UNIFYING DISTILLATION WITH PERSONALIZATION IN FEDERATED LEARNING

by

Siddharth Divi

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer Science

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Z. Berkay Celik, Chair
School of Computer Science

Dr. Kamyar Azizzadenesheli
School of Computer Science

Dr. Ming Yin
School of Computer Science

Approved by:
Dr. Kihong Park

To my family, friends, and collaborators, this would not have been possible without your support.

ACKNOWLEDGMENTS

Portions of this thesis have been submitted for review.

I received a lot of support and guidance throughout this work.

Firstly, I'd like to thank my advisor Prof. Berkay Celik, who was pivotal in this work, and with whom I've had countless discussions on this work. Your constant motivation provided the impetus for me to keep marching on.

I'd also like to thank my committee members, Prof. Kamyar Azizzadenesheli and Prof. Ming Yin, for their insightful feedback on the work, which has opened up exciting directions for future work.

I'd also like to thank Yi-Shan, Habiba, and all my other lab members for the insightful discussions that I've had with them and their help in writing this thesis.

For helping me to get away from work and relax, I'd also like to thank all my friends, both here in West Lafayette and the US and back home in India.

Finally, I would like to thank my family and relatives for their counsel, guidance, and being there for me when it mattered the most.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\mathtt{f}_{\mathtt{large}}$	Teacher model in the distillation process
$\mathtt{f}_{\mathtt{small}}$	Student model in the distillation process
$\mathtt{F}_{\mathtt{s}}$	Hypothesis space of models for the student model
$\mathtt{n}$	Number of data samples
λ	Imitation parameter
$\mathcal{L}$	Generic loss function
$(\mathtt{x}_{\mathtt{i}}, \mathtt{y}_{\mathtt{i}})$	Data sample $\mathtt{i}$ and its associated ground truth label
$\mathtt{s}_{\mathtt{i}}$	Soft-labels of the teacher model for a given data sample $\mathtt{x}_{\mathtt{i}}$
$\mathtt{T}$	Temperature parameter
$\mathtt{O}_{\mathtt{k}}$	Optimal teacher model for user $\mathtt{k}$
$\lvert\mathtt{G}\rvert_{\mathtt{E}}$	`FedAvg` model learned in each global aggregation round $\mathtt{e}$
$\mathcal{L}_{cross}$	Cross-entropy loss function
σ	Softmax function
$(\mathtt{x}_{\mathtt{k}}^{\mathtt{val}}, \mathtt{y}_{\mathtt{k}}^{\mathtt{val}})$	Validation data samples for user $\mathtt{k}$ and the associated ground truth labels
$\mathtt{K}$	Number of clients present during the training phase
$\mathtt{E}_{\mathtt{G}}$	Number of global communication rounds
$\mathtt{l}_{\mathtt{k}}$	Loss of $\mathtt{O}_{\mathtt{k}}$ on the user $\mathtt{k}$'s validation data
$\mathtt{B}$	Mini-batch size
$\mathtt{E}_{\mathtt{L}}$	Number of local epochs
$\mathtt{A}_{\mathtt{k}}$	Local personalized model of user $\mathtt{k}$
$\mathcal{KL}$	Kullback-Leibler divergence
$\lvert\mathtt{T}\rvert_{\mathtt{C}}$	Class complexity measure of the search space of the temperature parameter
$\lvert\lambda\rvert_{\mathtt{C}}$	Class complexity measure of the search space of the imitation parameter.
η	Learning rate
$\mathtt{w}$	Model weights
$\nabla\mathcal{L}(.)$	Gradient of the loss-function

ABBREVIATIONS

`FL`	Federated Learning
`FedAvg`	Federated Averaging
`pFedMe`	Personalized Federated Learning with Moreau Envelopes
`Per-FedAvg`	Personalized Federated Learning: A Meta-Learning Approach
`FedPer`	Federated Learning with Personalization Layers
`non-IID`	non-Identical and Independently Distributed
`MAML`	Model Agnostic Meta-Learning
`MOCHA`	A Framework for Federated Multi-Task Learning
`APFL`	Adaptive Personalized Federated Learning
`LotteryFL`	LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets
`FedBE`	Making Bayesian Model Ensemble Applicable to Federated Learning
`FedMD`	Heterogenous Federated Learning via Model Distillation
`LUPI`	Learning under privileged information
`DS`	Data-split strategy
`CNN`	Convolutional Neural network
`FC`	Fully connected (dense) layers
`ReLu`	Rectified linear unit
`DNN`	Deep Neural network
`SD`	Standard Deviation
`PersFL-GD`	A variant of PersFL
`QoI`	Quantum of Improvement
`PUI`	Percentage of user models improved
`PUD`	Percentage of user models decreased
`API`	Average percentage of user models improved
`APD`	Average percentage of user models decreased
`MPI`	Median percentage of user models improved
`MPD`	Median percentage of user models decreased

`AV`	Average Variance
`CS`	Cosine Similarity
`JI`	Jain’s Index

GLOSSARY

`Federated Learning`	Decentralized privacy-preserving machine learning technique
`Federated Averaging`	An algorithm to learn a common model in the federated learning framework
`PersFL`	An algorithm we propose by unifying distillation with personalization in federated learning
`pFedMe`	A personalization algorithm based on model regularization
`Per-FedAvg`	A personalization algorithm based on meta-learning, a type of local fine-tuning technique
`FedPer`	A personalization algorithm based on dividing the deep network into base and personalization layers
`Distillation`	A machines-teaching-machines paradigm
`MAML`	An algorithm to learn a common initialization point across all the tasks involved
`MOCHA`	A personalization algorithm in FL based on the idea of multi-task learning
`APFL`	A personalization algorithm in FL based on model interpolation, i.e., a mixture of the local and the global models
`FedBE`	An algorithm that introduces the application of Bayesian model ensembles to the federated learning framework
`FedMD`	An algorithm that introduces the idea of heterogeneous model distillation in federated learning
`LUPI`	Learning Using Privileged Information, a framework to learn from privileged information, i.e., information that is available during the training phase but not during the inference phase

ABSTRACT

Federated learning (FL) is a decentralized privacy-preserving learning technique in which clients learn a joint collaborative model through a central aggregator without sharing their data. In this setting, all clients learn a single common predictor (`FedAvg`), which does not generalize well on each client's local data due to the statistical data heterogeneity among clients. In this paper, we address this problem with PersFL, a discrete two-stage personalized learning algorithm. In the first stage, PersFL finds the optimal teacher model of each client during the FL training phase. In the second stage, PersFL distills the useful knowledge from optimal teachers into each user's local model. The teