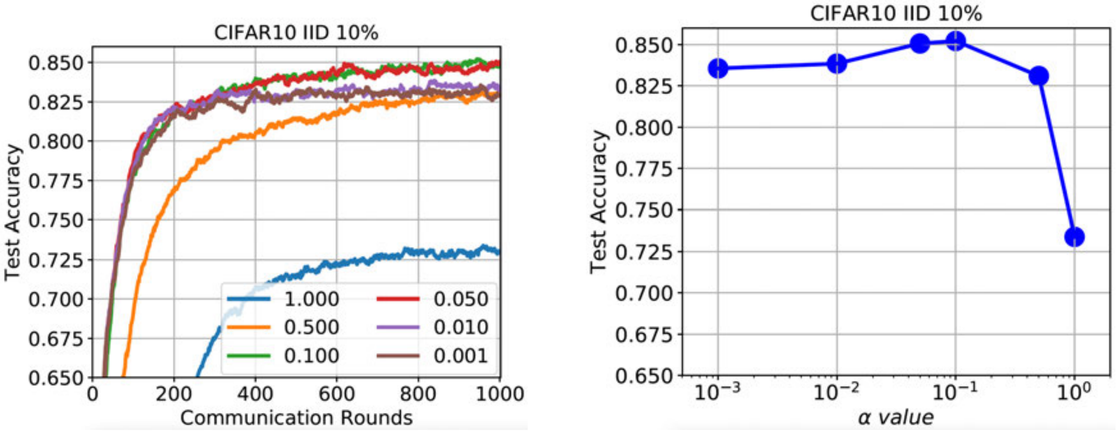
We observed that averaging all device models gives more stable convergence curves hence we report the performance of the average model from all devices in each communication round. We note that we do not modify the algorithms, this part is only for reporting purposes.

Additional to experiments stated, we test our algorithm with a more complex model. We consider ResNet18 (He et al., 2016a) structure on CIFAR-10 IID, 100 devices, balanced data, 10% participation setting. Batch normalization layers have inherent statistics which can be problematic in FL. Therefore, we use group normalization (Wu and He, 2018) instead of Batch normalization in ResNet18. The convergence curves are shown in Figure A·11. FedDyn still outperforms the baseline methods in a higher capacity model setup.

Sensitivity Analysis of FedDyn

α is an important parameter of FedDyn. Indeed, it is the only hyperparameter of the algorithm when devices have access to an optimization solver. In theory, α balances two problem dependent constants as shown in Theorem 5, Theorem 6 and Theorem 7. Consequently, optimal value of α depends on these constants. Since these constants are independent of T, the value of α does not asymptotically affect convergence rate.

To test sensitivity, we consider CIFAR-10, IID, 100 devices, 10% participation setting. Figure A·1.a shows convergence plots for different α configurations while keeping all other parameters constant in FedDyn. Figure A·1.b presents the best achieved test accuracy with respect to different α values. We see that best test performance is obtained when $\alpha = 10^{-1}$. We note that all configurations converge, but some of them converges to a better stationary points. This aligns with the theory because we guarantee convergence to a stationary point.



(a)

(b)

Figure A·1: CIFAR-10 - α sensitivity analysis of FedDyn.

Comparison to A Full Participation Method

Recently, FedSplit (Pathak and Wainwright, 2020) is introduced to target non IID data distributions among devices. The work simplifies FL setting by considering full device participation. It characterizes FedAvg convergence and shows that FedAvg should do only one step update per device in each round to achieve global minima if device losses are different. In such cases, FedAvg becomes decentralized SGD. After pointing out this inconsistency, FedSplit is given as a potential solution.

In this work, we aim to solve FL problem with four principle characteristic which are partial participation due to unreliable communication links, massive number of devices, heterogeneous device data and unbalanced data amounts per device. Partial participation is a critical property, because, it is inconceivable that we will not be in a situation where we have all devices participating in each round. However, FedSplit does not support partial participation.

Nevertheless, we adapt FedSplit to partial participation setting with the following changes. If a device is not active in the current round, its model $z_k^{t+1} = z_k^t$ and its intermediate state $z_k^{t+\frac{1}{2}} = z_k^{t-1+\frac{1}{2}}$ are frozen. For the server model, we have two options. First option is to keep the server model as average of all device models, $x^t = \frac{1}{m}\sum_{k\in m} z_k^t$, which is named as FedSplit All. Second option is to have the server model as the average of only current round's active devices $x^t = \frac{1}{|\mathcal{P}_t|}\sum_{k\in\mathcal{P}_t} z_k^t$, which is named as FedSplit Act. In passing, we do not claim that these modifications are optimal.

For empirical evaluation, we consider CIFAR-10, 100 devices, 100% and 10% participation settings. Figure A·2.a and A·2.b show comparison between FedSplit and FedDyn for 100% and 10% participation levels respectively. FedSplit All and FedSplit Act are the same in full participation setting hence shown as one method. We observe that FedDyn performs better than FedSplit in both cases. We see that FedSplit All where the server model averages all device models is significantly underperforming than FedSplit Act where the server only averages active devices. This is due to the fact that the server model is too slow to change when all devices are averaged because most of the devices are the same across consecutive rounds. We further note that it might not be easy to get convergence theory of FedSplit in the partial participation setting.

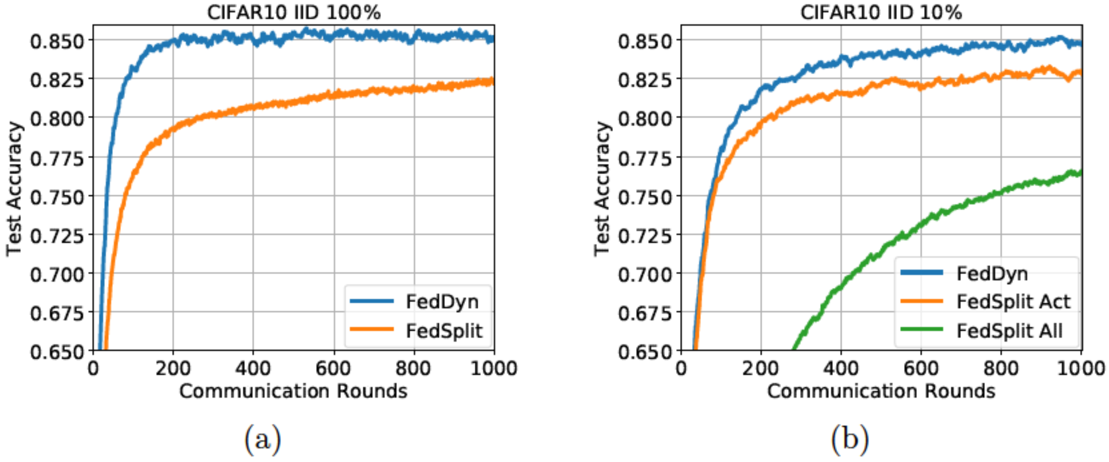


Figure A·2: CIFAR-10 - FedSplit and FedDyn comparison in full and 10% participation settings.

Table A.1: Datasets

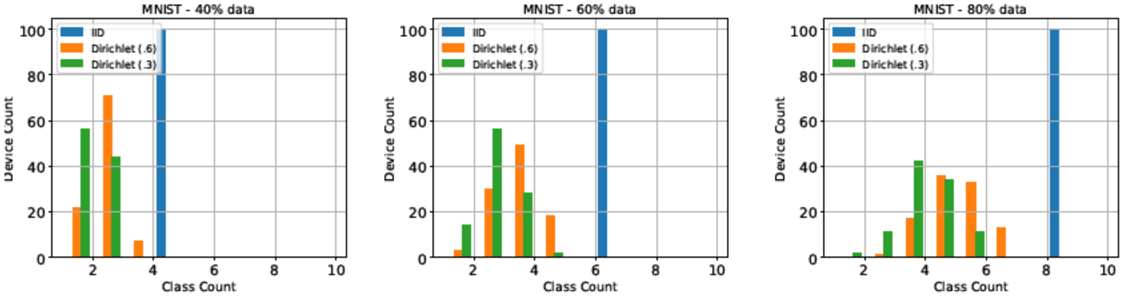
Dataset	Train Samples Amount	Test Samples Amount
CIFAR-10	50000	10000
CIFAR-100	50000	10000
MNIST	60000	10000
EMNIST-L	48000	8000
Shakespeare	200000	40000

Table A.2: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for balanced data and unbalanced data in IID and Dirichlet .3 settings with 10% participation. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Local Data	Dataset	Accuracy	FedDyn	SCAFFOLD	FedAvg	FedProx
Balanced	**IID**					
	CIFAR-10	84.5	637	1852(2.9×)	1000+(>1.6×)	1000+(>1.6×)
		82.3	240	512(2.1×)	994(4.1×)	825(3.4×)
	CIFAR-100	51.0	522	1854(3.6×)	1000+(>1.9×)	1000+(>1.9×)
		40.9	159	286(1.8×)	822(5.2×)	873(5.5×)
	Dirichlet (.3)					
	CIFAR-10	82.5	444	1880(4.2×)	1000+(>2.3×)	1000+(>2.3×)
		80.7	232	594(2.6×)	863(3.7×)	930(4.0×)
	CIFAR-100	51.0	561	1884(3.4×)	1000+(>1.8×)	1000+(>1.8×)
		42.3	170	330(1.9×)	959(5.6×)	882(5.2×)
Unbalanced	**IID**					
	CIFAR-10	84.0	335	1152(3.4×)	1000+(>3.0×)	1000+(>3.0×)
		82.3	213	548(2.6×)	834(3.9×)	834(3.9×)
	CIFAR-100	53.0	386	1656(4.3×)	1000+(>2.6×)	1000+(>2.6×)
		48.2	209	800(3.8×)	968(4.6×)	945(4.5×)
	Dirichlet (.3)					
	CIFAR-10	82.5	524	1998(3.8×)	1000+(>1.9×)	1000+(>1.9×)
		80.1	274	652(2.4×)	893(3.3×)	1000+(>3.6×)
	CIFAR-100	52.0	503	1928(3.8×)	1000+(>2.0×)	1000+(>2.0×)
		47.3	234	942(4.0×)	871(3.7×)	1000+(>4.3×)

Table A.3: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for 1% participation regime in the IID, non-IID settings. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Dataset	Accuracy	FedDyn	SCAFFOLD	FedAvg	FedProx
IID					
CIFAR-10	82.6	660	1544(2.3×)	892(1.4×)	1000+(>1.5×)
	81.6	543	1150(2.1×)	603(1.1×)	707(1.3×)
CIFAR-100	39.8	409	1982(4.8×)	428(1.0×)	512(1.3×)
	38.8	396	1862(4.7×)	392(1.0×)	454(1.1×)
MNIST	98.3	529	956(1.8×)	644(1.2×)	451(0.9×)
	97.3	145	290(2.0×)	151(1.0×)	143(1.0×)
EMNIST-L	94.9	483	1136(2.4×)	826(1.7×)	1000+(>2.1×)
	93.9	210	554(2.6×)	216(1.0×)	238(1.1×)
Shakespeare	43.0	170	460(2.7×)	188(1.1×)	151(0.9×)
	42.0	148	342(2.3×)	149(1.0×)	142(1.0×)
Dirichlet (.6)					
CIFAR-10	81.0	561	1510(2.7×)	977(1.7×)	841(1.5×)
	80.0	436	1100(2.5×)	673(1.5×)	623(1.4×)
CIFAR-100	36.6	355	1996(5.6×)	341(1.0×)	352(1.0×)
	35.6	342	1876(5.5×)	317(0.9×)	342(1.0×)
MNIST	98.2	486	1502(3.1×)	863(1.8×)	754(1.6×)
	97.2	180	332(1.8×)	199(1.1×)	166(0.9×)
EMNIST-L	94.8	405	1230(3.0×)	504(1.2×)	1000+(>2.5×)
	93.8	195	576(3.0×)	256(1.3×)	294(1.5×)
Dirichlet (.3)					
CIFAR-10	79.0	590	1580(2.7×)	955(1.6×)	738(1.3×)
	78.0	452	1272(2.8×)	653(1.4×)	497(1.1×)
CIFAR-100	36.1	343	1990(5.8×)	317(0.9×)	342(1.0×)
	35.1	321	1866(5.8×)	294(0.9×)	314(1.0×)
MNIST	98.2	521	954(1.8×)	951(1.8×)	974(1.9×)
	97.2	157	318(2.0×)	169(1.1×)	177(1.1×)
EMNIST-L	94.4	442	1860(4.2×)	481(1.1×)	1000+(>2.3×)
	93.4	241	694(2.9×)	286(1.2×)	279(1.2×)
Non-IID					
Shakespeare	43.8	158	388(2.5×)	159(1.0×)	153(1.0×)
	42.8	143	318(2.2×)	146(1.0×)	145(1.0×)

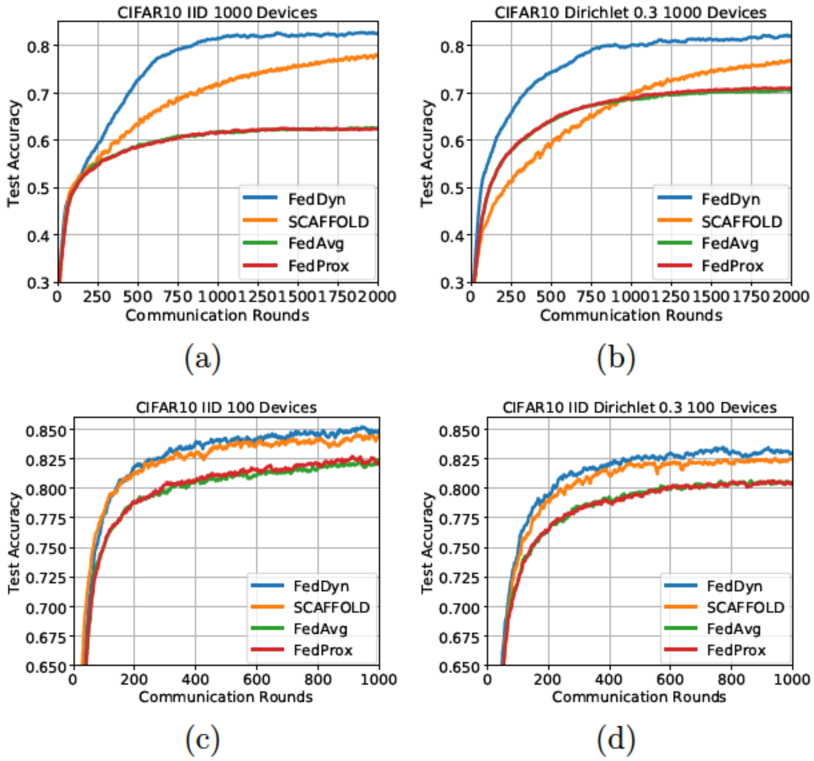


(a) (b) (c)

Figure A·3: MNIST- Histogram of device counts whose **a**: 40% , **b**: 60%, and **c**: 80% datapoints belong to k classes.

Table A.4: Number of parameters transmitted relative to one round of FedAvg to reach target test accuracy for convex synthetic problem in different types of heterogeneity settings. SCAFFOLD communicates the current model and its associated gradient per round, while others communicate only the current model. As such number of rounds for SCAFFOLD is one half of those reported.

Loss	FedDyn	SCAFFOLD	FedAvg	FedProx
Homogeneous				
0.0603	32	70(2.2×)	136(4.2×)	49(1.5×)
Type 1 Heterogeneous				
1.5717	17	88(5.2×)	20(1.2×)	18(1.1×)
Type 2 Heterogeneous				
0.1205	150	164(1.1×)	274(1.8×)	275(1.8×)
Type 3 Heterogeneous				
0.0854	34	260(7.6×)	79(2.3×)	106(3.1×)



(a)

(b)

(c)

(d)

Figure A·4: CIFAR-10- Convergence curves for different 100 and 1000 devices in the IID and Dirichlet (.3) settings with 10% participation level and balanced data.

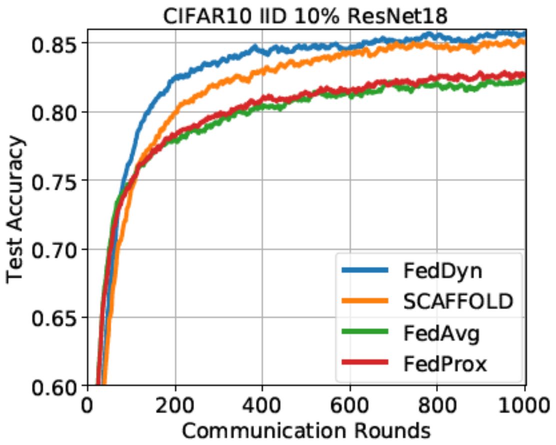
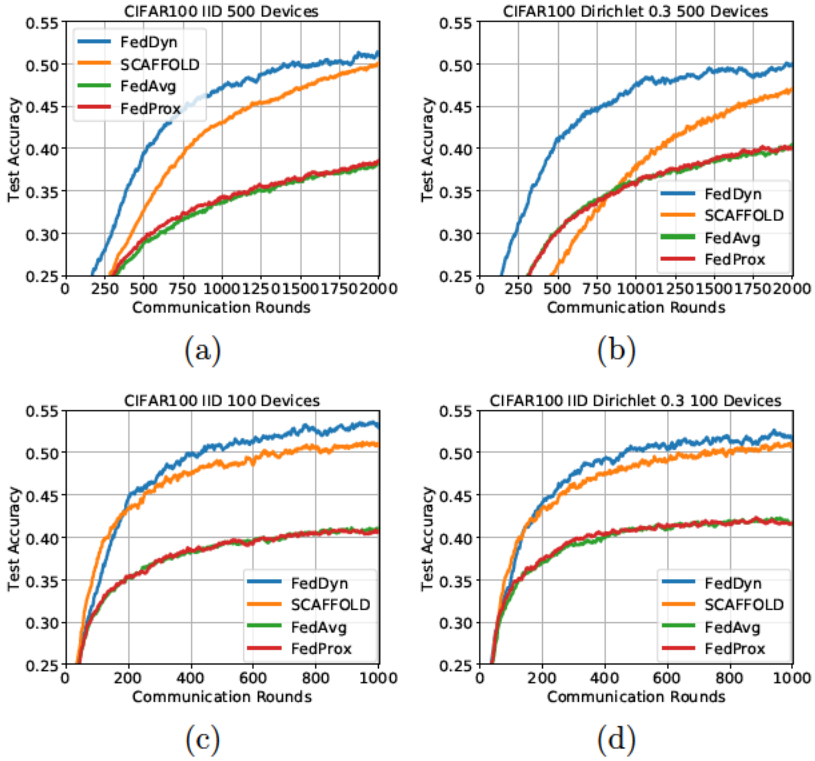


Figure A·11: Convergence curves for ResNet18 with 1000 devices and balanced data.



(a) (b)

(c) (d)

Figure A·5: CIFAR-100- Convergence curves for different 100 and 500 devices in the IID and Dirichlet (.3) settings with 10% participation level and balanced data.

A.1.2 PFL

Experiment Details

We give details omitted from the manuscript related to the experimentation for reproducibility.

Federated Datasets. We use 100 devices in our experiments. We first assign a fixed number of classes to each of the device. More specifically, device i has class list as $\{i \bmod C, (i+1) \bmod C, \ldots, (i+S-1) \bmod C\}$ where C is the total number of classes and S is the size of the class list. For instance, in CIFAR-10, ACID 5 class per device setting, device 10 has class list of $\{0, 1, 2, 3, 4\}$ whereas device 25 has class list of $\{5, 6, 7, 8, 9\}$. With this construction we guarantee that many devices such as device 10 and 25 have non overlapping classes. After fixing the class list, we distribute training and test data instances for each device based on its class list without replacement from the training and test split of the original dataset. This construction gives a training set of size 500 datapoints and a test set of size 100 datapoints for each device. Different from ACID, in ALID setting, we further permute class labels for each device. We use the same label permutation for the training and test dataset of a device.

Models. We use a convolutional network in our experiments similar to the one in (McMahan et al., 2017a; Acar et al., 2021a). Our architecture has two convolutional layers with 64 filter size and 5×5 kernels. Each convolutional layers are followed by a max pooling layer. After the second max pooling layer, we use two fully connected layers of size 384 and 192 with ReLU activation. Finally, we use a softmax layer to get predictions.

Hyperparameters. We fix the batch size as 50, the number of SGD steps as $K = 50$, the learning rate as $\beta = 0.1$ and the weight decay as 0.001 in our experiments. To avoid divergence, we set a learning rate decay across communication rounds as 0.997.

MAML adaptation has two hyperpameters. First one is the adaptation learning rate which is used to customize to the device model (η). Second hyperparameter is the number of gradient steps. This quantifies the number gradient updates to reach a device model from the meta model. We search the adaptation learning rate in range $\{0.1, 0.01\}$ and the number of gradient steps in range $\{1, 5\}$.

Different from MAML, Proto adaptation is a non parametric adaptation and it does not have extra hyperparameters.

Lastly, PFLDyn has α parameter. We search this parameters in range $\{0.1, 0.01\}$.

We run each method with the aforementioned hyperparameter search list for 100 communication rounds. Then, we pick the best performing configuration for each method and continue to run them for 1000 communication rounds.

Convergence curves. We give the convergence curves for CIFAR-10 and CIFAR-100 in Figure A·14 and A·15 respectively. We see that PFL based methods using Proto adaptation outperforms the baselines.

No personalization baselines strictly under-perform compared to personalization methods which shows a need to do personalization. We further investigate a case where no personalization methods are given a chance to personalize during inference time. We note that this does not effect training procedure. In no personalization baselines, the server model is used as the device model at each device without personalization. We consider another inference where the server model is personalized at each device using Proto or MAML adaptation. We found out that Proto adaptation gives higher performance than MAML adaptation. However, the performance is still worse than PFLDyn (Proto). We present no personalization baselines with using direct server model and Proto adaptation in inference time as well as PFLDyn (Proto) for CIFAR-10 and CIFAR-100 in Figure A·16 and A·17 respectively. Methods that perform poorly are omitted from the plots. Even though customization during inference time helps, PFLDyn (Proto) still outperforms the baselines.

The highest and the lowest level of personalization comparison. The Average level personalization metric has been reported in Table 3.1 and 3.2. We report the comparison of methods in the highest and the lowest level of personalization metrics for ACID and ALID settings in Table A.5 and A.6 respectively. Similar to Table 3.1 and 3.2, PFLDyn (Proto) method outperforms (Fallah et al., 2020b).

Implementation best practices. We give some subtle details of the implementation we think as useful practices in the following.

- No personalization baselines draw one batch of data at each round of SGD steps. Different from no personalization baselines, we draw two batches of data for P-Avg (MAML), (Fallah et al., 2020b) and PFL based methods. For MAML adaptation, first batch is used to customize the meta model into device model and the second batch is used to take gradient with respect to the meta model as in (Finn et al., 2017). For Proto adaptation, first batch is used to construct the class representations $\boldsymbol{c}_{\boldsymbol{w}}^{i,k}$ for all classes k in device i. Then, the second batch is used to calculate the loss of this representation. The first and second batches corresponds to support and query samples respectively according to (Snell et al., 2017).

- During inference time, we adapt meta model using all available training data data for each device. Namely, if MAML adaptation is used, the meta model is updated with the gradient using all training data. In case of proto adaptation, class representations are derived using all available training data. This is for reporting purposes.

- Personalized federated learning is an iterative process where a global meta model is updated over communication rounds. To increase stability of the algorithm, we perform gradient clipping at each device in each round. This stabilizes the cases where device meta models diverge.

- Gradient clipping increases stability. However, even with clipping some device meta models can diverge. This failure in one device causes the server meta model to diverge. To avoid this effect, we check each device before averaging the parameters. If a device meta model has been diverged, we do not include that device in our server model update. We found that this is a rare case but it improves the stability of the algorithms.

- We report performance of the meta model in our tables and figures. There are two options for the meta model which are average of active device meta models and average of all device meta models. We found that having average of all devices meta models give smoother curves than former one. We note that this is just for reporting purposes and we do not change the training dynamics.

Ablative Analysis of PFL

Analysis of α parameter. We test the sensitivity of α hyperparameter in CIFAR-10, 5 class per device ALID setting using Proto adaptation. By freezing other hyperparameters, we train PFLDyn (Proto) models with α varies in a logarithmic range as $\{10^{-2}, 10^{-1.5}, 10^{-1}, 10^{-0.5}, 10^{0}\}$. The highest average test accuracies obtained are $\{89.0\%, 89.5\%, 90.0\%, 89.9\%, 89.0\%\}$. The performances are close to each other as such they differ within 1% for the α range.

miniImageNet dataset (Vinyals et al., 2016). We further compare the algorithms in miniImageNet dataset. miniImageNet dataset is a subset of ImageNet ILSVRC-2012 (Deng et al., 2009). There are a total of 100 classes where each class has 600 images. The images in miniImageNet are more realistic and harder than CIFAR-100. To use miniImageNet in personalized setting, we first split dataset into training and test data points as such it becomes a dataset consists of 50000 training and 10000 test points. Then, we repeat the federated dataset generation procedure as explained in Appendix A.1.2.

We consider ALID, 5 classes per device setting with 100 deivices and 10% participation ratio. Table A.7 shows the performances of methods. As seen in the table, PFLDyn (Proto) leads to high communication savings.

A.1.3 FedHen

Algorithm 8 Algorithm Decouple

```
**Input:** $T$, $E$, $\eta$, initial models $\boldsymbol{w}_s^1, \boldsymbol{w}_c^1$,
for $t = 1 \ldots T$ do
    Randomly sample active devices, $\mathcal{Z} \subset [N]$,
    Divide $\mathcal{Z}$ into simple and complex devices, $\mathcal{Z}_s, \mathcal{Z}_c$
    # Client Optimization
    for $i \in \mathcal{Z}_s$ do
        Receive the server simple model, $\boldsymbol{w}_s^t$,
        $\boldsymbol{w}_{s,i}^{t+1} = \text{ClientTraining}\left(\boldsymbol{w}_s^t, \mathcal{D}_i, E, \eta\right)$
        Transmit $\boldsymbol{w}_{s,i}^{t+1}$ back to the server.
    end for
    for $j \in \mathcal{Z}_c$ do
        Receive the server complex model, $\boldsymbol{w}_c^t$,
        $\boldsymbol{w}_{c,j}^{t+1} = \text{ClientTraining}\left(\boldsymbol{w}_c^t, \mathcal{D}_j, E, \eta\right)$
        Transmit $\boldsymbol{w}_{c,j}^{t+1}$ back to the server.
    end for
    # Server Optimization
    Set $\boldsymbol{w}_s^{t+1}$ using weights from simple active devices,
    $\boldsymbol{w}_s^{t+1} = \frac{1}{|\mathcal{Z}_s|} \sum_{i \in \mathcal{Z}_s} \boldsymbol{w}_{s,i}^{t+1}$
    Set $\boldsymbol{w}_c^{t+1}$ using weights from complex active devices,
    $\boldsymbol{w}_c^{t+1} = \frac{1}{|\mathcal{Z}_c|} \sum_{i \in \mathcal{Z}_c} \boldsymbol{w}_{c,i}^{t+1}$
end for
```

Algorithm 9 Algorithm NoSide

```
Input: $T$, $E$, $\eta$, initial models $\boldsymbol{w}_s^1, \boldsymbol{w}_c^1$,
for $t = 1 \dots T$ do
    Randomly sample active devices, $\mathcal{Z} \subset [N]$,
    Divide $\mathcal{Z}$ into simple and complex devices, $\mathcal{Z}_s, \mathcal{Z}_c$
    # Client Optimization
    for $i \in \mathcal{Z}_s$ do
        Receive the server simple model, $\boldsymbol{w}_s^t$,
        $\boldsymbol{w}_{s,i}^{t+1} = \text{ClientTraining}\left(\boldsymbol{w}_s^t, \mathcal{D}_i, E, \eta\right)$
        Transmit $\boldsymbol{w}_{s,i}^{t+1}$ back to the server.
    end for
    for $j \in \mathcal{Z}_c$ do
        Receive the server complex model, $\boldsymbol{w}_c^t$,
        $\boldsymbol{w}_{c,j}^{t+1} = \text{ClientTraining}\left(\boldsymbol{w}_c^t, \mathcal{D}_j, E, \eta\right)$
        Transmit $\boldsymbol{w}_{c,j}^{t+1}$ back to the server.
    end for
    # Server Optimization
    Set $\boldsymbol{w}_s^{t+1}$ using weights from all active devices,
    $\boldsymbol{w}_s^{t+1} = \frac{1}{|\mathcal{Z}|}\left(\sum_{i\in\mathcal{Z}_s} \boldsymbol{w}_{s,i}^{t+1} + \sum_{j\in\mathcal{Z}_c} \left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}}\right)$
    Set $\boldsymbol{w}_c^{t+1}$'s sub-net as the updated simple model,
    $[\boldsymbol{w}_c^{t+1}]_{\mathcal{M}} = \boldsymbol{w}_s^{t+1}$
    Set rest of the $\boldsymbol{w}_c^{t+1}$ using complex active devices,
    $[\boldsymbol{w}_c^{t+1}]_{\mathcal{M}'} = \frac{1}{|\mathcal{Z}|_c}\sum_{j\in\mathcal{Z}_c} \left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}'}$
end for
```

Decouple Algorithm. We present Decouple methods in Algorithm 8. Decouple fully decouples simple and complex architecture training. We explain the method in detail below.

In each round, a random subset of devices become active, $\mathcal{Z}$. $\mathcal{Z}$ is then divided into simple active and complex active device sets as $\mathcal{Z}_s$ and $\mathcal{Z}_c$ respectively. Simple active devices receive the server simple model. A local model is trained using 'ClientTraining' method (Alg. 5). The trained model is transmitted back to the server. Complex active devices follow a similar process where they receive the server complex model. Then, a local model is trained using 'ClientTraining' method. The trained model is transmitted back to the server.

The server simple model is constructed by averaging weights from all active simple devices, $\{\boldsymbol{w}_{s,i}^{t+1}\}_{i\in\mathcal{Z}_s}$. The server complex model is constructed using the all active complex devices, $\left\{\boldsymbol{w}_{c,j}^{t+1}\right\}_{j\in\mathcal{Z}_c}$.

NoSide Algorithm. NoSide method is presented in Algorithm 9. We explain the method in detail below.

In each round, a random subset of devices become active, $\mathcal{Z}$. We divide set $\mathcal{Z}$ into simple active and complex active device sets as $\mathcal{Z}_s$ and $\mathcal{Z}_c$ respectively. Simple and complex device training is the same as in Decouple method where each active device receive the current server model based on their capacity, then train a local model using 'ClientTraining' method and transmit the trained model back to the server.

The server step is the same as in FedHeN method. The server simple model is constructed by averaging weights from all active devices, .i.e the simple devices $\{\boldsymbol{w}_{s,i}^{t+1}\}_{i\in\mathcal{Z}_s}$ as well as the common sub-net of the complex active devices $\left\{\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}}\right\}_{j\in\mathcal{Z}_c}$. The common sub-architecture of the server complex model is set equal to the constructed server simple model. The rest of the server complex model is constructed by averaging the corresponding weights of the active complex models, .i.e $\left\{\left[\boldsymbol{w}_{c,j}^{t+1}\right]_{\mathcal{M}'}\right\}_{j\in\mathcal{Z}_c}$.

Hyperparameters. We train each method for 1000 communication rounds. Each active device trains models for $E = 5$ epochs with learning rate as $\eta = 0.1$. We use SGD optimizer during training and clip gradients (at norm 10) to improve stability. If a device model fails in training, .i.e gets NaN weights, we ignore that device in server model construction only for that round. The methods are implemented using PyTorch library (Paszke et al., 2019).

Figures and Algorithms. Decouple and NoSide are summarized in Algorithm 8 and 9 respectively. The convergence curves of FedHeN, Decouple and NoSide are presented in Figure 4·1, A·18 and A·19.

Reporting Results. Methods average only active devices to set server models. This introduces noise in convergence curves and communication gain calculations. We report/present model performances when we average all devices (all complex devices for server complex model and all simple devices for server simple model). We note that this is just for the reporting purposes and the training is performed as stated in Algorithm 4, 8 & 9.

A.1.4 MOML

Experiment Details

We cover details of experiments in this section. We use PyTorch framework (Paszke et al., 2019) to train and evaluate our models. MAML meta training is implemented with Higher (Grefenstette et al., 2019) library. We explain the datasets and the baselines used for evaluations. We implement FTML baseline since there is no official implementation from (Finn et al., 2019).

Algorithm 6 stores a state as $\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t)$. Updating this state requires one full pass on the training dataset. To avoid this, we utilize the first order condition stated in Eq. 5.10. Using the first order condition, we linearly update $\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t)$ states as $\nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) = \nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t) - \alpha\left(\boldsymbol{w}^{t+1} - \boldsymbol{w}^t\right)$ in our implementation.

Performance metrics. We give details of CTM, LTM and Task Learning Efficiency Metrics.

Current Task Metric (CTM). We consider the performance with respect to the current revealed task. At each round, the meta model is adapted using the revealed limited supervised data of the current task and the performance is recorded on the test set. Since we are using the meta model trained on previous losses, this metric shows how models perform in a new task. CTM corresponds to $\frac{1}{T}\sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^t)$ performance where $\boldsymbol{w}^t$ is the meta model that is not trained on loss t.

Long-Term Task Metric (LTM). We consider the performance with respect to previous tasks. At each round, the meta model is adapted using the revealed limited supervised data of the each previous and the performance is recorded on the test set. Then we compute the mean performance of all previous tasks as the LTM. Since we are using the current meta model as a initialization of previous tasks, this metric measures the ability of catastrophic forgetting. LTM corresponds to $\frac{1}{T}\sum_{t=1}^{T} f^t \circ U^t\left(\boldsymbol{w}^{T+1}\right)$ performance.

Task Learning Efficiency Metric. At each round before updating the meta model, we record the number of data points required to achieve a sufficient performance on the current revealed task instance. For S-MNIST dataset, each task contains fixed number training data samples. We randomly sample a subset of training data of current task and adapt the meta model on this subset of data and record performance on the test data of current task. We use $[0, 10, 20, 30, 40, 50, 60]$ as the size of subset where 0 means we directly test current task on meta model. For CIFAR-100, we use $[0, 50, 100, 150, 200, 250]$ as the size of subset.

Task Learning Efficiency is recorded on top of the meta model trained on previous losses. In this metric, we allow meta models to be updated using the current task. It is a metric to measure the ability to adapt to a new task. Figure A·20 shows comparison of smoothed task efficiency results of MOML, FTML and FS on both S-MNIST and 5-way CIFAR-100 where shadow corresponds to the standard deviation of multiple runs. In general, both meta learning methods need less number of data samples to reach a threshold accuracy with more rounds. This means after revealing more tasks, both methods improve Task Learning Efficiency. We note that FS is not a meta learning methods as such its task efficiency does not improve with rounds as expected.

MOML needs to see less data as new tasks arrive. For example, in S-MNIST, MOML requires maximum 45 datapoints to achieve 80% accuracy after 300 tasks whereas FTML achieves the same point with 600 tasks. Similarly in CIFAR-100 dataset, MOML only needs 180 datapoints to achieve 55% accuracy after 110 rounds which is better than FTML.

Hyperparameters. We use SGD optimizer with learning rate of 0.1 and with weight decay of 10^{-3}. The batch size is set to 10, 20, and 50 for S-MNIST, CIFAR-100, and miniImageNet datasets respectively.

We notice that MOGD baseline improves if we do more than one gradient descent update for meta backbone. Hence, we use $K > 1$ updates in the experiments. Similar to other meta learning methods, we consider the number of gradient descent updates as an hyper parameter for MOGD method.

We search K values in range $\{10, 20\}$, $\{100, 200\}$, and $\{100, 200\}$ for S-MNIST, CIFAR-100, and miniImageNet. In MAML adaptation, we consider the number of gradient steps in $\{1, 5\}$ and the adaptation learning rates in $\{0.1, 0.01\}$. Different from these parameters, MOML has one extra parameter as α. We search α values in range $\{1, 5, 10\}$, $\{1, 0.1.0.01\}$, and $\{0.1, 0.01, 0.001\}$ for S-MNIST, CIFAR-100, and miniImageNet respectively.

Algorithm 10 Hedge

```
Input: $\gamma \in [0,1], T, \{e_i\}_{i=1}^n$experts,$\boldsymbol{p}^1 = [0,1]^n$ probability vector initialized to uniform probability,
for $t = 1, 2, \dots T$ do
    Observe new loss and adaptation function $f^t, U^t$,
    Suffer the average loss of the experts using $\boldsymbol{p}^t$ as $\sum_{i=1}^n \boldsymbol{p}_i^t f^t \circ U^t(\boldsymbol{w}_i^t)$ where $\boldsymbol{w}_i^t$ is the recent model in expert $e_i$,
    Let experts update their model based on their configurations,
    Update $\boldsymbol{p}$ vector as $\boldsymbol{p}_i^{t+1} = \frac{1}{Z}\boldsymbol{p}_i^t \gamma^{\ell_i^t}$ where $Z = \sum_{i=1}^n \boldsymbol{p}_i^t \gamma^{\ell_i^t}$ and $\ell_i^t$ is the loss of expert $i$ in the current round.
end for
```

Parameter tuning. Tuning hyper parameters is a problem in online learning

Algorithm 11 MOML $_{\nabla=\mathbf{0}}$

```
Input: $T, \boldsymbol{w}^1 = \boldsymbol{w}^1 = \mathbf{0}, \alpha, K, \beta$
for $t = 1, 2, \ldots T$ do
    Output $\boldsymbol{w}^t$, reveal $f^t$ and $U^t$, suffer $f^t \circ U^t(\boldsymbol{w}^t)$,
    $\mathcal{R}^t(\boldsymbol{w}) = \frac{\alpha}{2} \left\|\boldsymbol{w} - \boldsymbol{w}^t\right\|^2$,
    $\boldsymbol{w}_1^{t+1} = \boldsymbol{w}^{t-1}$,
    for $k = 1, 2, \ldots K$ do
        $\boldsymbol{w}_{k+1}^{t+1} = \boldsymbol{w}_k^{t+1} - \beta \left(\nabla f^t \circ U^t(\boldsymbol{w}_k^{t+1}) + \nabla \mathcal{R}^t(\boldsymbol{w}_k^{t+1})\right)$
    end for
    $\boldsymbol{w}^{t+1} = \boldsymbol{w}_{K+1}^{t+1}$,
    $\boldsymbol{w}^{t+1} = \frac{1}{2}\left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1}\right)$,
end for
```

as such we do not observe the tasks beforehand. To address this issue, we test the methods using Hedge algorithm (Freund and Schapire, 1997). For each method, we train experts using different configurations by changing the hyperparameters. Then, we consider these experts as black boxes and run Hedge method, Algorithm 10, on top of them. Hedge has one parameter γ that trade-offs the confidence on the experts. We fix it to be $\gamma = 0.1$ in our setting.

Hedge algorithm allows us to avoid parameter tuning as such we do not set a configuration in advance. However, it requires to train all experts in parallel. Our tables are based on Hedge Algorithm as such we do not pick a hyperparameter configuration instead, we report the configuration Hedge algorithm chooses in each round.

Further Ablative Analysis of MOML

Different Model Architectures. We examine the effect of the model architecture. We consider 5-way CIFAR-100 setting and test the methods using with a model that has 5 CNN layers different from the model with 2 CNN layers. Table A.8 shows the results for this architecture. As seen from Table A.8, our results indicate that MOML is superior to the competitors in this setting as well.

Non-overlapping Classes Experiment. We consider a setting where tasks have completely non-overlapping classes. We constructed tasks with 5 classes out of CIFAR-100 dataset. Since the tasks are non-overlapping, we have in total 20 tasks. Table A.9 shows the results for this non-overlapping class setting. We note that this setting is not ideal. As such meta learning needs large number of tasks whereas there are only 20 tasks. Nevertheless, MOML outperforms the competitors.

Ablation study on ∇ state. MOML introduces a regularizer $\mathcal{R}^t(\boldsymbol{w})$ (Eq. 5.6) to modify task objectives. $\mathcal{R}^t(\boldsymbol{w})$ consists of two states which are $\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t)$ and $\boldsymbol{w}^t$. According to Proposition 5.2.1, $\boldsymbol{w}$ and $\boldsymbol{w}$ converge to the same model. The linear term with $\nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t)$ state can be interpreted as an adjustment to the optimization problem. To see the effect of this term, we test a variant of MOML where this linear term is assumed to be $\mathbf{0}$ in Algorithm 11 . Table A.10 shows comparison between original MOML and the variant with eliminating the linear term (MOML $_{[\nabla=\mathbf{0}]}$) on S-MNIST and 5-way CIFAR-100 settings.

MOML is better than MOML $_{\nabla=\mathbf{0}}$ *variant.* The linear term improves the performance of MOML. For instance, in 5 way-CIFAR-100, there is an increase of 4.5% in CTM accuracy as seen in Table A.10. This improvement is consistent with our theory. As such the linear term debiases the current loss and it is essential in deriving regret guarantees of MOML.

A.1.5 DISK

Details for Illustrative Example (1D Intervals)

Dataset Overview. We generate a synthetic toy dataset with one dimensional features $x \in [0, 9]$ and binary class labels $y \in \{\text{Red}, \text{Blue}\}$. We use $\sigma(x)$ to denote the sigmoid function with a scaled by parameter $\kappa > 0$, i.e., $\sigma(x) = \frac{1}{1+\exp(-\kappa x)}$.

Function Classes. Let $\mathcal{H}$ be the 1-interval function class parametrized by two variables $\{a, b\}$, i.e., for $h \in \mathcal{H}$

$$h(x; a, b) = \sigma(x - a) - \sigma(x - b); \quad 0 < a < b < 9$$

Similarly, let $\mathcal{F}$ be the 2-interval family parametrized by four variables $\{a, b, c, d\}$, i.e., for $f \in \mathcal{F}$

$$f(x; a, b, c, d) = h(x; a, b) + h(x; c, d); \quad 0 < a < b < c < d < 9$$

Note that any function in $\mathcal{H}$ behaves as an indicator for the interval (a, b). Similarly, any function in $\mathcal{F}$ behaves as an indicator for two exclusive intervals $\{(a, b), (c, d)\}$.

Data Generation. We assume that the data is generated using the function $f^* \in \mathcal{F}$ with parameters (a^*, b^*, c^*, d^*). Dataset is sampled with balanced data from both classes. We label x as red if $f^*(x) < 0.5$, otherwise we label the point as blue. We sample 1000 i.i.d. data points as the training set and 100 data points as the test set. Figure 6·1 shows the train data. We draw an independent validation set of 100 data points for hyper-parameter tuning.

Large Capacity Teacher T belongs to the 2-interval function class $\mathcal{F}$ and is learnt with all the training data points. We learn the teacher with cross-entropy loss. We use the SGD optimizer with momentum 0.9, learning rate 0.1, weight decay 0.01, and minimize the loss for 200 epochs. Note that the teacher recovers the underlying function f^* as shown by the two intervals in Figure 6·1.

Capacity Constrained Student S belongs to the 1-interval function class $\mathcal{H}$ and it has access to all the training dataset. Note that best possible hypothesis in $\mathcal{H}$ cannot recover the performance of the function f^* and hence the student will have to settle on one of the many local minima. We show these minima as well the contour plot for the student in the Figure 6·1. We learn the student with three different loss functions (cross-entropy $\mathcal{L}_{CE}$, vanilla KD $\mathcal{L}_{KD}^{\tau,\alpha}(\mathbf{s})$, and DiSK Algorithm 7). We use similar training setup as the teacher in terms of the optimizer and training steps. For DiSK method, our guide function g has similar capacity as the student but utilizes the teacher features to learn the decision as to which points are hard-to-learn for the student. For both, KD and DiSK, we scan the α hyper-parameter over the range $\{0.0, 0.1, 0.5, 0.9, 1.0\}$. Similarly, we scan the temperature τ in the range $\{1, 2, 4\}$.

For DiSK, we scan the different hyper-parameters in the following ranges: (a) $\tau_s \in \{1, 2, 4\}$, (b) $K \in \{1, 2, 3\}$, (c) $\lambda_{\text{min}} \in \{0.01, 0.1, 1, 5, 10\}$, (d) $\lambda_T \in \{20, 50\}$, (e) Budget $\delta \in \{0.1, 0.05, 0.0\}$, and (f) $\lambda_{\text{max}} \in \{1, 5, 10, 20, 50, 100, 1000\}$. We replace the $\arg\min$ in Algorithm 7, with three gradient steps.

Note that although the hyper-parameter scan looks daunting, the default hyper-parameters: $\tau_s = \tau$ (teacher temperature), $K = 2$, $\lambda_{\text{min}} = 0.1$, $\lambda_{\text{max}} = 50$, $\delta = 0.0$ (approximate error of the global minima), $\lambda_T = 50$, work well in this setup as well as the 2D gaussian example described below.

Vanilla KD suffers from local minima. The loss landscape of the Vanilla KD contains many local minima (see Figure 6·1(b)). Since there is a big gap between student and teacher capacity, the teacher is unable to help the student discern between these bad minima. Hence, Vanilla KD leads to one of the bad local minima with high probability (see Table 6.1).

DiSK *removes bad local minima.* In contrast, DiSK deletes harder points from the landscape and as a result settles onto the global minima for S with high probability (see Figure 6·1(c) and Table 6.1 where one cluster of blue points have been removed). Note that this also removes bad minima from the S loss landscape. Finally, DiSK learns the student that has the best performance.

Details for Illustrative Example (2D Gaussians)

Dataset Overview & Data Generation. We generate a synthetic toy dataset with two dimensional features $\mathbf{x} \in \mathbb{R}^2$ and three class labels $y \in \{\text{Red}, \text{Green}, \text{Blue}\}$.

We generate six cluster centers. We assign a color to each cluster center and spread input features around these centers. We list the cluster centers along with their class labels as Red: $(0, 0)$, Blue: $(1.5, 0)$, Green: $(3, 0)$, Blue: $(0, 1.5)$, Green: $(1.5, 1.5)$, and Red: $(3, 1.5)$.

Given a cluster center $\mathbf{c}$, we draw input features $\mathbf{x}$ using a Gaussian ball with radius $r = 0.05$ around the center using multi-variate Gaussian $\mathcal{N}(\mathbf{c}, r\mathbb{I})$, where $\mathbb{I}$ is the Identity matrix.

Figure 6·2.a shows the labelled data. We sample 1000 i.i.d. data points as the training set and 1000 data points as the test set with equal representation from all three classes.

Function Classes. We use two feed-forward neural networks as function classes in this example. Let $\phi(\cdot)$ denote the Batch-Norm followed by ReLU operation.

Let $\mathcal{H}$ be the two feed-forward layer neural network. Any $h \in \mathcal{H}$ can be written as

$$h(x) = W_2\phi(W_1x)$$

where $W_1 \in \mathbb{R}^{2\times2}$ and $W_2 \in \mathbb{R}^{3\times2}$. Note that h has only two neurons and hence a very small network.

Let $\mathcal{F}$ be the three feed-forward layer neural network. Any $f \in \mathcal{F}$ can be written as

$$f(x) = \hat{W}_3\phi(\hat{W}_2\phi(\hat{W}_1x))$$

where $\hat{W}_1 \in \mathbb{R}^{8\times2}$, $\hat{W}_2 \in \mathbb{R}^{16\times8}$ and $\hat{W}_3 \in \mathbb{R}^{3\times2}$.

Note that f has 8 neurons in first and 16 neurons in the second layer. The final layer in above networks is the classifier layer that transforms the features into the class probabilities.

Large Capacity Teacher T is a 3 layer neural network with 8, 16 and 3 neurons. In between each feed-forward layer, we have batch-norm and ReLU activation non-linearity. We point out that the teacher being an over-parameterized network in this feature space, easily learns the correct decision boundary. We show this decision boundary in the Figure 6·2.a. We learn the teacher with cross-entropy loss. We use the SGD optimizer with momentum 0.9, learning rate 0.1, weight decay 0.01, and minimize the loss for 200 epochs.

Capacity Constrained Student S is a 2 layer neural network with 2 and 3 neurons. Similar to the teacher, we have batch-norm and ReLU non-linearity in between the feed-forward layers. Since the student is severely constrained as compared to the teacher, it suffers in learning the task. Different training runs lead to some popular local minima. We show the teacher solution as well as the student local minima in Figure 6·2.a. For DiSK method, our guide function g has similar capacity as the student but utilizes the teacher features to learn the decision as to which points are hard-to-learn for the student. The contour plots for the student models under KD loss and DiSK loss are shown in Figure 6·2.b-6·2.c using the visualization toolkit described in (Li et al., 2018). We following similar setup for hyper-parameter tuning as in Sec. A.1.5.

We see a similar result as in 1D example. KD suffers from bad local minima and converges to the global minima with only 43% of the initializations. Differently, DiSK escapes the local minima solutions and focus on the learnable part of the input space as shown in Figure 6·2.c. Our method converges to the global minima with very high probability (see Table 6.1).

Dataset Details

We use publicly available CIFAR-100 (Krizhevsky, 2009), Tiny-Imagenet (Le and Yang, 2015) and ImageNet-1K (Russakovsky et al., 2015) datasets. CIFAR-100 contains 50K training and 10K test images from 100 classes with size $32 \times 32 \times 3$. While Tiny-Imagenet contains 100K training and 10K test images from 200 classes with size

$64 \times 64 \times 3$. Imagenet contains 1.2M training and 100K test images from 1000 classes with size $224 \times 224 \times 3$.

For the CIFAR-100 and Tiny-Imagenet datasets, we use data augmentations including ‘RandomCrop‘, ‘RandomHorizontalFlip‘, ‘AutoAugment‘(Cubuk et al., 2019), ‘Cutout‘ (DeVries and Taylor, 2017), and ‘Mean-Std-Normalization‘. We use the same augmentation strategy in all our experiments, across baselines and different model families.

For the Imagenet-1K dataset, following the previous work (Chen et al., 2022), we use the ‘RandomCrop‘, ‘RandomHorizontalFlip‘, and ‘Mean-Std-Normalization‘.

Model Details

In this section, we list the model characteristics as well as their accuracy obtained using standard cross-entropy (CE) loss. Table A.11 lists all the models used in large capacity mismatch setting. While Table A.12 lists all the models in the small capacity mismatch setting. Below, we describe individual model for completeness.

Large Student-Teacher Capacity Mismatch All models in the Table A.11 belong to the same ResNet family and use the standard ‘BasicBlock‘ as the building block. It consists of a convolutional block, followed by four residual block stages, followed by the adaptive average pooling layer and the classifier layer. Different capacity models in this family differ only in the number of repetitions of the residual block and the number of filters in each stage. Below, we write the different of repetitions and the number of filters for the four different residual stages.

- *ResNet34* : $[64, 128, 256, 512]$ filters and repeats the ‘BasicBlock‘ $[3, 4, 6, 3]$.
- *ResNet18* : $[64, 128, 256, 512]$ filters and repeats the ‘BasicBlock‘ $[2, 2, 2, 2]$.
- *ResNet10* :$[64, 128, 256, 512]$ filters and repeats the ‘BasicBlock‘ $[1, 1, 1, 1]$.

- *ResNet10-ℓ* : $[32, 64, 128, 256]$ filters and repeats the 'BasicBlock' $[1, 1, 1, 1]$.
- *ResNet10-m* : $[16, 32, 64, 128]$ filters and repeats the 'BasicBlock' $[1, 1, 1, 1]$.
- *ResNet10-s* : $[8, 16, 32, 64]$ filters and repeats the 'BasicBlock' $[1, 1, 1, 1]$.
- *ResNet10-xs* : $[8, 16, 16, 32]$ filters and repeats the 'BasicBlock' $[1, 1, 1, 1]$.
- *ResNet10-xxs* : $[8, 8, 16, 16]$ filters and repeats the 'BasicBlock' $[1, 1, 1, 1]$.

Small Student-Teacher Capacity Mismatch Definitions of all models in the Table A.12 are borrowed from (Chen et al., 2022). We refer the reader to their official github repository (https://github.com/DefangChen/SimKD.git) for the exact definition. We trained these models on our end using the data augmentations mentioned above and found that our cross-entropy baseline as well as the vanilla KD baselines are much better than the ones reported in their work.

Guide Function Our guide function g is a three layer feed-forward network. It uses the last layer features and logits of the teacher as the input. It has 64, 128, and 1 neurons in the three layers. We include batch-norm followed by ReLU non-linearity between these layers. The final layer contains a sigmoid activation to contain the scaler output in the range $[0, 1]$.

Warm Start We note that we warm start each student model by first training them with cross entropy loss without teacher. We observe that the warm start benefits both DiSK and KD. Note that, we do not change the algorithms. We only start from a CE pre-trained student model.

Hyper-parameters

For both, KD and DiSK, we scan the α over the range $\{0.0, 0.1, 0.5, 0.9, 1.0\}$. As per recommendations from previous works(Chen et al., 2022; Cho and Hariharan, 2019; Tung and Mori, 2019), we use $\tau = 4$ as the temperature in Eq. 6.1.

For DiSK, we scan the different hyper-parameters in the following ranges: (a) $\tau_s \in \{1, 2, 4\}$, (b) $K \in \{1, 3, 5, 10, 20, 50\}$, (c) $\lambda_{\min} \in \{0.01, 0.1, 1, 5, 10\}$, (d) $\lambda_{\max} \in \{1, 5, 10, 20, 50, 100, 1000\}$, (e) Budget δ within 0.2 distance from the cross-entropy trained student model's error, and (f) $\lambda_T \in \{20, 50\}$. We replace the $\arg\min$ in the Algorithm 7, with three SGD steps over the entire dataset. For all our experiments (both KD and DiSK), we use the popular cosine learning rate scheduler for the SGD optimizer 0.1 learning rate, 0.9 momentum and $5e-4$ weight decay. We use 200 as the batch size.

Note that the default hyper-parameters: $\tau_s = \tau$ (teacher temperature), $K = 20$, $\lambda_{\min} = 0.1$, $\lambda_{\max} = 50$, $\delta =$ approximate error of the global minima (replaced by the cross-entropy error), $\lambda_T = 50$, work well in most of our experiments.

We point out that the denominator N in the budget constraint should be calibrated for the correct numerical implementation. Instead of N we use the number of wrong student predictions as the normalizer, i.e., $\sum_{i=1}^{N} y_i \neq \arg\max_y s_y(x_i)$ as this is the term that appears in the Eq. 6.2 alongside the g term in the budget constraint.

We note that we use similar parameters to train with CE in ResNet based models. Improving CE training would improve DiSK and KD as well since both are initialized with the CE trained model.

Different Experiment Setups

Scaling upto ImageNet setting. We show that DiSK scales easily to the large-scale ImageNet-1K dataset. We train two configurations with DiSK, namely (a) ResNet18 student and ResNet50 teacher, and (b) ViT-Tiny student and ViT-Large teacher. We borrow these models from the timm(Wightman, 2019) library. Table A.13 shows the DiSK performance on these two configurations along with the baseline. It clearly shows that DiSK scales well to ImageNet-1K setup and achieves significant improvements over the baselines.

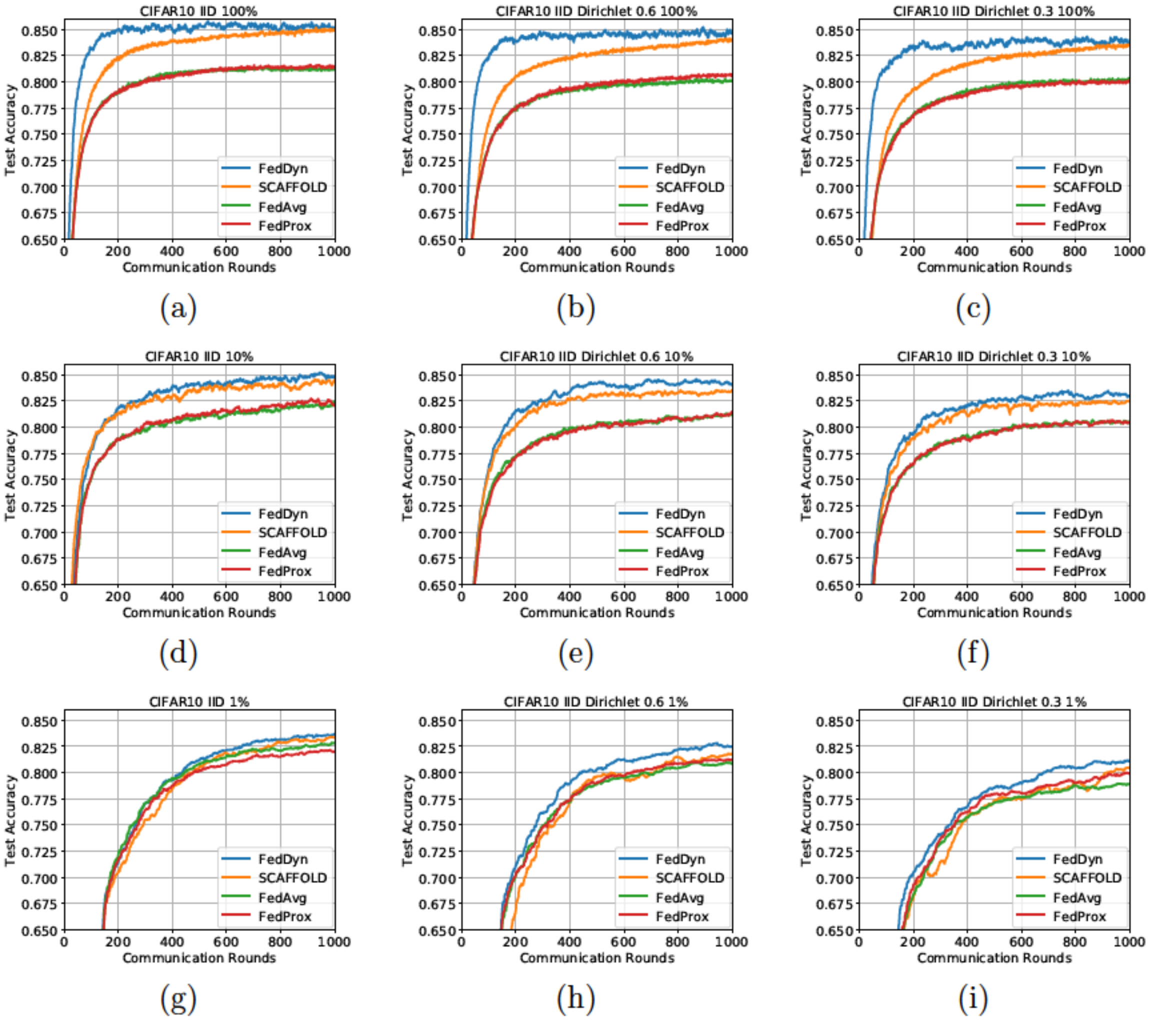
Extensions beyond Vanilla KD.

In this section, we discuss potential extension of our method beyond Vanilla KD to feature based distillation as well as self-supervised distillation.

Feature based KD methods. Let f_s and f_t denote the features for the student and teacher respectively and Ψ denote the operator such that $\Psi(f_s)$ lies in the same feature space as f_t . Commonly used feature transfer strategy is to minimize the distance between these two representations via loss function such as mean-squared error as shown below by the loss $\mathcal{L}_{ft}$.

$$\mathcal{L}_{ft} = -\frac{1}{N}\sum_{i=1}^{N}\|\Psi(f_s(\mathbf{x}_i)) - f_t(\mathbf{x}_i)\|^2; \quad \mathcal{L}_{ft-g} = -\frac{1}{N}\sum_{i=1}^{N}(1-g(\mathbf{x}_i))\|\Psi(f_s(\mathbf{x}_i)) - f_t(\mathbf{x}_i)\|^2 \tag{A.1}$$

A simple extension of this feature alignment loss to the selective distillation is shown by the loss $\mathcal{L}_{ft-g}$ that weighs each data point with the helper function decision. Table A.14 shows this feature matching extension for the SimKD(Chen et al., 2022) scheme. We leave question of finding better selective distillation losses in feature alignment to future work.

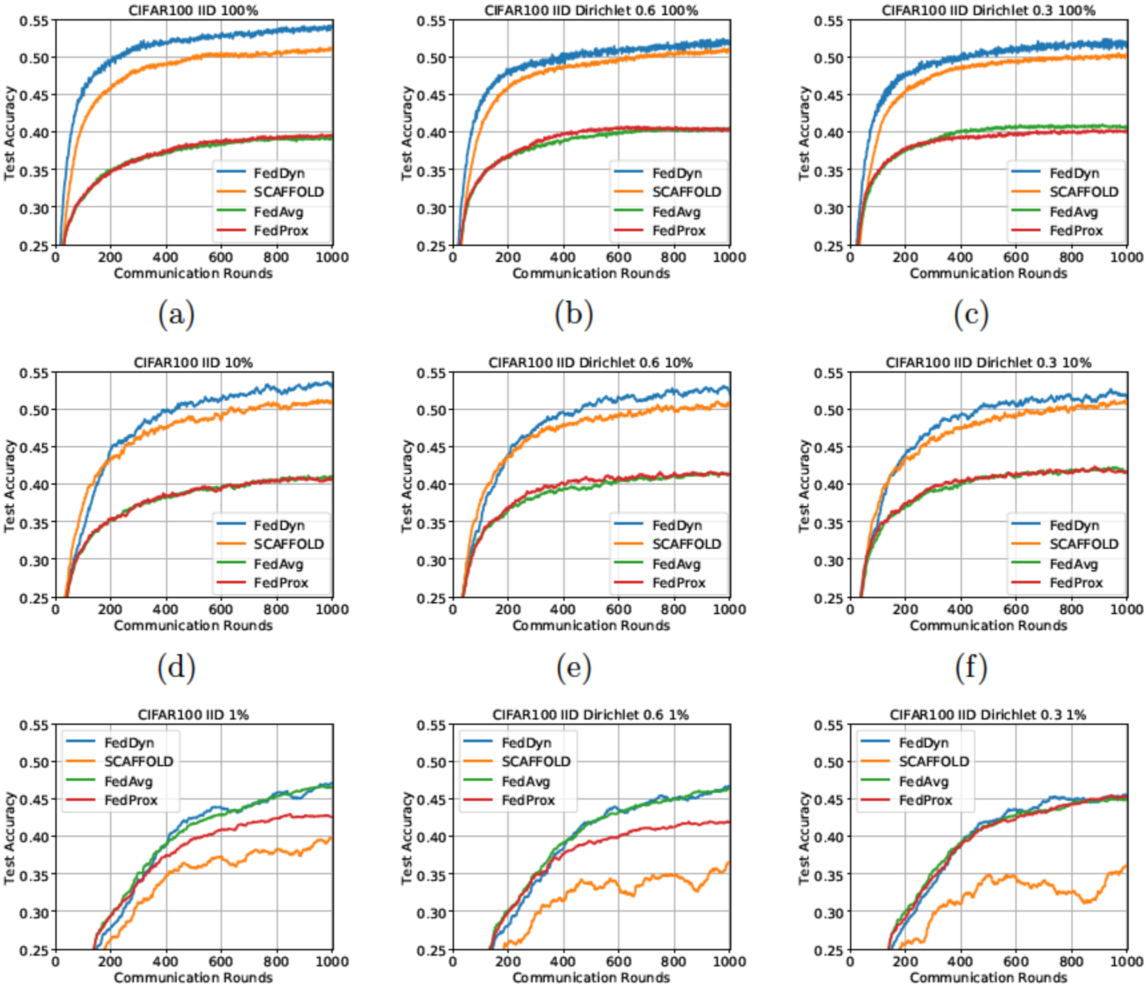


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A·6: CIFAR-10- Convergence curves for participation fractions ranging from 100% to 10% to 1% in the IID, Dirichlet (.6) and Dirichlet (.3) settings with 100 devices and balanced data.

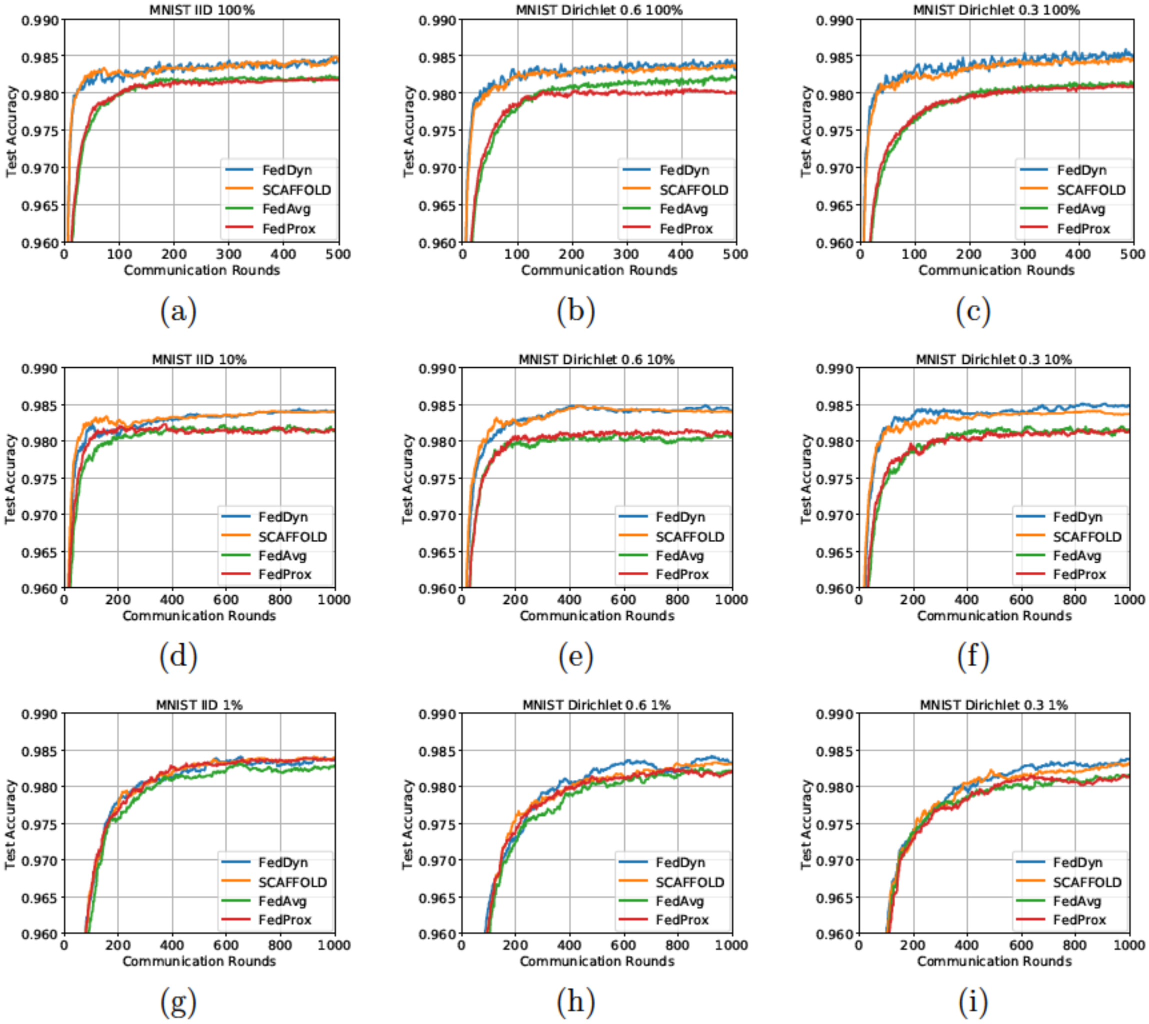


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A·7: CIFAR-100- Convergence curves for participation fractions ranging from 100% to 10% to 1% in the IID, Dirichlet (.6) and Dirichlet (.3) settings with 100 devices and balanced data.

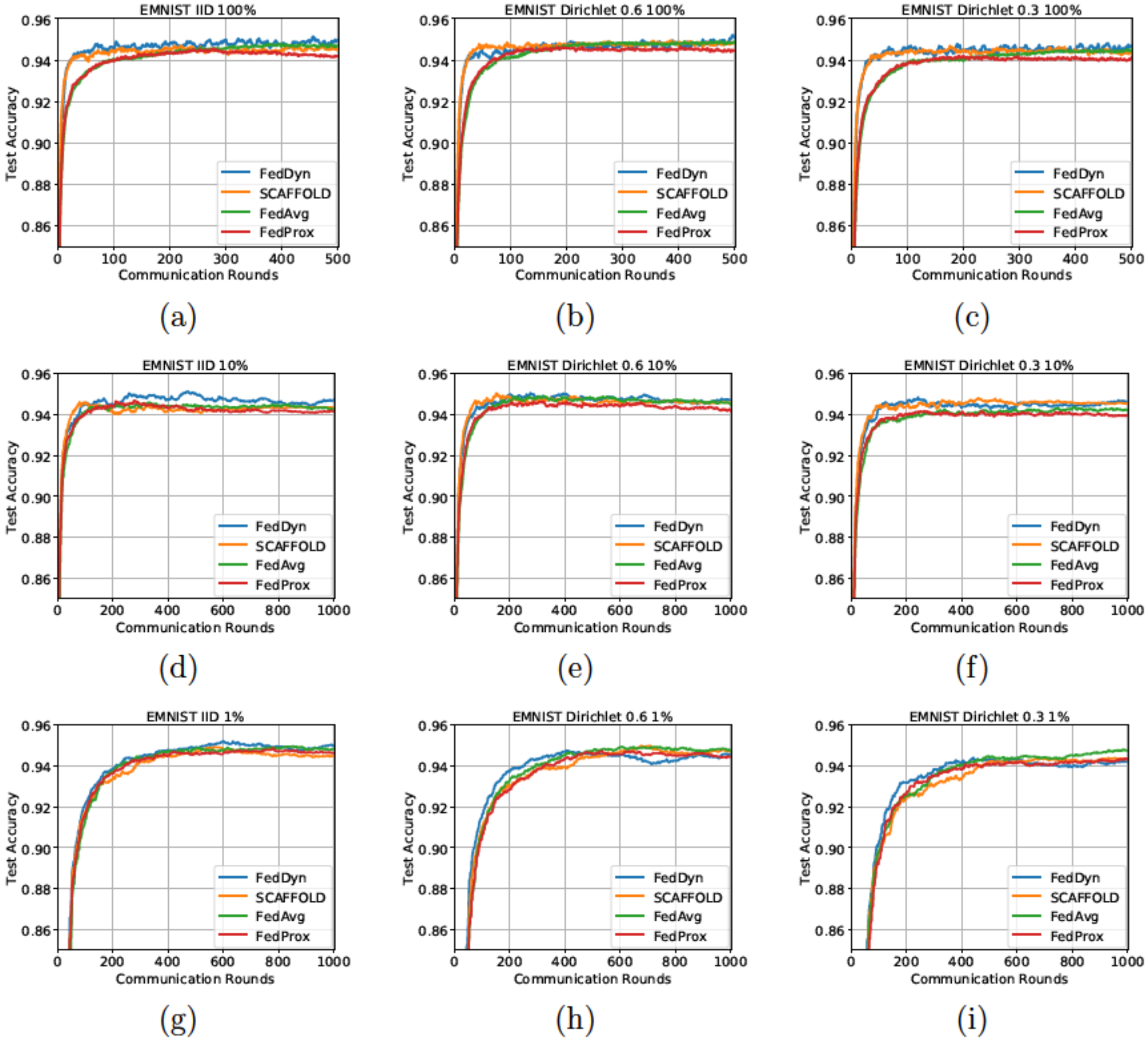


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A·8: MNIST- Convergence curves for participation fractions ranging from 100% to 10% to 1% in the IID, Dirichlet (.6) and Dirichlet (.3) settings with 100 devices and balanced data.

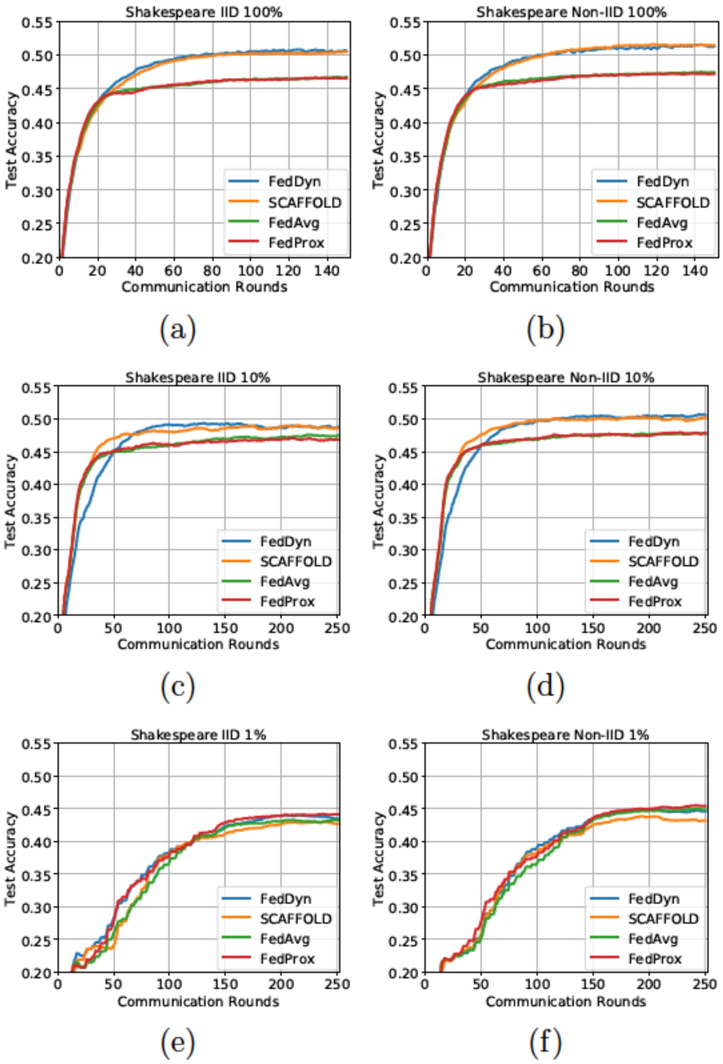


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A·9: EMNIST-L- Convergence curves for participation fractions ranging from 100% to 10% to 1% in the IID, Dirichlet (.6) and Dirichlet (.3) settings with 100 devices and balanced data.

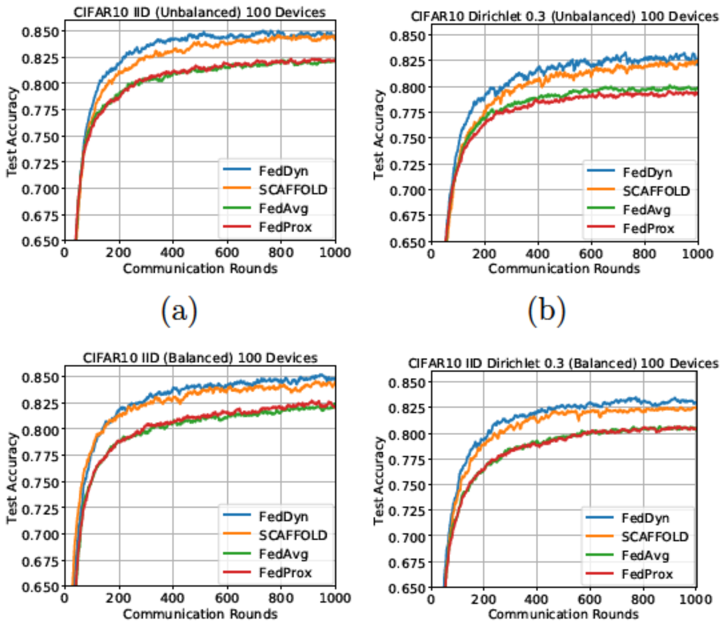


(a) (b)

(c) (d)

(e) (f)

Figure A·10: Shakespeare- Convergence curves for participation fractions ranging from 100% to 10% to 1% in the IID, and non-IID settings with 100 devices and balanced data.



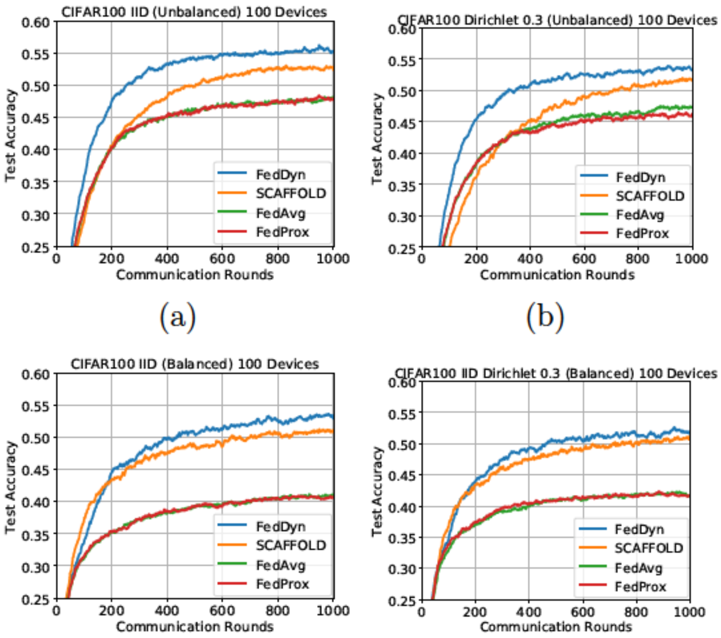
(a)

(b)

(c)

(d)

Figure A·12: CIFAR-10- Convergence curves for balanced and unbalanced data distributions with 10% participation level as well as 100 devices in the IID and Dirichlet (.3) settings.



(a)

(b)

(c)

(d)

Figure A·13: CIFAR-100- Convergence curves for balanced and unbalanced data distributions with 10% participation level as well as 100 devices in the IID and Dirichlet (.3) settings.

Table A.5: The number of model transmissions relative to one round of (Fallah et al., 2020b) required to reach the target test accuracy for the highest and the lowest level personalization performance in the Active Class Induced Diversity (ACID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of the competing method (Fallah et al., 2020b). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and gain with respect to (Fallah et al., 2020b) method is shown.

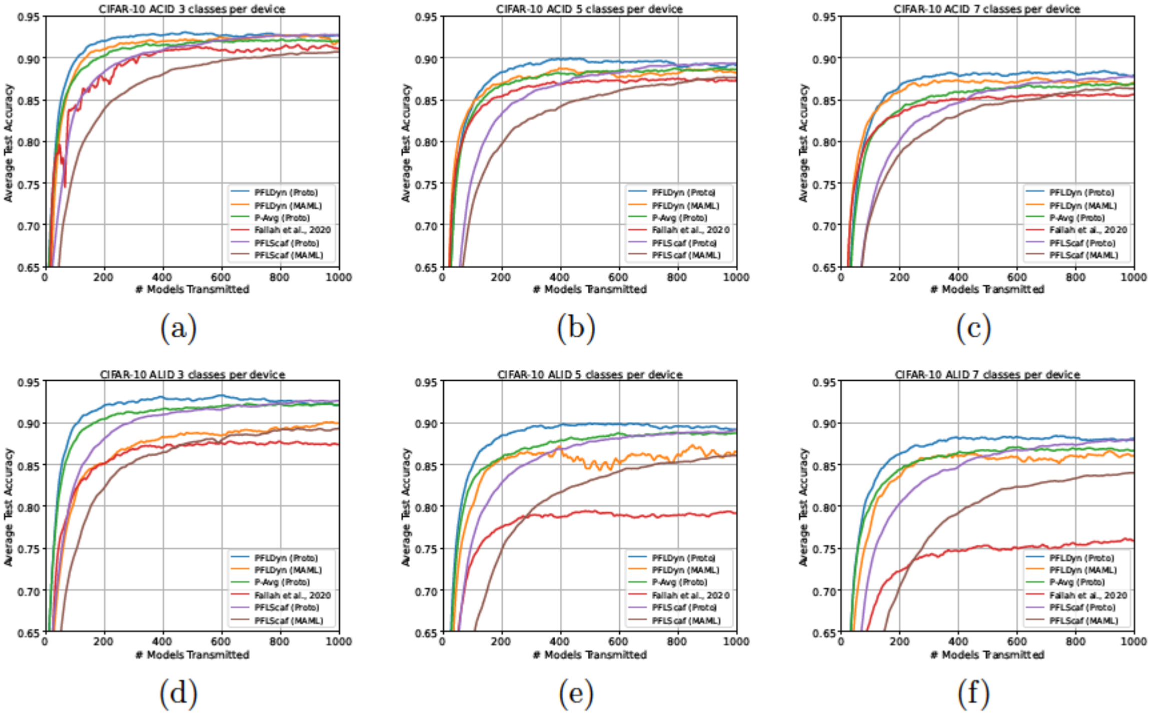
Test Performance	Dataset	Accuracy	(Fallah et al., 2020b)	PFLDyn (Proto)	PFLDyn (MAML)	PFLScaf (Proto)	PFLScaf (MAML)	P-Avg (Proto)	Gain
Highest Level Personalization	**3 Classes per Device**								
	CIFAR-10	100.0	381	**83**	111	118	874	154	**4.6×**
		99.0	106	64	86	106	186	**54**	**2.0×**
	CIFAR-100	100.0	>1000	**144**	388	>1000	>1000	539	>**7.0×**
		99.0	312	**76**	170	990	990	112	**4.1×**
	5 Classes per Device								
	CIFAR-10	99.0	297	**199**	265	680	886	352	**1.5×**
		98.0	221	**114**	199	224	518	170	**1.9×**
	CIFAR-100	99.0	>1000	463	168	**148**	>1000	294	>**6.0×**
		98.0	121	165	166	92	532	**60**	**2.0×**
	7 Classes per Device								
	CIFAR-10	96.0	358	142	**134**	422	656	170	**2.7×**
		95.0	288	**123**	128	336	646	161	**2.3×**
	CIFAR-100	98.0	363	**286**	>1000	532	>1000	397	**1.3×**
		97.0	329	285	320	370	>1000	**281**	**1.2×**
Lowest Level Personalization	**3 Classes per Device**								
	CIFAR-10	80.0	>1000	522	638	780	>1000	**483**	>**2.1×**
		79.0	512	312	**211**	474	>1000	482	**2.4×**
	CIFAR-100	75.0	>1000	**254**	949	750	750	714	>**3.9×**
		66.0	950	**127**	365	660	660	275	**7.5×**
	5 Classes per Device								
	CIFAR-10	76.0	>1000	**240**	698	982	>1000	284	>**4.2×**
		75.0	585	207	**159**	582	892	213	**3.7×**
	CIFAR-100	71.0	857	238	**150**	674	>1000	848	**5.7×**
		70.0	817	235	**148**	510	>1000	284	**5.5×**
	7 Classes per Device								
	CIFAR-10	77.0	782	**180**	306	708	>1000	487	**4.3×**
		76.0	409	**123**	305	492	742	393	**3.3×**
	CIFAR-100	73.0	>1000	**195**	287	616	>1000	825	>**5.1×**
		71.0	307	**160**	252	362	672	538	**1.9×**

Table A.6: The number of model transmissions relative to one round of (Fallah et al., 2020b) required to reach the target test accuracy for the highest and the lowest level personalization performance in the Anonymous Label Induced Diversity (ALID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of the competing method (Fallah et al., 2020b). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and gain with respect to (Fallah et al., 2020b) method is shown.

Test Performance	Dataset	Accuracy	(Fallah et al., 2020b)	PFLDyn (Proto)	PFLDyn (MAML)	PFLScaf (Proto)	PFLScaf (MAML)	P-Avg (Proto)	Gain
Highest Level Personalization	**3 Classes per Device**								
	CIFAR-10	100.0	>1000	**84**	92	126	216	173	**>11.9×**
		99.0	153	**59**	73	86	114	83	**2.6×**
	CIFAR-100	100.0	>1000	**133**	685	342	>1000	184	**>7.5×**
		97.0	134	**30**	49	62	126	44	**4.5×**
	5 Classes per Device								
	CIFAR-10	99.0	>1000	**110**	641	478	>1000	547	**>9.1×**
		96.0	123	79	99	172	360	**78**	**1.6×**
	CIFAR-100	100.0	>1000	683	**445**	>1000	>1000	628	**>2.2×**
		97.0	552	122	143	124	446	**41**	**13.5×**
	7 Classes per Device								
	CIFAR-10	98.0	>1000	**245**	475	432	>1000	350	**>4.1×**
		91.0	343	**70**	83	138	300	74	**4.9×**
	CIFAR-100	94.0	>1000	**185**	450	272	968	225	**>5.4×**
		88.0	948	**63**	144	120	478	82	**15.0×**
Lowest Level Personalization	**3 Classes per Device**								
	CIFAR-10	73.0	>1000	**114**	813	350	>1000	278	**>8.8×**
		69.0	710	**100**	250	280	586	166	**7.1×**
	CIFAR-100	73.0	>1000	**192**	721	662	>1000	692	**>5.2×**
		61.0	391	**78**	256	188	846	109	**5.0×**
	5 Classes per Device								
	CIFAR-10	72.0	>1000	**100**	245	306	988	263	**>10.0×**
		61.0	349	60	68	142	292	**55**	**6.3×**
	CIFAR-100	69.0	>1000	330	634	552	>1000	**209**	**>4.8×**
		64.0	896	258	243	300	898	**153**	**5.9×**
	7 Classes per Device								
	CIFAR-10	75.0	>1000	**177**	423	546	>1000	241	**>5.7×**
		63.0	402	**54**	74	130	276	60	**7.4×**
	CIFAR-100	65.0	>1000	165	400	206	>1000	**155**	**>6.5×**
		56.0	934	**58**	231	104	434	65	**16.1×**

Table A.7: The number of model transmissions relative to one round of (Fallah et al., 2020b) required to reach the target test accuracy for the highest, the average and the lowest level personalization performance in miniImageNet, 5 class per device Anonymous Label Induced Diversity (ALID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of the competing method (Fallah et al., 2020b). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and gain with respect to (Fallah et al., 2020b) method is shown.

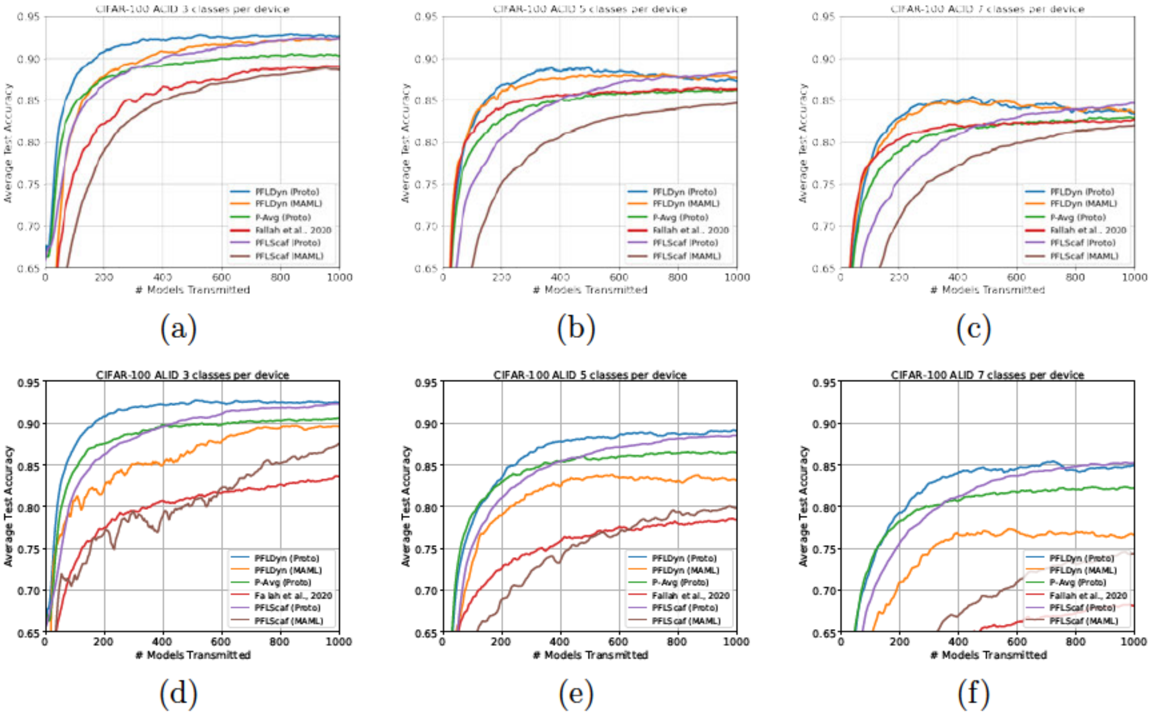
Test Performance	Accuracy	(Fallah et al., 2020b)	PFLDyn (Proto)	PFLDyn (MAML)	PFLScaf (Proto)	PFLScaf (MAML)	P-Avg (Proto)	Gain
Highest Level Personalization	94.0	> 1000	**171**	499	370	> 1000	960	**>5.8×**
	90.0	977	**73**	123	152	332	131	**13.4×**
Average Personalization	74.5	> 1000	**141**	353	338	> 1000	349	**>7.1×**
	66.2	943	**63**	137	152	386	83	**14.9×**
Lowest Level Personalization	53.0	> 1000	**140**	224	264	> 1000	487	**>7.1×**
	43.0	778	67	151	124	504	**66**	**11.7×**



(a) (b) (c)

(d) (e) (f)

Figure A·14: Smoothed convergence curves of methods in CIFAR-10 for average test accuracy among devices.



(a) (b) (c)

(d) (e) (f)

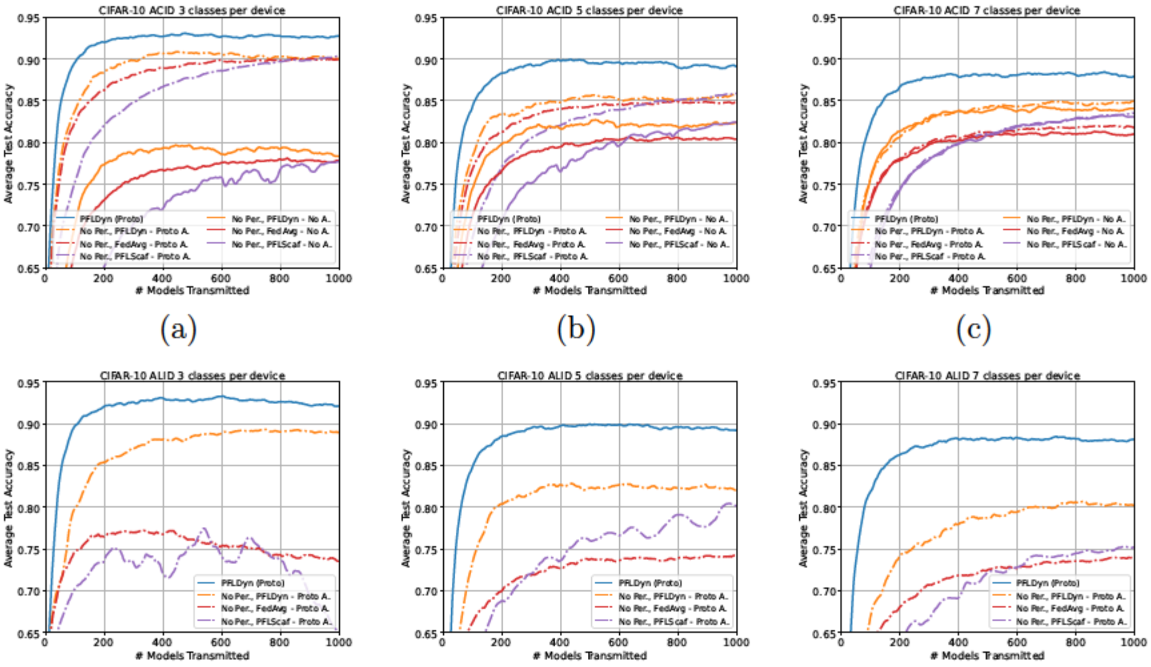
Figure A·15: Smoothed convergence curves of methods in CIFAR-100 for average test accuracy among devices.

Table A.8: MOML and FTML performances for 2 CNN layers and 5 CNN layers backbones for 5 way-CIFAR-100 settings.

		2 CNN	**5 CNN**
FTML	CTM	50.68	54.90
	LTM	54.50	62.92
MOML	CTM	55.83	56.17
	LTM	60.78	64.72

Table A.9: Non-overlapping classes performance for 5 way-CIFAR-100 settings.

	MOGD	**FTML**	**MOML**
CTM	36.89	38.38	39.32
LTM	45.82	50.90	50.54



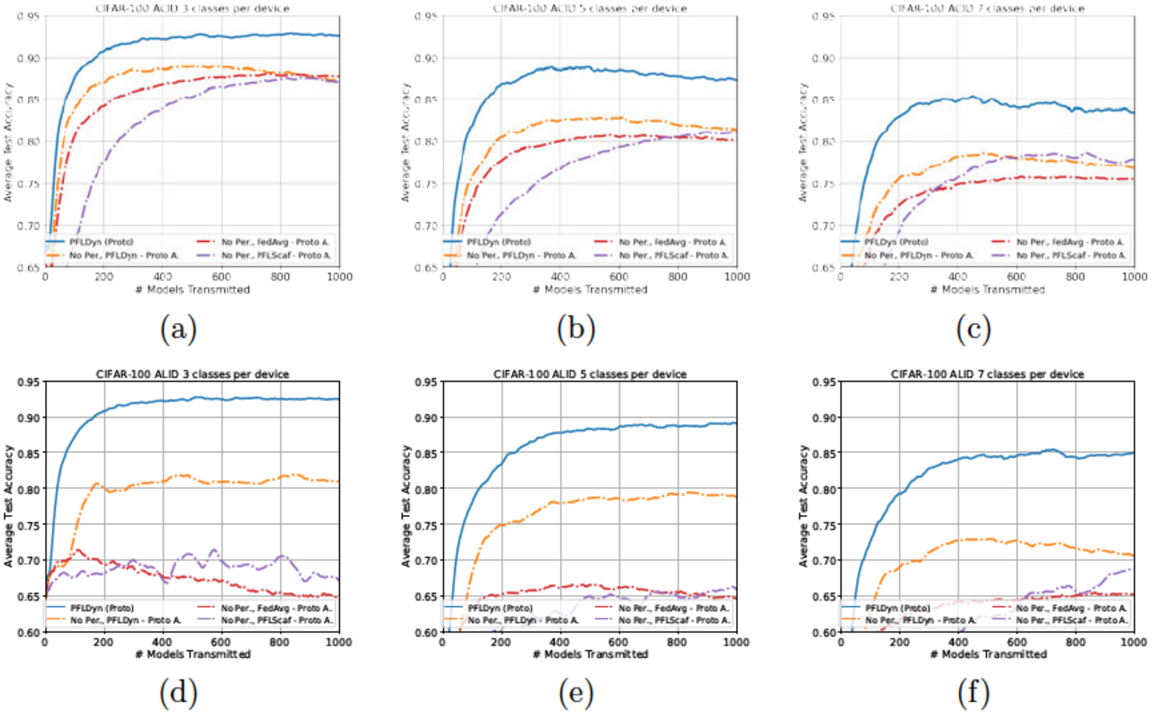
(a) (b) (c)

(d) (e) (f)

Figure A·16: Smoothed convergence curves in CIFAR-10 of PFLDyn (Proto) and no customization baselines without adaptation and with Proto adaptation in inference time.

Table A.10: Ablative study on the regularizer term of MOML.

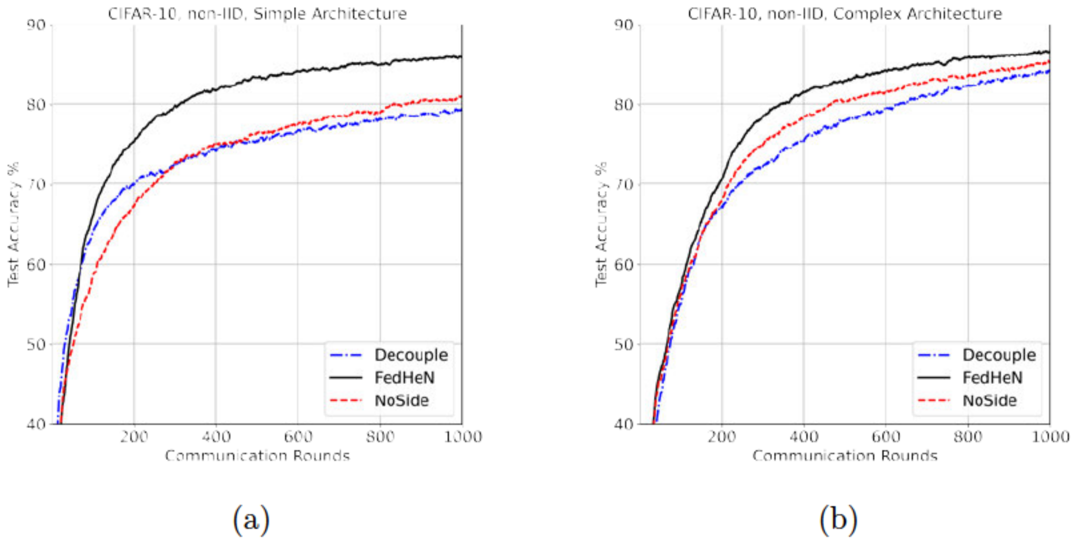
	CTM	**LTM**
S-MNIST		
MOML	85.82	87.49
MOML $_{\nabla=0}$	84.09	86.53
5 way-CIFAR-100		
MOML	55.83	60.78
MOML $_{\nabla=0}$	51.33	57.87



(a) (b) (c)

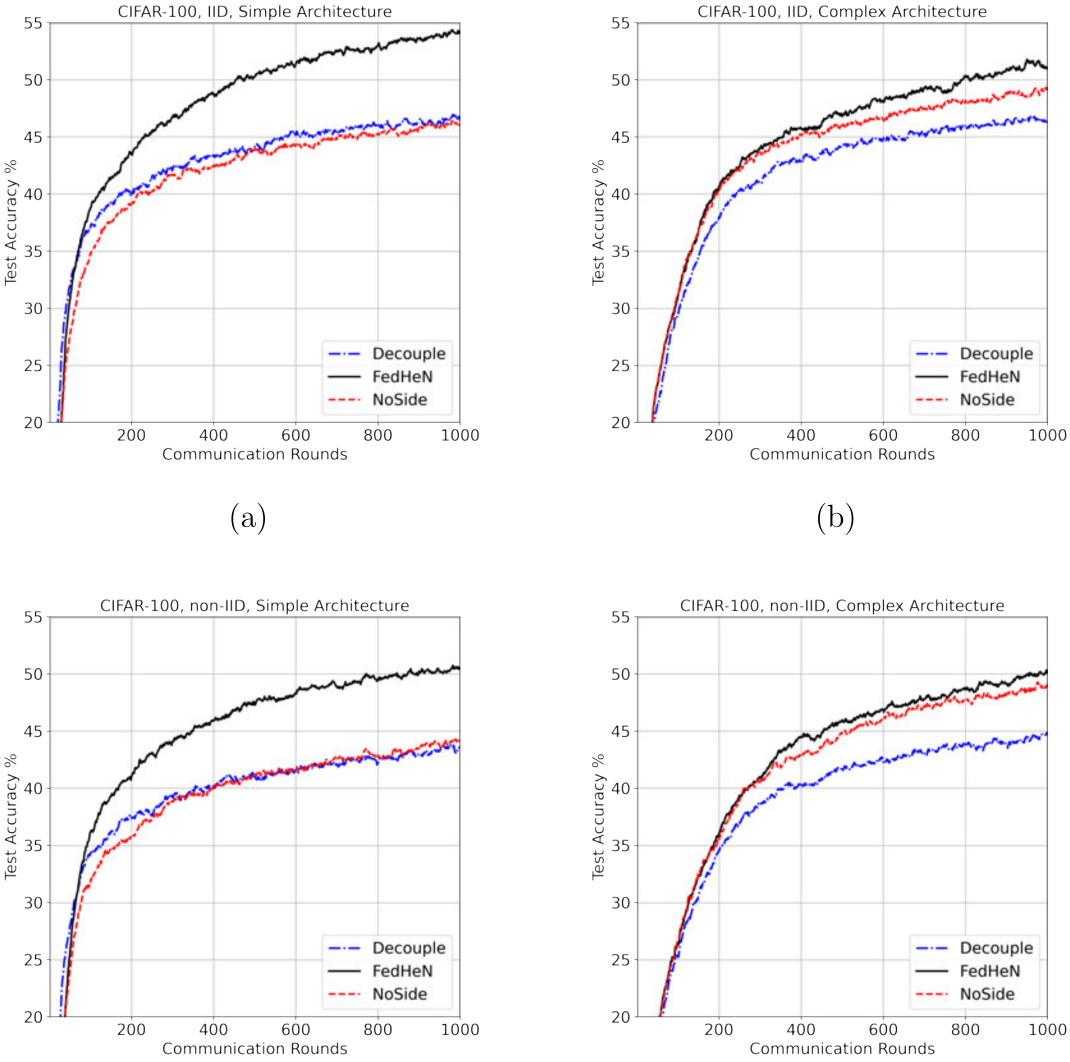
(d) (e) (f)

Figure A·17: Smoothed convergence curves in CIFAR-100 of PFLDyn (Proto) and no customization baselines without adaptation and with Proto adaptation in inference time.



(a) (b)

Figure A·18: Test accuracy vs. communication rounds on CIFAR-10 non-IID split. **a**: Simple, **b**: Complex.



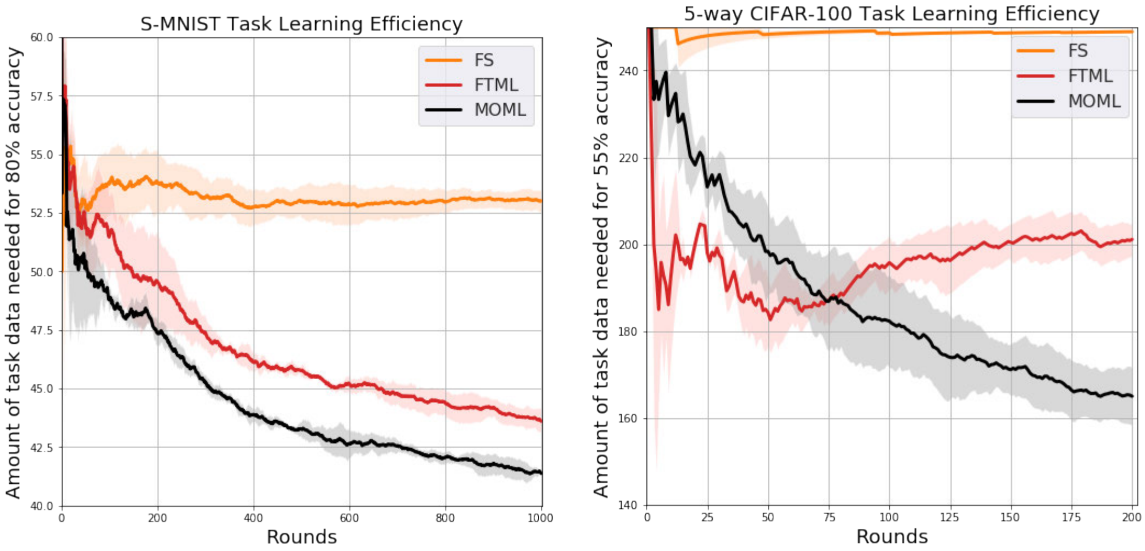
(a)

(b)

(c)

(f)

Figure A·19: Test accuracy vs. communication rounds on CIFAR-10. **a** & **b**: IID split simple and complex. **c** & **d**: non-IID split simple and complex.



(a) (b)

Figure A·20: Experiment results of Task Learning Efficiency. **(a)**: S-MNIST with efficiency threshold as 80%, **(b)**: 5-way CIFAR-100 with proficiency threshold as 55%.

Table A.11: Models used in large capacity mismatch setting along with storage and computational requirements.

Architecture		**CIFAR-100**			**Tiny-Imagenet**		
		CE Acc.	**MACs**	**Params**	**CE Acc.**	**MACs**	**Params**
Teacher	ResNet10-ℓ	71.99	64M	1.25M	52.14	255M	1.28M
	ResNet10	75.25	253M	4.92M	56.04	1013M	5M
	ResNet18	76.56	555M	11.22M	62.48	2221M	11.27M
	ResNet34	80.46	1159M	21.32M	63.06	4637M	21.38M
Student	ResNet10-xxs	32.05	2M	13K	17.44	8M	15K
	ResNet10-xs	42.99	3M	28K	25.89	12M	31K
	ResNet10-s	52.16	4M	84K	34.65	16M	90K
	ResNet10-m	65.24	16M	320K	44.74	64M	333K

Table A.12: Models used in in small capacity mismatch setting along with storage and computational requirements.

Architecture		**CIFAR-100**		
		CE Acc.	**MACs**	**Params**
Teacher	ResNet32x4	81.45	1083M	7.4M
	Wide-ResNet-40-2	78.41	327M	2.25M
Student	ResNet8x4	73.89	177M	1.2M
	ShuffleNetV2	73.74	44.5M	1.4M
	Wide-ResNet-16-2	74.29	101M	700K
	Wide-ResNet-40-1	72.81	83M	570K
	MobileNetV2x2	69.24	22M	2.4M

Table A.13: Imagenet-1K: We pick some student and teacher configurations to show that we can scale DiSK to ImageNet dataset with significant improvements in Top-1 accuracy. We borrow model definitions from timm(Wightman, 2019) repository including the convolutional and transformer vision models.

Teacher	**Student**	**CE**	**KD**	**DiSK**
ResNet50	ResNet18	69.73	71.29	72.35
ViT-Large (Patch 16, Res. 224)	ViT-Tiny (Patch 16, Res. 224)	75.45	-	77.86

Table A.14: DiSK performance against feature matching KD on CIFAR-100: Similar setup as in Table 6.4. We integrate DiSK within SimKD (Chen et al., 2022) (see Appx. A.1.5) The gains of using DiSK over KD and using SimKD + DiSK over SimKD are reported. Feature matching KD baselines are due to (Chen et al., 2022).

Architecture		**Response Matching KD**					**Feature Matching KD**				
		Accuracy (%)					**Accuracy (%)**				
Teacher	**Student**	**Teacher**	**CE**	**KD**	**DiSK**	**Gain**	**FitNet**	**SemCKD**	**SimKD**	**SimKD + DiSK**	**Gain**
Wide-ResNet-40-2	ResNet8x4	78.41	73.89	75.15	**76.05**	**0.9**	75.02	75.85	76.75	**77.13**	**0.38**
	Wide-ResNet-40-1		72.81	74.44	**75.92**	**1.48**	74.17	74.4	75.56	**76.21**	**0.65**

A.2 Proof of Theorems

A.2.1 FedDyn

Convex Analysis

Definition 1 *L_k is L smooth if*

$$\|\nabla L_k(\boldsymbol{x}) - \nabla L_k(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y}$$

Smoothness implies the following quadratic bound,

$$L_k(\boldsymbol{y}) \leq L_k(\boldsymbol{x}) + \langle \nabla L_k(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 \quad \forall \boldsymbol{x}, \boldsymbol{y} \tag{A.2}$$

If $\{L_k\}_{k=1}^m$s are convex and L smooth we have

$$\frac{1}{2Lm}\sum_{k\in[m]} \|\nabla L_k(\boldsymbol{x}) - \nabla L_k(\boldsymbol{x}_*)\|^2 \leq \ell(\boldsymbol{x}) - \ell(\boldsymbol{x}_*) \quad \forall \boldsymbol{x} \tag{A.3}$$

$$-\langle \nabla L_k(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{y} \rangle \leq -L_k(\boldsymbol{z}) + L_k(\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{x}\|^2 \quad \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \tag{A.4}$$

where $\ell(\boldsymbol{x}) = \frac{1}{m}\sum_{k=1}^m L_k(\boldsymbol{x})$ and $\nabla \ell(\boldsymbol{x}_*) = \mathbf{0}$.

We state convergence as,

Theorem 5 *For convex and L smooth $\{L_k\}_{k=1}^m$ functions and $\alpha \geq 25L$, Algorithm 1 satisfies*

$$E\left[\ell\left(\frac{1}{T}\sum_{t=0}^{T-1}\boldsymbol{g}^t\right) - \ell(\boldsymbol{w}_*)\right]$$
$$\leq \frac{1}{T}\left(10\alpha\left\|\boldsymbol{w}^0 - \boldsymbol{w}_*\right\|^2 + 100\frac{m}{P}\frac{1}{\alpha}\left(\frac{1}{m}\sum_{k\in[m]}\|\nabla L_k(\boldsymbol{w}_*)\|^2\right)\right) = O\left(\frac{1}{T}\right)$$

where $\boldsymbol{g}^t = \frac{1}{P}\sum_{k\in\mathcal{P}_t}\boldsymbol{w}_k^t$, $\boldsymbol{w}_* = arg\min_{\boldsymbol{w}} \ell(\boldsymbol{w})$.

If $\alpha = 30L\sqrt{\frac{m}{P}}$, we get the statement in Theorem 1. Throughout the proof, we utilize similar techniques as in SCAFFOLD (Karimireddy et al., 2019) convergence.

We define a set of variables which are useful in the analysis. Algorithm 1 freezes $\boldsymbol{w}_k$ and its gradients if the device is not active. Let's define virtual $\{\tilde{\boldsymbol{w}}_k^t\}$ variables as

$$\tilde{\boldsymbol{w}}_k^t = \arg\min_{\boldsymbol{w}} \; L_k(\boldsymbol{w}) - \langle \nabla L_k(\boldsymbol{w}_k^{t-1}), \boldsymbol{w}\rangle + \frac{\alpha}{2}\|\boldsymbol{w} - \boldsymbol{w}^{t-1}\|^2 \;\; \forall k \in [m], t > 0 \tag{A.5}$$

We see that $\tilde{\boldsymbol{w}}_k^t = \boldsymbol{w}_k^t$ if $k \in \mathcal{P}_t$ and $\tilde{\boldsymbol{w}}_k^t$ doesn't depend on $\mathcal{P}_t$. First order condition in Eq. A.5 and in device optimization give

$$\begin{aligned} \tilde{\boldsymbol{w}}_k^t - \boldsymbol{w}^{t-1} &= \frac{1}{\alpha}(\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t)) \;\; \forall k \in [m]; \\ \boldsymbol{w}_k^t - \boldsymbol{w}^{t-1} &= \frac{1}{\alpha}(\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_k^t)) \;\; \forall k \in \mathcal{P}_t \end{aligned} \tag{A.6}$$

$\boldsymbol{w}^t$ consists of active device average and gradient parts. Let's express active device average and its relation with the server model as,

$$\boldsymbol{g}^t = \frac{1}{P}\sum_{k\in\mathcal{P}_t} \boldsymbol{w}_k^t; \quad \boldsymbol{g}^t = \boldsymbol{w}^t + \frac{1}{\alpha}\boldsymbol{h}^t \tag{A.7}$$

Due to linear update of ∇L_k, $\boldsymbol{h}$ state in the server becomes as $\boldsymbol{h}^t = \frac{1}{m}\sum_{k\in[m]} \nabla L_k(\boldsymbol{w}_k^t)$.

Let's define some quantities that we would like to control.

$$C_t = \frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^t) - \nabla L_k(\boldsymbol{w}_*)\|^2, \quad \epsilon_t = \frac{1}{m}\sum_{k\in[m]} E\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2$$

C_t tracks how well local gradients of device models approximate the gradient of optimal model. If models converge to $\boldsymbol{w}_*$, C_t will be 0. ϵ_t keeps track of how much local models change compared to average of device models from previous round. Again, upon convergence ϵ_t will be 0.

After these definitions, Theorem 5 can be seen as a direct consequence of the following Lemma,

Lemma 1 *For convex and L smooth $\{L_k\}_{k=1}^m$ functions, if $\alpha \geq 25L$, Algorithm 1* *satisfies*

$$E\|\boldsymbol{g}^t-\boldsymbol{w}_*\|^2+\kappa C_t \le E\|\boldsymbol{g}^{t-1}-\boldsymbol{w}_*\|^2+\kappa C_{t-1}-\kappa_0 E\left[\ell(\boldsymbol{g}^{t-1})-\ell(\boldsymbol{w}_*)\right]$$

where $\kappa = 8\frac{m}{P}\frac{1}{\alpha}\frac{L+\alpha}{\alpha^2-20L^2}, \kappa_0 = 2\frac{1}{\alpha}\frac{\alpha^2-20\alpha L-40L^2}{\alpha^2-20L^2}$

Lemma 1 can be telescoped in the following way,

$$\kappa_0 E\left[\ell(\boldsymbol{g}^{t-1})-\ell(\boldsymbol{w}_*)\right] \le \left(E\|\boldsymbol{g}^{t-1}-\boldsymbol{w}_*\|^2+\kappa C_{t-1}\right)-\left(E\|\boldsymbol{g}^t-\boldsymbol{w}_*\|^2+\kappa C_t\right)$$

$$\kappa_0\sum_{t=1}^{T} E\left[\ell(\boldsymbol{g}^{t-1})-\ell(\boldsymbol{w}_*)\right] \le \left(E\|\boldsymbol{g}^{0}-\boldsymbol{w}_*\|^2+\kappa C_{0}\right)-\left(E\|\boldsymbol{g}^T-\boldsymbol{w}_*\|^2+\kappa C_T\right)$$

If $\alpha \ge 25L$, κ_0 and κ become positive. By definition, we also have C_t sequences as positive. Eliminating negative terms on RHS gives,

$$\kappa_0\sum_{t=1}^{T} E\left[\ell(\boldsymbol{g}^{t-1})-\ell(\boldsymbol{w}_*)\right] \le E\|\boldsymbol{g}^{0}-\boldsymbol{w}_*\|^2+\kappa C_{0}$$

Applying Jensen on LHS gives,

$$E\left[\ell\left(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{g}^{t-1}\right)-\ell(\boldsymbol{w}_*)\right] \le \frac{1}{T}\frac{1}{\kappa_0}\left(\|\boldsymbol{g}^{0}-\boldsymbol{w}_*\|^2+\kappa C_0\right)=O\left(\frac{1}{T}\right)$$

which proves the statement in Theorem 5.

Similar to fundamental gradient descent analysis, $\|\boldsymbol{g}^t-\boldsymbol{w}_*\|^2$ is expressed as $\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}+\boldsymbol{g}^{t-1}-\boldsymbol{w}_*\|^2$ and expanded in the proof of Lemma 1. The resulting expression has $(\boldsymbol{g}^t-\boldsymbol{g}^{t-1})$ and $\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\|^2$ terms. To tackle these extra terms, we state the following Lemmas and prove long ones at the end.

Lemma 2 *Algorithm 1 satisfies*

$$E\left[\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\right]=\frac{1}{\alpha m}\sum_{k\in[m]}E\left[-\nabla L_k(\tilde{\boldsymbol{w}}_k^t)\right]$$

Proof.

$$\begin{aligned}
E\left[\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\right] &= E\left[\left(\frac{1}{P}\sum_{k\in\mathcal{P}_t}\boldsymbol{w}_k^t\right) - \boldsymbol{w}^{t-1} - \frac{1}{\alpha}\boldsymbol{h}^{t-1}\right] \\
&= E\left[\frac{1}{P}\sum_{k\in\mathcal{P}_t}\left(\boldsymbol{w}_k^t - \boldsymbol{w}^{t-1} - \frac{1}{\alpha}\boldsymbol{h}^{t-1}\right)\right] \\
&= E\left[\frac{1}{\alpha P}\sum_{k\in\mathcal{P}_t}\left(\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_k^t) - \boldsymbol{h}^{t-1}\right)\right] \\
&= E\left[\frac{1}{\alpha P}\sum_{k\in\mathcal{P}_t}\left(\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \boldsymbol{h}^{t-1}\right)\right] \\
&= E\left[\frac{1}{\alpha m}\sum_{k\in[m]}\left(\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \boldsymbol{h}^{t-1}\right)\right] \\
&= \frac{1}{\alpha m}\sum_{k\in[m]} E\left[-\nabla L_k(\tilde{\boldsymbol{w}}_k^t)\right]
\end{aligned}$$

where first equation is from definition in Eq. A.7. The following equations come from Eq. A.6 and $\tilde{\boldsymbol{w}}_k^t = \boldsymbol{w}_k^t$ if $k \in \mathcal{P}_t$ respectively. Fifth equation is due to taking expectation while conditioning on randomness before time t. If conditioned on randomness prior to t, every variable except $\mathcal{P}_t$ is revealed and each device is selected with probability $\frac{P}{m}$. Last one is due to definition of $\boldsymbol{h}^t = \frac{1}{m}\sum_{k\in[m]}\nabla L_k(\boldsymbol{w}_k^t)$. □

Similarly, $\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2$ is bounded with the following,

Lemma 3 *Algorithm 1 satisfies*

$$E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \leq \epsilon_t$$

Proof.

$$\begin{aligned}
E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 =& E\left\|\frac{1}{P}\sum_{k\in\mathcal{P}_t}\left(\boldsymbol{w}_k^t - \boldsymbol{g}^{t-1}\right)\right\|^2 \leq \frac{1}{P}E\left[\sum_{k\in\mathcal{P}_t}\left\|\boldsymbol{w}_k^t - \boldsymbol{g}^{t-1}\right\|^2\right] \\
&= \frac{1}{P}E\left[\sum_{k\in\mathcal{P}_t}\left\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\right\|^2\right] = \frac{1}{P}\frac{P}{m}\sum_{k\in[m]}E\left\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\right\|^2 = \epsilon_t
\end{aligned}$$

where first equality comes from Eq. A.7. The following inequality applies Jensen. Remaining relations are due to $\tilde{\boldsymbol{w}}_k^t = \boldsymbol{w}_k^t$ if $k \in \mathcal{P}_t$, taking expectation by conditioning on randomness before time t and definition of ϵ_t.□

We need to further bound excess ϵ_t term arising in Lemma 3. We introduce two more Lemmas to handle this term.

Lemma 4 *For convex and L smooth $\{L_k\}_{k=1}^m$ functions, Algorithm 1 satisfies*

$$\left(1 - 4L^2\frac{1}{\alpha^2}\right)\epsilon_t \le 8\frac{1}{\alpha^2}C_{t-1} + 8L\frac{1}{\alpha^2}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]$$

Lemma 5 *For convex and L smooth $\{L_k\}_{k=1}^m$ functions, Algorithm 1 satisfies*

$$C_t \le \left(1 - \frac{P}{m}\right)C_{t-1} + 2L^2\frac{P}{m}\epsilon_t + 4L\frac{P}{m}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]$$

$E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]$ terms constitute LHS of the telescopic sum. Let's express $\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2$ term as,

$$\begin{aligned}
E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 &= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_* + \boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + 2E\left[\left\langle \boldsymbol{g}^{t-1} - \boldsymbol{w}_*, \boldsymbol{g}^t - \boldsymbol{g}^{t-1}\right\rangle\right] + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + \frac{2}{\alpha m}\sum_{k\in[m]} E\left[\left\langle \boldsymbol{g}^{t-1} - \boldsymbol{w}_*, -\nabla L_k(\tilde{\boldsymbol{w}}_k^t)\right\rangle\right] + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&\le E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + \frac{2}{\alpha m}\sum_{k\in[m]} E\left[L_k(\boldsymbol{w}_*) - L_k(\boldsymbol{g}^{t-1}) + \frac{L}{2}\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2\right] \\
&\quad + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 - \frac{2}{\alpha}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] + \frac{L}{\alpha}\epsilon_t + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \qquad \text{(A.8)}
\end{aligned}$$

where we first expand the square term and use Lemma 2. Following inequality is due to Eq. A.4.

Let's scale Lemma 4 and 5 with $\alpha\frac{L+\alpha}{\alpha^2-20L^2}$ and $8\frac{m}{P}\frac{1}{\alpha}\frac{L+\alpha}{\alpha^2-20L^2}$ respectively. We note that the coefficients are positive due to the condition on α. Summing Eq. A.8, Lemma

3, scaled versions of Lemma 5 and 4 gives the statement in Lemma 1. □

We give the omitted proofs here.

Lemma 6 $\forall \{\boldsymbol{v}_j\}_{j=1}^n \in \mathcal{R}^d$, *triangular inequality satisfies*

$$\left\| \sum_{j=1}^n \boldsymbol{v}_j \right\|^2 \leq n \sum_{j=1}^n \|\boldsymbol{v}_j\|^2$$

Proof.

Using Jensen we get, $\left\| \frac{1}{n} \sum_{j=1}^n \boldsymbol{v}_j \right\|^2 \leq \frac{1}{n} \sum_{j=1}^n \|\boldsymbol{v}_j\|^2$. Multiplying both sides with n^2 gives the inequality. □

Lemma 7 *Algorithm 1 satisfies*

$$E \left\| \boldsymbol{h}^t \right\|^2 \leq C_t$$

Proof.

$$\begin{aligned} E \left\| \boldsymbol{h}^t \right\|^2 =& E \left\| \frac{1}{m} \sum_{k \in [m]} \nabla L_k(\boldsymbol{w}_k^t) \right\|^2 = E \left\| \frac{1}{m} \sum_{k \in [m]} \left(\nabla L_k(\boldsymbol{w}_k^t) - \nabla L_k(\boldsymbol{w}_*) \right) \right\|^2 \\ \leq& \frac{1}{m} \sum_{k \in [m]} E \left\| \nabla L_k(\boldsymbol{w}_k^t) - \nabla L_k(\boldsymbol{w}_*) \right\|^2 = C_t \end{aligned}$$

First equality is due to server update rule of $\boldsymbol{h}$ vector; second adds ($\nabla \ell(\boldsymbol{w}_*) = 0$); third applies Jensen Inq.; and last one is the definition of C_t. □

Proof of Lemma 4

$$\begin{aligned} \epsilon_t =& \frac{1}{m} \sum_{k \in [m]} E\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2 = \frac{1}{m} \sum_{k \in [m]} E \left\| \tilde{\boldsymbol{w}}_k^t - \boldsymbol{w}^{t-1} - \frac{1}{\alpha} \boldsymbol{h}^{t-1} \right\|^2 \\ =& \frac{1}{\alpha^2} \frac{1}{m} \sum_{k \in [m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \boldsymbol{h}^{t-1}\|^2 \\ =& \frac{1}{\alpha^2} \frac{1}{m} \sum_{k \in [m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_*) + \nabla L_k(\boldsymbol{w}_*) - \nabla L_k(\boldsymbol{g}^{t-1}) \\ & + \nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \boldsymbol{h}^{t-1}\|^2 \end{aligned}$$

$$\begin{aligned}
\leq & \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
& + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\tilde{\boldsymbol{w}}_k^{t}) - \nabla L_k(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}E\|\boldsymbol{h}^{t-1}\|^2 \\
\leq & \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
& + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\tilde{\boldsymbol{w}}_k^{t}) - \nabla L_k(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}C_{t-1} \\
\leq & \frac{8}{\alpha^2}C_{t-1} + \frac{4L^2}{\alpha^2}\epsilon_t + \frac{8L}{\alpha^2}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]
\end{aligned}$$

where first and second come from Eq. A.7 and A.6. Following inequalities come from Lemma 6, 7, smoothness and Eq. A.3. Rearranging terms gives the Lemma.□

Proof of Lemma 5

$$\begin{aligned}
C_t = & \frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
= & \left(1 - \frac{P}{m}\right)\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
& + \frac{P}{m}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\tilde{\boldsymbol{w}}_k^{t}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
= & \left(1 - \frac{P}{m}\right)C_{t-1} \\
& + \frac{P}{m}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\tilde{\boldsymbol{w}}_k^{t}) - \nabla L_k(\boldsymbol{g}^{t-1}) + \nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
\leq & \left(1 - \frac{P}{m}\right)C_{t-1} + \frac{2P}{m}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\tilde{\boldsymbol{w}}_k^{t}) - \nabla L_k(\boldsymbol{g}^{t-1})\|^2 \\
& + \frac{2P}{m}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
\leq & \left(1 - \frac{P}{m}\right)C_{t-1} + \frac{2L^2P}{m}\epsilon_t + \frac{2P}{m}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_*)\|^2 \\
\leq & \left(1 - \frac{P}{m}\right)C_{t-1} + \frac{2L^2P}{m}\epsilon_t + \frac{4LP}{m}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]
\end{aligned}$$

where first equality comes from taking expectation with respect to $\mathcal{P}_t$; second equality comes from definition of C_t. Inequalities follow from Lemma 6, smoothness and Eq. A.3 respectively.□

Strongly Convex Analysis

We state convergence for μ strongly convex and L smooth $\{L_k\}_{k=1}^m$ functions as,

Theorem 6 *For μ strongly convex and L smooth $\{L_k\}_{k=1}^m$ functions and $\alpha \geq \max\left(5\frac{m}{P}\mu, 30L\right)$, Algorithm 1 satisfies*

$$E\left[\ell\left(\frac{1}{R}\sum_{t=0}^{T-1} r^t \boldsymbol{g}^t\right) - \ell(\boldsymbol{w}_*)\right] \\ \leq \frac{1}{r^{T-1}}\left(20\alpha\left\|\boldsymbol{w}^0 - \boldsymbol{w}_*\right\|^2 + 400\frac{m}{P}\frac{1}{\alpha}\left(\frac{1}{m}\sum_{k\in[m]}\|\nabla L_k(\boldsymbol{w}_*)\|^2\right)\right)$$

where $\boldsymbol{g}^t = \frac{1}{P}\sum_{k\in\mathcal{P}_t}\boldsymbol{w}_k^t$, $r = \left(1+\frac{\mu}{\alpha}\right)$, $R = \sum_{t=0}^{T-1} r^t$, $\boldsymbol{w}_* = arg\min_{\boldsymbol{w}} \ell(\boldsymbol{w})$.

If $\alpha = \max\left(5\frac{m}{P}\mu, 30L\right)$ we get the statement in Theorem 1. We will use the same $\{\tilde{\boldsymbol{w}}_k^t\}$, $\boldsymbol{g}^t$, C_t, ϵ_t variables defined in Eq. A.5, A.6, A.7.

With these definitions in mind, Theorem 6 can be seen as a direct consequence of the following Lemma,

Lemma 8 *For μ strongly convex and L smooth $\{L_k\}_{k=1}^m$ functions, if $\alpha \geq \max\left(5\frac{m}{P}\mu, 30L\right)$, Algorithm 1 satisfies*

$$r\left(E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 + \kappa C_t\right) \leq E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + \kappa C_{t-1} - \kappa_0 E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right]$$

where $\kappa = \frac{8m(L+\alpha)}{z}$,
$\kappa_0 = \frac{2\alpha^3P + 2\alpha^2P\mu - 2\alpha^2m\mu - 40\alpha^2LP - 80\alpha L^2P - 40\alpha LP\mu + 8\alpha Lm\mu + 16L^2m\mu - 80L^2P\mu}{\alpha z}$,
$z = \alpha^3P + \alpha^2P\mu - \alpha^2m\mu - 20\alpha L^2P + 4L^2m\mu - 20L^2P\mu, r = \left(1+\frac{\mu}{\alpha}\right).$

Let's multiply Lemma 8 with r^{t-1} and telescope as,

$$\begin{aligned}
\kappa_0 r^{t-1} E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] \leq & r^{t-1}\left(E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + \kappa C_{t-1}\right) \\
& - r^t\left(E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 + \kappa C_t\right) \\
\kappa_0 \sum_{t=1}^{T} r^{t-1} E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] \leq & \left(E\|\boldsymbol{g}^0 - \boldsymbol{w}_*\|^2 + \kappa C_0\right) \\
& - r^T\left(E\|\boldsymbol{g}^T - \boldsymbol{w}_*\|^2 + \kappa C_T\right)
\end{aligned}$$

If $\alpha \geq \max\left(5\frac{m}{P}\mu, 30L\right)$, κ_0 and κ become positive. Dividing both sides with $R = \sum_{t=0}^{T-1} r^t$ and eliminating negative terms on RHS gives,

$$\kappa_0 \frac{1}{R} \sum_{t=1}^{T} r^{t-1} E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] \leq \frac{1}{R}\left(E\|\boldsymbol{g}^0 - \boldsymbol{w}_*\|^2 + \kappa C_0\right)$$

Applying Jensen on LHS gives,

$$E\left[\ell\left(\frac{1}{R}\sum_{t=1}^{T} r^{t-1}\boldsymbol{g}^{t-1}\right) - \ell(\boldsymbol{w}_*)\right] \leq \frac{1}{R}\frac{1}{\kappa_0}\left(\|\boldsymbol{g}^0 - \boldsymbol{w}_*\|^2 + \kappa C_0\right)$$

We have $\frac{1}{R} = \frac{r-1}{r^T - 1} \leq \frac{1}{r^{T-1}}$. Combining two inequalities, we get,

$$E\left[\ell\left(\frac{1}{R}\sum_{t=1}^{T} r^{t-1}\boldsymbol{g}^{t-1}\right) - \ell(\boldsymbol{w}_*)\right] \leq \frac{1}{r^{T-1}}\frac{1}{\kappa_0}\left(\|\boldsymbol{g}^0 - \boldsymbol{w}_*\|^2 + \kappa C_0\right)$$

which proves the statement in Theorem 6.

The proof of Lemma 8 is similar to the convex analysis. We generalize Eq. A.4 to strongly convex functions for$\{L_k\}_{k=1}^m$s are μ strongly convex and L smooth as,

$$-\langle \nabla L_k(\boldsymbol{x}), \boldsymbol{z} - \boldsymbol{y}\rangle \leq -L_k(\boldsymbol{z}) + L_k(\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{x}\|^2 - \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \quad \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \quad \text{(A.9)}$$

Since strongly convex functions are convex functions and we only change Eq. A.4, we can directly use Lemma 2, 3, 4 and 5. Let's rewrite $\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2$ expression as,

$$\begin{aligned}
E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 &= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_* + \boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + 2E\left[\left\langle \boldsymbol{g}^{t-1} - \boldsymbol{w}_*, \boldsymbol{g}^t - \boldsymbol{g}^{t-1}\right\rangle\right] + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + \frac{2}{\alpha m}\sum_{k\in[m]} E\left[\left\langle \boldsymbol{g}^{t-1} - \boldsymbol{w}_*, -\nabla L_k(\tilde{\boldsymbol{w}}_k^t)\right\rangle\right] + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&\leq \frac{2}{\alpha m}\sum_{k\in[m]} E\left[L_k(\boldsymbol{w}_*) - L_k(\boldsymbol{g}^{t-1}) + \frac{L}{2}\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2 - \frac{\mu}{2}\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{w}_*\|^2\right] \\
&\quad + E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&= E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 - \frac{2}{\alpha}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] + \frac{L}{\alpha}\epsilon_t - \frac{\mu}{\alpha}\frac{1}{m}\sum_{k\in[m]} E\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{w}_*\|^2 \\
&\quad + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \\
&\leq E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 - \frac{2}{\alpha}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] + \frac{L}{\alpha}\epsilon_t - \frac{\mu}{\alpha}E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 \\
&\quad + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2 \qquad \text{(A.10)}
\end{aligned}$$

where we first expand the square term and use Lemma 2. Following inequalities use Eq. A.9 and Lemma 9. Rearranging Eq. A.10 gives,

$$\begin{aligned}
&\left(1 + \frac{\mu}{\alpha}\right) E\|\boldsymbol{g}^t - \boldsymbol{w}_*\|^2 \\
&\qquad \leq E\|\boldsymbol{g}^{t-1} - \boldsymbol{w}_*\|^2 - \frac{2}{\alpha}E\left[\ell(\boldsymbol{g}^{t-1}) - \ell(\boldsymbol{w}_*)\right] + \frac{L}{\alpha}\epsilon_t + E\|\boldsymbol{g}^t - \boldsymbol{g}^{t-1}\|^2
\end{aligned} \qquad \text{(A.11)}$$

Let's define $z = \alpha^3 P + \alpha^2 P\mu - \alpha^2 m\mu - 20\alpha L^2 P + 4L^2 m\mu - 20L^2 P\mu$. Let's scale Lemma 4 and 5 with $\frac{\alpha(L+\alpha)(P\alpha+P\mu-m\mu)}{z}$ and $\frac{8m(L+\alpha)(\alpha+\mu)}{\alpha z}$ respectively. We note that the coefficients are positive due to the condition on α. Summing Eq. A.11, Lemma 3, scaled versions of Lemma 5 and 4 gives the statement in Lemma 8. □

We give Lemma 9 and its proof here.

Lemma 9 *Algorithm 1 satisfies*

$$-\frac{1}{m}\sum_{k\in[m]} E\|\tilde{\boldsymbol{w}}_k^t-\boldsymbol{w}_*\|^2\leq -E\|\boldsymbol{g}^t-\boldsymbol{w}_*\|^2$$

Proof.

$$\begin{aligned}E\|\boldsymbol{g}^t-\boldsymbol{w}_*\|^2 =&E\left\|\frac{1}{P}\sum_{k\in\mathcal{P}_t}\left(\boldsymbol{w}_k^t-\boldsymbol{w}_*\right)\right\|^2\leq\frac{1}{P}E\left[\sum_{k\in\mathcal{P}_t}\left\|\boldsymbol{w}_k^t-\boldsymbol{w}_*\right\|^2\right]\\ =&\frac{1}{P}E\left[\sum_{k\in\mathcal{P}_t}\left\|\tilde{\boldsymbol{w}}_k^t-\boldsymbol{w}_*\right\|^2\right]=\frac{1}{m}\sum_{k\in[m]}E\left\|\tilde{\boldsymbol{w}}_k^t-\boldsymbol{w}_*\right\|^2\end{aligned}$$

where first equality comes from Eq. A.7. The following inequality applies Jensen. Remaining relations are due to $\tilde{\boldsymbol{w}}_k^t=\boldsymbol{w}_k^t$ if $k\in\mathcal{P}_t$ and taking expectation by conditioning on randomness before time t. Rearranging the terms gives the statement in Lemma.□

Nonconvex Analysis

We state convergence for nonconvex L smooth $\{L_k\}_{k=1}^m$s as,

Theorem 7 *For nonconvex and L smooth $\{L_k\}_{k=1}^m$ functions and $\alpha\geq 20L\frac{m}{P}$, Algorithm 1 satisfies*

$$\begin{aligned}&E\left[\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla\ell(\boldsymbol{g}^{t-1})\right\|^2\right]\\ &\quad\leq\frac{1}{T}\left(3\alpha\left(\ell(\boldsymbol{w}^0)-\ell_*\right)+30L^3\frac{m}{P}\frac{1}{\alpha}\left(\frac{1}{m}\sum_{k\in[m]}E\|\boldsymbol{w}_k^0-\boldsymbol{w}^0\|^2\right)\right)=O\left(\frac{1}{T}\right)\end{aligned}$$

where $\boldsymbol{g}^t=\frac{1}{P}\sum_{k\in\mathcal{P}_t}\boldsymbol{w}_k^t$, $\ell_*=\min_{\boldsymbol{w}}\ell(\boldsymbol{w})$.

If $\alpha=30L\frac{m}{P}$, we get the statement in Theorem 1. We will use $\{\tilde{\boldsymbol{w}}_k^t\}$ and $\boldsymbol{g}^t$ variables as defined Eq. A.5, A.6, A.7. Since we aim to find a stationary in the nonconvex case, let's define a new C_t and keep ϵ_t the same as,

$$C_t = \frac{1}{m} \sum_{k \in [m]} E\|\boldsymbol{w}_k^t - \boldsymbol{g}^t\|^2, \quad \epsilon_t = \frac{1}{m} \sum_{k \in [m]} E\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2$$

Similarly, C_t tracks how well local models approximate the current active device average. Upon convergence C_t and ϵ_t will be 0.

Theorem 7 can be seen as a direct consequence of the following Lemma,

Lemma 10 *For L smooth $\{L_k\}_{k=1}^m$ functions, if $\alpha \geq 20L\frac{m}{P}$, Algorithm 1 satisfies*

$$E\left[\ell(\boldsymbol{g}^t)\right] + \kappa C_t \leq E\left[\ell(\boldsymbol{g}^{t-1})\right] + \kappa C_{t-1} - \kappa_0 E\left\|\nabla \ell(\boldsymbol{g}^{t-1})\right\|^2$$

where $\kappa = 4L^3P\frac{\alpha+L}{\alpha}\frac{2m-P}{z}$, $\kappa_0 = \frac{1}{2\alpha}\frac{\alpha^2P^2 - 4\alpha LP^2 - 32L^2m^2 - 16L^2Pm - 24L^2P^2}{z}$,
$z = \alpha^2P^2 - 32L^2m^2 + 16L^2Pm - 20L^2P^2$.

Lemma 10 can be telescoped as,

$$\kappa_0 E\left\|\nabla \ell(\boldsymbol{g}^{t-1})\right\|^2 \leq \left(E\left[\ell(\boldsymbol{g}^{t-1})\right] - \ell_* + \kappa C_{t-1}\right) - \left(E\left[\ell(\boldsymbol{g}^t)\right] - \ell_* + \kappa C_t\right)$$

$$\kappa_0 \sum_{t=1}^{T} E\left\|\nabla \ell(\boldsymbol{g}^{t-1})\right\|^2 \leq \left(E\left[\ell(\boldsymbol{g}^0)\right] - \ell_* + \kappa C_0\right) - \left(E\left[\ell(\boldsymbol{g}^T)\right] - \ell_* + \kappa C_T\right)$$

If $\alpha \geq 20L\frac{m}{P}$, we have κ_0 and κ as positive quantities. By definition, we also have C_t sequences as positive. Eliminating negative terms on RHS and summing over time give,

$$E\left[\frac{1}{T}\sum_{t=1}^{T}\left\|\nabla \ell(\boldsymbol{g}^{t-1})\right\|^2\right] \leq \frac{1}{T}\frac{1}{\kappa_0}\left(\ell(\boldsymbol{w}^0) - \ell_* + \kappa\left(\frac{1}{m}\sum_{k \in [m]} E\|\boldsymbol{w}_k^0 - \boldsymbol{w}^0\|^2\right)\right)$$

which proves the statement in Theorem 5.

The proof of Lemma 10 builds on Eq. A.2 where we upper bound $\ell(\boldsymbol{g}^t)$ with $\ell(\boldsymbol{g}^{t-1})$. Eq. A.2 gives $(\boldsymbol{g}^t - \boldsymbol{g}^{t-1})$ and $\nabla \ell(\boldsymbol{g}^{t-1})$ on RHS. We state a set of Lemmas to tackle these terms. We note here that Lemma 2 and 3 holds since ϵ_t is the same as in convex case.

To bound excess ϵ_t term, we introduce two more Lemmas as

Lemma 11 *For L smooth $\{L_k\}_{k=1}^m$ functions, Algorithm 1 satisfies*

$$\left(1-4L^2\frac{1}{\alpha^2}\right)\epsilon_t \leq 8L^2\frac{1}{\alpha^2}C_{t-1}+4\frac{1}{\alpha^2}E\left\|\nabla\ell(\boldsymbol{g}^{t-1})\right\|^2$$

Lemma 12 *For L smooth $\{L_k\}_{k=1}^m$ functions, Algorithm 1 satisfies*

$$C_t \leq 2\frac{m-P}{2m-P}C_{t-1}+2\frac{P}{2m-P}\epsilon_t+2\frac{m}{P}E\left\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\right\|^2$$

Using Eq. A.2 we get,

$$\begin{aligned}
E\left[\ell(\boldsymbol{g}^t)\right]-E\left[\ell(\boldsymbol{g}^{t-1})\right]-\frac{L}{2}E\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\|^2 &\leq E\left[\left\langle\nabla\ell(\boldsymbol{g}^{t-1}),\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\right\rangle\right]\\
=&\frac{1}{\alpha}E\left[\left\langle\nabla\ell(\boldsymbol{g}^{t-1}),\frac{1}{m}\sum_{k\in[m]}-\nabla L_k(\tilde{\boldsymbol{w}}_k^t)\right\rangle\right]\\
\leq&\frac{1}{2\alpha}E\left\|\frac{1}{m}\sum_{k\in[m]}\left(\nabla L_k(\tilde{\boldsymbol{w}}_k^t)-\nabla L_k(\boldsymbol{g}^{t-1})\right)\right\|^2-\frac{1}{2\alpha}E\left\|\nabla\ell(\boldsymbol{g}^{t-1})\right\|^2\\
\leq&\frac{1}{2\alpha}\frac{1}{m}\sum_{k\in[m]}E\left\|\nabla L_k(\tilde{\boldsymbol{w}}_k^t)-\nabla L_k(\boldsymbol{g}^{t-1})\right\|^2-\frac{1}{2\alpha}E\left\|\nabla\ell(\boldsymbol{g}^{t-1})\right\|^2\\
\leq&\frac{L^2}{2\alpha}\epsilon_t-\frac{1}{2\alpha}E\left\|\nabla\ell(\boldsymbol{g}^{t-1})\right\|^2 \qquad \text{(A.12)}
\end{aligned}$$

where first equality uses Lemma 2. The following inequalities are due to $\langle\boldsymbol{a},\boldsymbol{b}\rangle \leq \frac{1}{2}\|\boldsymbol{b}+\boldsymbol{a}\|^2-\frac{1}{2}\|\boldsymbol{a}\|^2$, Jensen Inq. and smoothness.

Let's define $z=\alpha^2P^2-32L^2m^2+16L^2Pm-20L^2P^2$ and scale Lemma 12, 3 and 11 with $z_0=4L^3P\frac{\alpha+L}{\alpha}\frac{2m-P}{z}$,
$z_1=\frac{L}{2}+z_0\frac{2m}{P}$, and $z_2=LP^2\frac{\alpha}{2}\frac{L+\alpha}{z}$ respectively. We note that the coefficients are positive due to the condition on α. Summing Eq. A.12, scaled versions of Lemma 3, 11 and 12 gives the statement in Lemma 10. □

Lastly, we note that the convergence analysis is given with respect to L2 norm in the gradients. L2 norm arises in the analysis because Eq. A.2 has L2 norm due to our definition of smoothness. Furthermore, the analysis can be extended to different norms. To do so, smoothness needs to be defined with respect to primal and dual norms as in Eq. 3 in (Nesterov et al., 2020).

We give the omitted proofs here.

Proof of Lemma 11

$$
\begin{aligned}
\epsilon_t =& \frac{1}{m}\sum_{k\in[m]} E\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{g}^{t-1}\|^2 = \frac{1}{m}\sum_{k\in[m]} E\left\|\tilde{\boldsymbol{w}}_k^t - \boldsymbol{w}^{t-1} - \frac{1}{\alpha}\boldsymbol{h}^{t-1}\right\|^2 \\
=&\frac{1}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \boldsymbol{h}^{t-1}\|^2 \\
=&\frac{1}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{g}^{t-1}) + \nabla L_k(\boldsymbol{g}^{t-1}) \\
&- \nabla L_k(\tilde{\boldsymbol{w}}_k^t) - \nabla \ell(\boldsymbol{g}^{t-1}) + \nabla \ell(\boldsymbol{g}^{t-1}) - \boldsymbol{h}^{t-1}\|^2 \\
\leq&\frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t)\|^2 \\
&+ \frac{4}{\alpha^2}E\|\nabla \ell(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}E\|\nabla \ell(\boldsymbol{g}^{t-1}) - \boldsymbol{h}^{t-1}\|^2 \\
\leq&\frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{w}_k^{t-1}) - \nabla L_k(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\tilde{\boldsymbol{w}}_k^t)\|^2 \\
&+ \frac{4}{\alpha^2}E\|\nabla \ell(\boldsymbol{g}^{t-1})\|^2 + \frac{4}{\alpha^2}\frac{1}{m}\sum_{k\in[m]} E\|\nabla L_k(\boldsymbol{g}^{t-1}) - \nabla L_k(\boldsymbol{w}_k^{t-1})\|^2 \\
\leq&\frac{8L^2}{\alpha^2}C_{t-1} + \frac{4L^2}{\alpha^2}\epsilon_t + \frac{4}{\alpha^2}E\|\nabla \ell(\boldsymbol{g}^{t-1})\|^2
\end{aligned}
$$

where first, second and third come from definition of ϵ_t, Eq. A.7 and A.6. The following inequalities are due to Lemma 6, Jensen Inq. and smoothness. Rearranging terms gives the Lemma.□

Proof of Lemma 12

$$
C_t = \frac{1}{m}\sum_{k\in[m]} E\|\boldsymbol{w}_k^t - \boldsymbol{g}^t\|^2 = \frac{1}{m}\sum_{k\in[m]} E\|\boldsymbol{w}_k^t - \boldsymbol{g}^{t-1} + \boldsymbol{g}^{t-1} - \boldsymbol{g}^t\|^2
$$

$$\begin{aligned}
&\leq \left(1+\frac{P}{2m-P}\right)\frac{1}{m}\sum_{k\in[m]} E\|\boldsymbol{w}_k^t-\boldsymbol{g}^{t-1}\|^2 \\
&\quad+\left(1+\frac{2m-P}{P}\right)\frac{1}{m}\sum_{k\in[m]} E\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\|^2 \\
&=\frac{P}{m}\left(1+\frac{P}{2m-P}\right)\frac{1}{m}\sum_{k\in[m]} E\|\tilde{\boldsymbol{w}}_k^t-\boldsymbol{g}^{t-1}\|^2 \\
&\quad+\left(1-\frac{P}{m}\right)\left(1+\frac{P}{2m-P}\right)\frac{1}{m}\sum_{k\in[m]} E\|\boldsymbol{w}_k^{t-1}-\boldsymbol{g}^{t-1}\|^2 \\
&\quad+\left(1+\frac{2m-P}{P}\right)E\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\|^2 \\
&=\frac{P}{m}\left(1+\frac{P}{2m-P}\right)\epsilon_t+\left(1-\frac{P}{m}\right)\left(1+\frac{P}{2m-P}\right)C_{t-1} \\
&\quad+\left(1+\frac{2m-P}{P}\right)E\|\boldsymbol{g}^t-\boldsymbol{g}^{t-1}\|^2
\end{aligned}$$

where we start with definition of C_t. First inequality is due to $\|\boldsymbol{a}+\boldsymbol{b}\|^2 \leq (1+z)\,\|\boldsymbol{a}\|^2 + \left(1+\frac{1}{z}\right)\|\boldsymbol{b}\|^2$ for $z>0$. The following equality takes expectation conditioned on randomness before time t. Since each device is selected with probability $\frac{P}{m}$, $\boldsymbol{w}_k^t$ is a random variable that is equal to $\tilde{\boldsymbol{w}}_k^t$ with probability $\frac{P}{m}$. Otherwise, it is $\boldsymbol{w}_k^{t-1}$. Final equality is due to definitions of ϵ_t and C_t.□

PFL

In this section, we mainly follow the analysis in (Karimireddy et al., 2019) and (Acar et al., 2021a) by modifying device functions so that we now need to consider $f_i \circ T_i$. Additionally, we set variance to be 0 ($\sigma = 0$) and allow for arbitrary SGD updates to ensure reaching a stationary point at each round. We refer to the FedDyn proof presented above for a detailed analysis.

We give proof of Proposition 3.2.1.

Proof of Proposition 3.2.1

As stated in Proposition 3.2.1, we assume that the device meta models converge. Then, $\boldsymbol{g}_i$s converge as, $\lim_{t\to\infty} \boldsymbol{w}_i^t = \boldsymbol{w}_i^\infty \implies \lim_{t\to\infty} \boldsymbol{g}_i^t = \nabla f_i\left(\overline{\boldsymbol{w}}_{i,i}^\infty\right)$ where $\overline{\boldsymbol{w}}_{i,i}^\infty = T_i(\boldsymbol{w}_i^\infty)$.

Convergence of $\boldsymbol{g}_i$s and the update rule, $\boldsymbol{g}_i^{t+1} = \boldsymbol{g}_i^t - \alpha\left(\boldsymbol{w}_i^{t+1} - \boldsymbol{w}^t\right)$, imply $\boldsymbol{w}_i^\infty = \boldsymbol{w}^\infty$ and $\overline{\boldsymbol{w}}_{i,i}^\infty = T_i(\boldsymbol{w}^\infty)$ i.e. each device meta model converges to the same meta model. Rearranging the server update gives $\boldsymbol{g}^t = \alpha\left(-\boldsymbol{w}^t + \frac{1}{|\mathcal{P}_t|}\sum_{i\in\mathcal{P}_t} \boldsymbol{w}_i^t\right)$. Since we have $\boldsymbol{w}_i^\infty = \boldsymbol{w}^\infty$ for all is, we get $\lim_{t\to\infty} \boldsymbol{g}^t = \mathbf{0}$. Using server update we conclude that $\lim_{t\to\infty} \boldsymbol{g}^t = \frac{1}{m}\sum_{i\in[m]} \nabla f_i\left(\overline{\boldsymbol{w}}_i^\infty\right) = \mathbf{0}$ where $\overline{\boldsymbol{w}}_i^\infty = T_i(\boldsymbol{w}^\infty)$. Hence, PFL eliminates the bias coming from heterogeneity of devices and it converges to a stationary point of the personalized federated learning objective OPT. □

A.2.2 MOML

Convex Analysis

Assumption 2 *(Stationary point) We assume that MOML finds a stationary point of the risk it minimizes. Formally, at each round, MOML satisfies*

$$\nabla f^t \circ U^t(\boldsymbol{w}^{t+1}) - \nabla f^{t-1} \circ U^{t-1}(\boldsymbol{w}^t) + \alpha\left(\boldsymbol{w}^{t+1} - \boldsymbol{w}^t\right) = \mathbf{0}.$$

This assumption can be achieved by tuning parameter K. Parameter K controls the correctness of the solution as such it would introduce $O\left(\frac{1}{K}\right)$ noise at each round. Choosing $K = O\left(\sqrt{T}\right)$ would be sufficient.

Assumption 3 *(Bounded gradients with G) $\{f^t \circ U^t\}_{t=1}^T$ functions have bounded gradients with G .i.e*

$$\|\nabla f^t \circ U^t(\boldsymbol{w})\| \leq G \quad \forall \boldsymbol{w}, t$$

Theorem 8 *For adversarial convex $\{f^t \circ U^t\}_{t=1}^T$ functions we have, Algorithm 6 sat-*

isfies,

$$R_T = \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^t) - \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^*) \leq \alpha \|\boldsymbol{w}^*\|^2 + \frac{3}{\alpha} \sum_{t=1}^{T} \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2,$$

where $\boldsymbol{w}^* = arg \min\limits_{\boldsymbol{w} \in \mathbb{R}^d} \; \sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w})$.

If we plug in $\alpha = O\left(\sqrt{T}\right)$ and bound gradients with G in Theorem 8, we recover Theorem 3.

Proof.

We divide LHS with two terms as,

$$R_T = \left(\sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^t) - f^t \circ U^t(\boldsymbol{w}^{t+1}) \right) + \left(\sum_{t=1}^{T} f^t \circ U^t(\boldsymbol{w}^{t+1}) - f^t \circ U^t(\boldsymbol{w}^*) \right)$$

where first term corresponds to the cost we incur by not using $\boldsymbol{w}^{t+1}$ and the second term quantifies how good $\boldsymbol{w}^{t+1}$ is with respect to the competitor.

For the sake of simplicity, let us define the local states with the gradient as

$$\boldsymbol{\lambda}^t = \nabla f^t \circ U^t(\boldsymbol{w}^{t+1}). \tag{A.13}$$

We bound individual terms with the following Lemmas,

Lemma 13 *Algorithm 6 satisfies,*

$$\begin{aligned} f^t \circ U^t(\boldsymbol{w}^t) - f^t \circ U^t(\boldsymbol{w}^{t+1}) \leq & \frac{3}{\alpha} \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 \\ & + \frac{\alpha}{4} \left\| \boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 \\ & + \frac{\alpha}{4} \left\| \boldsymbol{w}^t - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 + \frac{\alpha}{16} \|\boldsymbol{\lambda}^t\|^2 \end{aligned}$$

Lemma 14 *Algorithm 6 satisfies,*

$$f^t \circ U^t(\boldsymbol{w}^{t+1}) - f^t \circ U^t(\boldsymbol{w}^*)$$

$$\begin{aligned}
&\leq\alpha\left(\left\|\boldsymbol{w}^* - \left(\boldsymbol{w}^t + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right)\right\|^2 - \left\|\boldsymbol{w}^* - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\right\|^2\right)\\
&\quad + \frac{1}{4\alpha}\left(\|\boldsymbol{\lambda}^{t-1}\|^2 - \|\boldsymbol{\lambda}^t\|^2\right) - \frac{\alpha}{2}\left\|\boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\right\|^2 - \frac{1}{4\alpha}\|\boldsymbol{\lambda}^t\|^2
\end{aligned}$$

If we plug in Lemma 13 and 14 in the regret statement we get,

$$\begin{aligned}
R_T \leq&\alpha\left(\sum_{t=1}^{T}\left\|\boldsymbol{w}^* - \left(\boldsymbol{w}^t + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right)\right\|^2 - \left\|\boldsymbol{w}^* - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\right\|^2\right)\\
&+ \frac{\alpha}{4}\left(\sum_{t=1}^{T}\left\|\boldsymbol{w}^t - \left(\boldsymbol{w}^t + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right)\right\|^2 - \left\|\boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\right\|^2\right)\\
&+ \frac{1}{4\alpha}\left(\sum_{t=1}^{T}\|\boldsymbol{\lambda}^{t-1}\|^2 - \|\boldsymbol{\lambda}^t\|^2\right) + \frac{3}{\alpha}\sum_{t=1}^{T}\|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 - \frac{3}{16\alpha}\sum_{t=1}^{T}\|\boldsymbol{\lambda}^t\|^2\\
\leq&\alpha\left\|\boldsymbol{w}^* - \left(\boldsymbol{w}^1 + \frac{1}{2\alpha}\boldsymbol{\lambda}^0\right)\right\|^2 + \frac{\alpha}{4}\left\|\boldsymbol{w}^1 - \left(\boldsymbol{w}^1 + \frac{1}{2\alpha}\boldsymbol{\lambda}^0\right)\right\|^2 + \frac{1}{4\alpha}\|\boldsymbol{\lambda}^0\|^2\\
&+ \frac{3}{\alpha}\sum_{t=1}^{T}\|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 = \alpha\left\|\boldsymbol{w}^*\right\|^2 + \frac{3}{\alpha}\sum_{t=1}^{T}\|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2
\end{aligned}$$

where the second inequality is due to telescoping and ignoring non-positive terms and the last equality comes from the initial conditions $\boldsymbol{\lambda}^0 = \boldsymbol{w}^1 = \boldsymbol{w}^1 = \mathbf{0}$. This completes the proof of Theorem 8. □

We give proof of Lemmas here. We first state two relations that are useful in the proof.

$$\begin{aligned}
\left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right) - \left(\boldsymbol{w}^t + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right) =& \left(\frac{1}{2}\left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1} - \frac{1}{\alpha}\boldsymbol{\lambda}^t\right) + \frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\\
&- \left(\boldsymbol{w}^t + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right)\\
=&\frac{1}{2}\left(-\boldsymbol{w}^t + \boldsymbol{w}^{t+1} - \frac{1}{\alpha}\boldsymbol{\lambda}^{t-1}\right) = -\frac{1}{2\alpha}\boldsymbol{\lambda}^t,
\end{aligned} \tag{A.14}$$

where the equalities are due to $\boldsymbol{w}$ update (Eq. 5.8) and first order condition (Eq. 5.10) respectively.

Proof of Lemma 13

$$
\begin{aligned}
f^t&\circ U^t(\boldsymbol{w}^t) - f^t \circ U^t(\boldsymbol{w}^{t+1}) \leq \langle \nabla f^t \circ U^t(\boldsymbol{w}^t), \boldsymbol{w}^t - \boldsymbol{w}^{t+1} \rangle \\
\leq& \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\| \|\boldsymbol{w}^t - \boldsymbol{w}^{t+1}\| \\
\leq& \frac{3}{\alpha} \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 + \frac{\alpha}{12} \|\boldsymbol{w}^t - \boldsymbol{w}^{t+1}\|^2 \\
\leq& \frac{3}{\alpha} \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 + \frac{\alpha}{4} \left\| \boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 \\
&+ \frac{\alpha}{4} \left\| \boldsymbol{w}^t - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 + \frac{\alpha}{4} \left\| \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 \\
=& \frac{3}{\alpha} \|\nabla f^t \circ U^t(\boldsymbol{w}^t)\|^2 + \frac{\alpha}{4} \left\| \boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 \\
&+ \frac{\alpha}{4} \left\| \boldsymbol{w}^t - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 + \frac{\alpha}{16} \|\boldsymbol{\lambda}^t\|^2
\end{aligned}
$$

where inequalities come from convexity, Cauchy–Schwarz Eq., $2\langle \boldsymbol{w}_1, \boldsymbol{w}_2 \rangle \leq \frac{1}{c} \|\boldsymbol{w}_1\|^2 + c\|\boldsymbol{w}_2\|^2$ and triangular inequality respectively. Last equality is due to Eq. A.14. □

We state a useful relation to be used in the proof of Lemma 14.

Proposition A.2.1

$$
\begin{aligned}
&-\alpha \left\| \boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right\|^2 + \alpha \left\| \boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right\|^2 + \langle \boldsymbol{\lambda}^t, \boldsymbol{w}^{t+1} \rangle \\
&\quad = \frac{1}{4\alpha} \left(\|\boldsymbol{\lambda}^{t-1}\|^2 - \|\boldsymbol{\lambda}^t\|^2 \right) - \alpha \left\| \boldsymbol{w}^{t+1} - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 - \frac{1}{4\alpha} \|\boldsymbol{\lambda}^t\|^2
\end{aligned}
$$

Proof of Lemma 14

$$
\begin{aligned}
f^t&\circ U^t(\boldsymbol{w}^{t+1}) - f^t \circ U^t(\boldsymbol{w}^*) \leq \langle \nabla f^t \circ U^t(\boldsymbol{w}^{t+1}), \boldsymbol{w}^{t+1} - \boldsymbol{w}^* \rangle = \langle \boldsymbol{\lambda}^t, \boldsymbol{w}^{t+1} - \boldsymbol{w}^* \rangle \\
=& \alpha \left(\left\| \boldsymbol{w}^* - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 - \left\| \boldsymbol{w}^* - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 \right) \\
&- \alpha \left\| \boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right\|^2 + \alpha \left\| \boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right\|^2 + \langle \boldsymbol{\lambda}^t, \boldsymbol{w}^{t+1} \rangle \\
=& \alpha \left(\left\| \boldsymbol{w}^* - \left(\boldsymbol{w}^t + \frac{1}{2\alpha} \boldsymbol{\lambda}^{t-1} \right) \right\|^2 - \left\| \boldsymbol{w}^* - \left(\boldsymbol{w}^{t+1} + \frac{1}{2\alpha} \boldsymbol{\lambda}^t \right) \right\|^2 \right)
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{4\alpha}\left(\|\boldsymbol{\lambda}^{t-1}\|^2-\|\boldsymbol{\lambda}^{t}\|^2\right)-\alpha\left\|\boldsymbol{w}^{t+1}-\left(\boldsymbol{w}^{t+1}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right)\right\|^2-\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t}\|^2\\
\leq&\alpha\left(\left\|\boldsymbol{w}^{*}-\left(\boldsymbol{w}^{t}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right)\right\|^2-\left\|\boldsymbol{w}^{*}-\left(\boldsymbol{w}^{t+1}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right)\right\|^2\right)\\
&+\frac{1}{4\alpha}\left(\|\boldsymbol{\lambda}^{t-1}\|^2-\|\boldsymbol{\lambda}^{t}\|^2\right)-\frac{\alpha}{2}\left\|\boldsymbol{w}^{t+1}-\left(\boldsymbol{w}^{t+1}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right)\right\|^2-\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t}\|^2
\end{aligned}
$$

where we use convexity, Eq. A.13, A.14 and Proposition A.2.1.

We give the proof of the proposition here.

Proof of Proposition A.2.1

Let us expand LHS and state it as a polynomial of $\boldsymbol{w}^{t+1}$, .i.e $A\|\boldsymbol{w}^{t+1}\|^2+\langle\boldsymbol{w}^{t+1},\boldsymbol{w}'\rangle+\boldsymbol{w}''$

$$
\begin{aligned}
LHS=&-\alpha\left\|\boldsymbol{w}^{t}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right\|^2+\alpha\left\|\frac{1}{2}\left(\boldsymbol{w}^{t}+\boldsymbol{w}^{t+1}-\frac{1}{\alpha}\boldsymbol{\lambda}^{t}\right)+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right\|^2+\langle\boldsymbol{\lambda}^{t},\boldsymbol{w}^{t+1}\rangle\\
=&-\alpha\left\|\boldsymbol{w}^{t}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right\|^2+\frac{\alpha}{4}\left\|\boldsymbol{w}^{t}+\boldsymbol{w}^{t+1}\right\|^2+\langle\boldsymbol{\lambda}^{t},\boldsymbol{w}^{t+1}\rangle\\
=&-\alpha\left\|\boldsymbol{w}^{t}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right\|^2+\frac{\alpha}{4}\left\|\boldsymbol{w}^{t}+\boldsymbol{w}^{t+1}\right\|^2+\langle\boldsymbol{\lambda}^{t-1}+\alpha\boldsymbol{w}^{t}-\alpha\boldsymbol{w}^{t+1},\boldsymbol{w}^{t+1}\rangle\\
=&\|\boldsymbol{w}^{t+1}\|^2\left(\frac{\alpha}{4}-\alpha\right)+\left\langle\boldsymbol{w}^{t+1},\frac{\alpha}{2}\boldsymbol{w}^{t}+\alpha\boldsymbol{w}^{t}+\boldsymbol{\lambda}^{t-1}\right\rangle\\
&-\alpha\left\|\boldsymbol{w}^{t}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1}\right\|^2+\frac{\alpha}{4}\left\|\boldsymbol{w}^{t}\right\|^2\\
=&\|\boldsymbol{w}^{t+1}\|^2\left(-\frac{3\alpha}{4}\right)+\left\langle\boldsymbol{w}^{t+1},\frac{3\alpha}{2}\boldsymbol{w}^{t}+\boldsymbol{\lambda}^{t-1}\right\rangle\\
&-\frac{3\alpha}{4}\left\|\boldsymbol{w}^{t}\right\|^2-\frac{1}{4\alpha}\left\|\boldsymbol{\lambda}^{t-1}\right\|^2-\langle\boldsymbol{w}^{t},\boldsymbol{\lambda}^{t-1}\rangle \qquad \text{(A.15)}
\end{aligned}
$$

where we use $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ updates. Similarly if we expand RHS we get,

$$
\begin{aligned}
RHS=&\frac{1}{4\alpha}\left(\|\boldsymbol{\lambda}^{t-1}\|^2-\|\boldsymbol{\lambda}^{t}\|^2\right)-\alpha\left\|\boldsymbol{w}^{t+1}-\left(\boldsymbol{w}^{t+1}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right)\right\|^2-\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t}\|^2\\
=&\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t-1}\|^2-\frac{1}{2\alpha}\|\boldsymbol{\lambda}^{t-1}+\alpha\boldsymbol{w}^{t}-\alpha\boldsymbol{w}^{t+1}\|^2-\alpha\left\|\boldsymbol{w}^{t+1}-\left(\boldsymbol{w}^{t+1}+\frac{1}{2\alpha}\boldsymbol{\lambda}^{t}\right)\right\|^2\\
=&\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t-1}\|^2-\frac{1}{2\alpha}\|\boldsymbol{\lambda}^{t-1}+\alpha\boldsymbol{w}^{t}-\alpha\boldsymbol{w}^{t+1}\|^2
\end{aligned}
$$

$$
\begin{aligned}
&-\alpha\left\|\boldsymbol{w}^{t+1}-\left(\frac{1}{2}\left(\boldsymbol{w}^t+\boldsymbol{w}^{t+1}-\frac{1}{\alpha}\boldsymbol{\lambda}^t\right)+\frac{1}{2\alpha}\boldsymbol{\lambda}^t\right)\right\|^2\\
=&\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t-1}\|^2-\frac{1}{2\alpha}\|\boldsymbol{\lambda}^{t-1}+\alpha\boldsymbol{w}^t-\alpha\boldsymbol{w}^{t+1}\|^2-\frac{\alpha}{4}\left\|\boldsymbol{w}^{t+1}-\boldsymbol{w}^t\right\|^2\\
=&\|\boldsymbol{w}^{t+1}\|^2\left(-\frac{\alpha}{2}-\frac{\alpha}{4}\right)+\left\langle\boldsymbol{w}^{t+1},\alpha\boldsymbol{w}^t+\boldsymbol{\lambda}^{t-1}+\frac{\alpha}{2}\boldsymbol{w}^t\right\rangle\\
&+\frac{1}{4\alpha}\|\boldsymbol{\lambda}^{t-1}\|^2-\frac{\alpha}{4}\|\boldsymbol{w}^t\|^2-\frac{1}{2\alpha}\|\boldsymbol{\lambda}^{t-1}+\alpha\boldsymbol{w}^t\|^2\\
=&\|\boldsymbol{w}^{t+1}\|^2\left(-\frac{3\alpha}{4}\right)+\left\langle\boldsymbol{w}^{t+1},\frac{3\alpha}{2}\boldsymbol{w}^t+\boldsymbol{\lambda}^{t-1}\right\rangle\\
&-\frac{3\alpha}{4}\left\|\boldsymbol{w}^t\right\|^2-\frac{1}{4\alpha}\left\|\boldsymbol{\lambda}^{t-1}\right\|^2-\langle\boldsymbol{w}^t,\boldsymbol{\lambda}^{t-1}\rangle
\end{aligned}
\tag{A.16}
$$

where we use $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ updates.

We have Eq. A.15 is equal to Eq. A.16 so the statement holds. □

Nonconvex Analysis

We prove regret statements of B-MOML and Theorem 4 with the following subsections.

(Hazan et al., 2017) regret with B-MOML

Assumption 4 *(Stationary point) Similar to Assumption 2, we assume that B-MOML finds a stationary point of the risk it minimizes. Formally, at each round, B-MOML satisfies*

$$\frac{1}{B}\sum_{i=0}^{B-1}\nabla f^{t-i}\circ U^{t-i}(\boldsymbol{w}^{t+1})-\frac{1}{B}\sum_{i=0}^{B-1}\nabla f^{t-i-1}\circ U^{t-i-1}(\boldsymbol{w}^{t-i})+\alpha\left(\boldsymbol{w}^{t+1}-\boldsymbol{w}^t\right)=\mathbf{0}.$$

We use Assumption 3 and 4 in this subsection.

Theorem 9 *For adversarial nonconvex functions, B-MOML satisfies,*

$$\sum_{t=1}^{T}\left\|\frac{1}{B}\sum_{i=0}^{B-1}\nabla f^{t-i}\circ U^{t-i}(\boldsymbol{w}^t)\right\|^2\leq 8\frac{T}{B^2}G^2$$

Proof. Let us define $S_B^t(\boldsymbol{w}) = \frac{1}{B}\sum_{i=0}^{B-1} \nabla f^{t-i} \circ U^{t-i}(\boldsymbol{w})$. Then, we have,

$$
\begin{aligned}
&\left\| \frac{1}{B}\sum_{i=0}^{B-1} \nabla f^{t-i} \circ U^{t-i}(\boldsymbol{w}^t) \right\|^2 \\
&= \|\nabla S_B^t(\boldsymbol{w}^t)\|^2 = \| \left(\nabla S_B^t(\boldsymbol{w}^t) - \nabla S_B^{t-1}(\boldsymbol{w}^t)\right) + \nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 \\
&\leq 2\|\nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 + 2\|\nabla S_B^t(\boldsymbol{w}^t) - \nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 \\
&= 2\|\nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 + \frac{2}{B^2}\left\|\nabla f^t \circ U^t(\boldsymbol{w}^t) - \nabla f^{t-B} \circ U^{t-B}(\boldsymbol{w}^t)\right\|^2 \\
&\leq 2\|\nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 + \frac{8}{B^2}G^2 \qquad \text{(A.17)}
\end{aligned}
$$

where inequalities come from triangular Inq. and Assumption 3.

We assume $\alpha = 0$. If $\alpha = 0$ we see that $\frac{1}{B}\sum_{i=0}^{B-1} \nabla f^{t-i} \circ U^{t-i}(\boldsymbol{w}^{t+1}) = \nabla S_B^t(\boldsymbol{w}^{t+1}) = \boldsymbol{0}$.

Plugging these relations in Eq. A.17, we get,

$$
\left\| \frac{1}{B}\sum_{i=0}^{B-1} \nabla f^{t-i} \circ U^{t-i}(\boldsymbol{w}^t) \right\|^2 \leq 2\|\nabla S_B^{t-1}(\boldsymbol{w}^t)\|^2 + \frac{8}{B^2}G^2 = 2\|\boldsymbol{\lambda}^{t-1}\|^2 + \frac{8}{B^2}G^2 = \frac{8}{B^2}G^2 \qquad \text{(A.18)}
$$

Summing Inq. A.18 over time gives the statement in Theorem 9. □

$\mathcal{T}$ collection of tasks type regret

Assumption 5 *(Smoothness) We assume $\{f^t \circ U^t\}_{t=1}^T$ functions to be L smooth .i.e*

$$
\|\nabla f^t \circ U^t(\boldsymbol{w}_1) - \nabla f^t \circ U^t(\boldsymbol{w}_2)\| \leq L\|\boldsymbol{w}_1 - \boldsymbol{w}_2\| \quad \forall \boldsymbol{w}_1, \boldsymbol{w}_2, t
$$

Smoothness imply the following inequality,

$$
f^t \circ U^t(\boldsymbol{w}_2) - f^t \circ U^t(\boldsymbol{w}_1) \leq \left\langle \nabla f^t \circ U^t(\boldsymbol{w}_1), \boldsymbol{w}_2 - \boldsymbol{w}_1 \right\rangle + \frac{L}{2}\|\boldsymbol{w}_2 - \boldsymbol{w}_1|^2 \quad \forall \boldsymbol{w}_1, \boldsymbol{w}_2, t \qquad \text{(A.19)}
$$

We use Assumption 2, 3 and 5 as well definition in Eq. A.13 for this subsection.

Theorem 10 *Suppose $\mathcal{T}$ is a collection of tasks, and for each $\tau \in \mathcal{T}$, $f^\tau \circ U^\tau$ is L smooth. We choose tasks i_t from some task distribution, $P_{\mathcal{T}}$ in an IID fashion. Then it follows that,*

$$\sum_{t=1}^{T} E \left\| E_\tau[\nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)] \right\|^2 \leq 4\alpha\Delta + T\frac{G^2L^2}{\alpha^2} + T\frac{1}{\alpha}\frac{11}{2}G^2L.$$

where $\Delta = E_\tau[f^\tau \circ U^\tau(\boldsymbol{w}^1)] - \min_{\boldsymbol{w}} E_\tau[f^\tau \circ U^\tau(\boldsymbol{w})]$

If we have $\alpha = O\left(\sqrt{T}\right)$, we get the bound in Theorem 4. Theorem 10 is can be derived from the following Lemma,

Lemma 15 *Algorithm 6 satisfies,*

$$\begin{aligned} & E \left\| E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t) \right\|^2 \\ \leq & 4\alpha \left(E\left[E_\tau f^\tau \circ U^\tau \left(\frac{1}{2}\left(\boldsymbol{w}^{t-1} + \boldsymbol{w}^t\right) \right) \right] - E\left[E_\tau f^\tau \circ U^\tau \left(\frac{1}{2}\left(\boldsymbol{w}^{t} + \boldsymbol{w}^{t+1}\right) \right) \right] \right) \\ & + \frac{G^2L^2}{\alpha^2} + \frac{1}{\alpha}\frac{11}{2}G^2L \end{aligned}$$

If we sum Lemma 15 over time we get

$$\begin{aligned} & \sum_{t=1}^{T} E \left\| E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t) \right\|^2 \\ & \leq 4\alpha \left(E\left[E_\tau f^\tau \circ U^\tau \left(\frac{1}{2}\left(\boldsymbol{w}^{0} + \boldsymbol{w}^1\right) \right) \right] - E\left[E_\tau f^\tau \circ U^\tau \left(\frac{1}{2}\left(\boldsymbol{w}^{T} + \boldsymbol{w}^{T+1}\right) \right) \right] \right) \\ & \quad + T\frac{G^2L^2}{\alpha^2} + T\frac{1}{\alpha}\frac{11}{2}G^2L \\ & \leq 4\alpha\Delta + T\frac{G^2L^2}{\alpha^2} + T\frac{1}{\alpha}\frac{11}{2}G^2L \end{aligned}$$

which is the statement in Theorem 10.

We use smoothness bound (Eq. A.19) to prove Lemma 15. First, we present a set of Lemmas that are useful to handle various terms and invoke these to prove this statement.

Lemma 16 *Define* $\boldsymbol{\rho}^t = \frac{1}{2}\left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1}\right)$*. Then, in Algorithm 6, we have,*

$$\boldsymbol{\rho}^t - \boldsymbol{\rho}^{t-1} = -\frac{1}{2\alpha}\boldsymbol{\lambda}^t$$

Proof.

$$\begin{aligned}\frac{1}{2}\left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1}\right) - \frac{1}{2}\left(\boldsymbol{w}^{t-1} + \boldsymbol{w}^t\right) &= \frac{1}{2}\left(\boldsymbol{w}^t + \boldsymbol{w}^{t+1}\right) - \boldsymbol{w}^t - \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1} \\ &= \frac{1}{2}\left(-\boldsymbol{w}^t + \boldsymbol{w}^{t+1}\right) - \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1} = -\frac{1}{2\alpha}\boldsymbol{\lambda}^t\end{aligned}$$

where first equality uses definition of $\boldsymbol{w}^t$ (Eq. 5.8) and the last one comes from the first order condition (Eq. 5.10). □

Lemma 17 *Algorithm 6 satisfies,*

$$E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\boldsymbol{\lambda}^t\right\rangle\right] \leq \frac{1}{\alpha}\frac{5}{2}G^2L + \frac{1}{2\alpha^2}G^2L^2 - \frac{1}{2}E\|E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)\|^2$$

We use smoothness Eq. A.19 on $E_\tau\left[f^\tau \circ U^\tau\right]$ as,

$$\begin{aligned}&E\left[E_\tau\left[f^\tau \circ U^\tau\left(\boldsymbol{\rho}^t\right)\right]\right] - E\left[E_\tau\left[f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right)\right]\right] \\ &\qquad \leq E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), \boldsymbol{\rho}^t - \boldsymbol{\rho}^{t-1}\right\rangle\right] + \frac{L}{2}E\|\boldsymbol{\rho}^t - \boldsymbol{\rho}^{t-1}\|^2 \\ &\qquad = \frac{1}{2\alpha}E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\boldsymbol{\lambda}^t\right\rangle\right] + \frac{L}{8\alpha^2}E\|\boldsymbol{\lambda}^t\|^2 \\ &\qquad \leq \frac{1}{\alpha^2}\frac{5}{4}G^2L + \frac{1}{4\alpha^3}G^2L^2 - \frac{1}{4\alpha}E\|E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)\|^2 + \frac{LG^2}{8\alpha^2}\end{aligned}$$

where we use Lemma 16, 17 and bound $\|\boldsymbol{\lambda}\|$ with Assumption 3. Rearranging terms give the statement in Lemma 15. □

Corollary 1 is a direct consequence of Theorem 10 in case of Polyak- Lojasiewicz (PL) functions. It gives a statement with respect to the best fixed competitor.

Proof of Corollary 1 Let's apply PL condition on LHS of Theorem 10. We get,

$$\sum_{t=1}^{T} E\left[E_\tau[f^\tau \circ U^\tau(\boldsymbol{w}^t)] - \min_{\boldsymbol{w}} E_\tau[f^\tau \circ U^\tau(\boldsymbol{w})]\right]$$

$$\leq \frac{1}{2\mu}\sum_{t=1}^{T} E\left\|E_\tau[\nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)]\right\|^2 = O\left(\sqrt{T}\frac{1}{\mu}\left(\Delta + G^2 L\right)\right)$$

This is the Corollary statement. □

We give proof of Lemma 17 here. We state two more Lemmas that are used in the proof.

Lemma 18 *Difference of consecutive meta models can be bounded as,*

$$\|\boldsymbol{w}^{t+1} - \boldsymbol{w}^t\| \leq \frac{5}{2}\frac{1}{\alpha}G$$

Proof.

If we subtract the first order condition (Eq. 5.10) for consecutive times, we get,

$$\boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-1} = \boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^{t-2} - \alpha(\boldsymbol{w}^{t+1} - \boldsymbol{w}^t) + \alpha(\boldsymbol{w}^t - \boldsymbol{w}^{t-1}) \tag{A.20}$$

Rearranging Eq. A.20 gives,

$$\begin{aligned}
\boldsymbol{w}^{t+1} - \boldsymbol{w}^t &= \frac{1}{\alpha}(2\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-2}) + (\boldsymbol{w}^t - \boldsymbol{w}^{t-1}) \\
&= \frac{1}{\alpha}(2\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-2}) + \frac{1}{2}(\boldsymbol{w}^t - \boldsymbol{w}^{t-1}) - \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1} \\
&= \frac{1}{\alpha}(2\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t - \boldsymbol{\lambda}^{t-2}) + \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-2} - \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1} - \frac{1}{2\alpha}\boldsymbol{\lambda}^{t-1} \\
&= \frac{1}{\alpha}\left(\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t - \frac{1}{2}\boldsymbol{\lambda}^{t-2}\right)
\end{aligned}$$

where we use update rule of $\boldsymbol{w}^t$ and $\boldsymbol{\lambda}^t$ in the second respectively. Since $\boldsymbol{\lambda}$ states store gradient information as in Eq. A.13 and gradients are bounded (Assumption 3), we can write,

$$\|\boldsymbol{w}^{t+1} - \boldsymbol{w}^t\| = \frac{1}{\alpha}\left\|\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^t - \frac{1}{2}\boldsymbol{\lambda}^{t-2}\right\| \leq \frac{1}{\alpha}\left\|\boldsymbol{\lambda}^{t-1}\right\| + \frac{1}{\alpha}\left\|\boldsymbol{\lambda}^t\right\| + \frac{1}{2}\frac{1}{\alpha}\left\|\boldsymbol{\lambda}^{t-2}\right\| \leq \frac{5}{2}\frac{1}{\alpha}G.$$

where we relax norm as $\|\boldsymbol{a} + \boldsymbol{b}\| \leq \|\boldsymbol{a}\| + \|\boldsymbol{b}\|$. □

Lemma 19

$$E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right] \leq \frac{1}{2\alpha^2}G^2L^2 - \frac{1}{2}E\|E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)\|^2$$

Proof.

$$\begin{aligned}
&E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right] \\
=&E\left[E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle|\mathcal{H}_t\right]\right] \\
=&E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla E_\tau f^\tau \circ U^\tau(\boldsymbol{w}^t)\right\rangle\right] \\
\leq& -\frac{1}{2}E\|E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)\|^2 \\
&+\frac{1}{2}E\left\|\nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right) - \nabla E_\tau f^\tau \circ U^\tau(\boldsymbol{w}^t)\right\|^2
\end{aligned}$$

where first equality comes from tower property noting that both $\boldsymbol{\rho}^{t-1}$ and $\boldsymbol{w}^t$ are independent of loss observed at time t, $f^t \circ U^t$. The inequality comes from $\langle \boldsymbol{a}, \boldsymbol{b}\rangle \leq \frac{1}{2}\|\boldsymbol{b}+\boldsymbol{a}\|^2 - \frac{1}{2}\|\boldsymbol{a}\|^2$. We bound the second term as

$$\begin{aligned}
&E\left\|\nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right) - \nabla E_\tau f^\tau \circ U^\tau(\boldsymbol{w}^t)\right\|^2 \\
\leq& L^2E\left\|\boldsymbol{\rho}^{t-1} - \boldsymbol{w}^t\right\|^2 \leq \frac{1}{4}L^2E\left\|\boldsymbol{w}^{t-1} - \boldsymbol{w}^t\right\|^2 \leq \frac{1}{4\alpha^2}L^2E\left\|\boldsymbol{\lambda}^{t-1} - \boldsymbol{\lambda}^{t-2}\right\|^2 \leq \frac{L^2G^2}{\alpha^2}
\end{aligned}$$

where we use smoothness, definition of $\boldsymbol{\rho}^{t-1}$, the first order condition (Eq. 5.10) and bound on gradients. Combining both inequalities gives the statement in the lemma. □

Proof of Lemma 17

$$\begin{aligned}
&E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\boldsymbol{\lambda}^t\right\rangle\right] = E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^{t+1})\right\rangle\right] \\
=&E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right] \\
&+E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), \nabla f^t \circ U^t(\boldsymbol{w}^t) - \nabla f^t \circ U^t(\boldsymbol{w}^{t+1})\right\rangle\right] \\
\leq& E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right]
\end{aligned}$$

$$
\begin{aligned}
&+ E\left[\left\|\nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right)\right\| \|\nabla f^t \circ U^t(\boldsymbol{w}^t) - \nabla f^t \circ U^t(\boldsymbol{w}^{t+1})\|\right] \\
\leq& E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right] + E\left[GL\|\boldsymbol{w}^t - \boldsymbol{w}^{t+1}\|\right] \\
\leq& E\left[\left\langle \nabla E_\tau f^\tau \circ U^\tau\left(\boldsymbol{\rho}^{t-1}\right), -\nabla f^t \circ U^t(\boldsymbol{w}^t)\right\rangle\right] + \frac{5}{2}\frac{1}{\alpha}G^2L \\
\leq& \frac{5}{2}\frac{1}{\alpha}G^2L + \frac{1}{2\alpha^2}G^2L^2 - \frac{1}{2}E\|E_\tau \nabla f^\tau \circ U^\tau(\boldsymbol{w}^t)\|^2
\end{aligned}
$$

where we use Cauchy–Schwarz Eq., smoothness, gradient bound, Lemma 18 and 19 respectively. □

References

Abernethy, J., Hazan, E., and Rakhlin, A. (2008). Competing in the dark: An efficient algorithm for bandit linear optimization. In *21st Annual Conference on Learning Theory, COLT 2008*, pages 263–274.

Acar, D. A. E. and Saligrama, V. (2022). Fedhen: Federated learning in heterogeneous networks. *arXiv preprint arXiv:2207.03031*.

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. (2021a). Federated learning based on dynamic regularization. In *International Conference on Learning Representations*.

Acar, D. A. E., Zhao, Y., Zhu, R., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. (2021b). Debiasing model updates for improving personalized federated training. In *International Conference on Machine Learning*, pages 21–31. PMLR.

Acar, D. A. E., Zhu, R., and Saligrama, V. (2021c). Memory efficient online meta learning. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.

Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. (2021). Personalized federated learning with gaussian processes. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8392–8406. Curran Associates, Inc.

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. (2017). Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720.

Antoniou, A., Edwards, H., and Storkey, A. (2018). How to train your maml. *arXiv preprint arXiv:1810.09502*.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. (2022). Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10925–10934.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. (2017). Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2018). LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., and Wallace, E. (2023). Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*.

Chen, D., Mei, J.-P., Zhang, H., Wang, C., Feng, Y., and Chen, C. (2022). Knowledge distillation with the reused teacher classifier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11933–11942.

Chen, D., Mei, J.-P., Zhang, Y., Wang, C., Wang, Z., Feng, Y., and Chen, C. (2021). Cross-layer distillation with semantic calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7028–7036.

Chen, F., Luo, M., Dong, Z., Li, Z., and He, X. (2018). Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*.

Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

Chen, Z. and Liu, B. (2018). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

Cho, J. H. and Hariharan, B. (2019). On the efficacy of knowledge distillation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4793–4801.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE.

Condat, L., Malinovsky, G., and Richtárik, P. (2020). Distributed proximal splitting algorithms with rates and acceleration. *arXiv preprint arXiv:2010.00952*.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Diao, E., Ding, J., and Tarokh, V. (2021). Hetero{fl}: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*.

Dutta, A., Bergou, E. H., Abdelmoniem, A. M., Ho, C.-Y., Sahu, A. N., Canini, M., and Kalnis, P. (2019). On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. *arXiv preprint arXiv:1911.08250*.

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020a). On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR.

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020b). Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. (2019). Online meta-learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930, Long Beach, California, USA. PMLR.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Gabay, D. and Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Gorbunov, E., Hanzely, F., and Richtárik, P. (2020). A unified theory of sgd: Variance reduction, sampling, quantization and coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 680–690. PMLR.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1311–1320. PMLR.

Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K., and Chintala, S. (2019). Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.

Hacohen, G. and Weinshall, D. (2019). On the power of curriculum learning in training deep networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2535–2544. PMLR.

Halgamuge, M. N., Zukerman, M., Ramamohanarao, K., and Vu, H. L. (2009). An estimation of sensor energy consumption. *Progress in Electromagnetics Research*, 12:259–295.

Hazan, E. (2019). Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*.

Hazan, E., Agarwal, A., and Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192.

Hazan, E., Singh, K., and Zhang, C. (2017). Efficient regret minimization in non-convex games. *arXiv preprint arXiv:1708.00075*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Ho, S., Liu, M., Du, L., Gao, L., and Xiang, Y. (2023). Prototype-guided memory replay for continual learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11.

Hong, M., Luo, Z.-Q., and Razaviyayn, M. (2016). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2022). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169.

Hsu, T. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. (2019). Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv: 1909.12488*.

Jiang, Z., Xu, Y., Xu, H., Wang, Z., Liu, J., Chen, Q., and Qiao, C. (2023). Computation and communication efficient federated learning with adaptive model pruning. *IEEE Transactions on Mobile Computing*, pages 1–18.

Kag, A., Acar, D. A. E., Gangrade, A., and Saligrama, V. (2023). Scaffolding a student to instill knowledge. In *The Eleventh International Conference on Learning Representations*.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. (2019). SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378.

Kaya, Y., Hong, S., and Dumitras, T. (2019). Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Khaled, A., Mishchenko, K., and Richtarik, P. (2020a). Tighter theory for local sgd on identical and heterogeneous data. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pages 4519–4529, Online. PMLR.

Khaled, A., Sebbouh, O., Loizou, N., Gower, R. M., and Richtárik, P. (2020b). Unified analysis of stochastic gradient methods for composite convex and smooth optimization. *arXiv preprint arXiv:2006.11573*.

Kim, M., Yu, S., Kim, S., and Moon, S.-M. (2023). DepthFL : Depthwise federated learning for heterogeneous clients. In *The Eleventh International Conference on Learning Representations*.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *http://www.cs.utoronto.ca/ kriz/learning-features-2009-TR.pdf*.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Latré, B., Braem, B., Moerman, I., Blondia, C., and Demeester, P. (2011). A survey on wireless body area networks. *Wireless networks*, 17(1):1–18.

Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *http://vision.stanford.edu/teaching/cs231n/reports/2015/pdfs/yle_project.pdf*.

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

Lee, C.-Y., Gallagher, P. W., and Tu, Z. (2016). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472. PMLR.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020a). Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020*, pages 429–450.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smithy, V. (2019a). Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1227–1231. IEEE.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2020b). On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. (2019b). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3925–3934. PMLR.

Li, Z., Kovalev, D., Qian, X., and Richtárik, P. (2020c). Acceleration for compressed gradient descent in distributed and federated optimization. *arXiv preprint arXiv:2002.11364*.

Li, Z. and Richtárik, P. (2020). A unified analysis of stochastic gradient methods for nonconvex federated optimization. *arXiv preprint arXiv:2006.07013*.

Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Liang, X., Shen, S., Liu, J., Pan, Z., Chen, E., and Cheng, Y. (2019). Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. (2016). Unifying distillation and privileged information. In *ICLR (Poster)*.

Lu, Y., Wang, H., and Wei, W. (2023). Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062*.

Luo, M., Chen, F., Hu, D., Zhang, Y., Liang, J., and Feng, J. (2021). No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5972–5984. Curran Associates, Inc.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ma, X., Zhang, J., Guo, S., and Xu, W. (2022). Layer-wised model aggregation for personalized federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10092–10101.

Makhdoumi, A. and Ozdaglar, A. (2017). Convergence rate of distributed admm over networks. *IEEE Transactions on Automatic Control*, 62(10):5082–5095.

Malinovsky, G., Kovalev, D., Gasanov, E., Condat, L., and Richtarik, P. (2020). From local sgd to local fixed point methods for federated learning. *arXiv preprint arXiv:2004.01442*.

Marcus, G. F. (2018). Deep learning: A critical appraisal. *ArXiv*, abs/1801.00631.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017a). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282.

McMahan, B., Ramage, D., and Scientists, R. (2017b). Federated learning: Collaborative machine learning without centralized training data. *https://ai.googleblog.com/2017/04/federated-learning-collaborative.html*.

Mishchenko, K., Gorbunov, E., Takáč, M., and Richtárik, P. (2019). Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2017). A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*.

Nesterov, Y., Gasnikov, A., Guminov, S., and Dvurechensky, P. (2020). Primal–dual accelerated gradient methods with small-dimensional relaxation oracle. *Optimization Methods and Software*, pages 1–38.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations*.

Nikolenko, S. I. (2019). Synthetic data for deep learning. *arXiv preprint arXiv: 1909.11512*.

Oh, J., Kim, S., and Yun, S.-Y. (2022). FedBABU: Toward enhanced representation for federated image classification. In *International Conference on Learning Representations*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pathak, R. and Wainwright, M. J. (2020). Fedsplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. (2020). Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Romero, A., Kahou, S. E., Montréal, P., Bengio, Y., Montréal, U. D., Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *in International Conference on Learning Representations (ICLR*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Shakespeare, W. (1994). The complete works of william shakespeare. *http://www.gutenberg.org/files/100/old/1994-01-100.zip.*

Shalev-Shwartz, S. (2007). Online learning: Theory, algorithms, and applications. *https://www.cs.huji.ac.il/ shais/papers/ShalevThesis07.pdf.*

Shalev-Shwartz, S. (2012). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press.

Shamir, O., Srebro, N., and Zhang, T. (2014). Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Stanton, S. D., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. (2021). Does knowledge distillation really work? In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Stich, S. U. (2019). Local SGD converges fast and communicates little. *International Conference on Learning Representations (ICLR)*, page arXiv:1805.09767.

Sun, K. and Sun, A. (2021). Dual descent alm and admm. *https://arxiv.org/abs/2109.13214.*

Thrun, S. and Pratt, L. (2012). *Learning to learn.* Springer Science & Business Media.

Tung, F. and Mori, G. (2019). Similarity-preserving knowledge distillation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1365–1374, Los Alamitos, CA, USA. IEEE Computer Society.

Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548.*

Vapnik, V. and Izmailov, R. (2015). Learning using privileged information: Similarity control and knowledge transfer. *Journal of Machine Learning Research*, 16(61):2023–2049.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638.

VS, V., Poster, D., You, S., Hu, S., and Patel, V. M. (2022). Meta-uda: Unsupervised domain adaptive thermal object detection using meta-learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1412–1423.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., et al. (2021). A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*.

Wightman, R. (2019). Pytorch image models. https://github.com/rwightman/pytorch-image-models.

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Yadav, S. and Yadav, R. S. (2016). A review on energy efficient protocols in wireless sensor networks. *Wireless Networks*, 22(1):335–350.

Yang, H., Zhang, X., Khanduri, P., and Liu, J. (2022). Anarchic federated learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25331–25363. PMLR.

Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., and Hua, X.-s. (2019). Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yao, H., Zhou, Y., Mahdavi, M., Li, Z. J., Socher, R., and Xiong, C. (2020). Online structured meta-learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6779–6790. Curran Associates, Inc.

Yu, T., Geng, X., Finn, C., and Levine, S. (2020). Variable-shot adaptation for online meta-learning. *arXiv preprint arXiv:2012.07769*.

Yuan, H. and Ma, T. (2020). Federated accelerated stochastic gradient descent. *arXiv preprint arXiv:2006.08950*.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*.

Zeng, X. and Martinez, T. R. (2000). Using a neural networks to approximate an ensemble of classifiers. In *Neural Processing Letters*, page 2000.

Zhang, L., Shen, L., Ding, L., Tao, D., and Duan, L.-Y. (2022). Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10174–10183.

Zhang, M., Sapra, K., Fidler, S., Yeung, S., and Alvarez, J. M. (2021a). Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*.

Zhang, Q., Fang, J., Meng, Z., Liang, S., and Yilmaz, E. (2021b). Variational continual bayesian meta-learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24556–24568. Curran Associates, Inc.

Zhang, X., Hong, M., Dhople, S., Yin, W., and Liu, Y. (2020). Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data. *arXiv preprint arXiv:2005.11418*.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zhong, Z., Wang, J., Bao, W., Zhou, J., Zhu, X., and Zhang, X. (2022). Semi-hfl: semi-supervised federated learning for heterogeneous devices. *Complex & Intelligent Systems*, 9:1995 – 2017.

Zhou, T., Wang, S., and Bilmes, J. (2020). Curriculum learning by dynamic instance hardness. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8602–8613. Curran Associates, Inc.

Zhuang, Z., Wang, Y., Yu, K., and Lu, S. (2020). No-regret non-convex online meta-learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3942–3946. IEEE.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 928–936. AAAI Press.

CURRICULUM VITAE

[redacted]

[redacted]

[redacted]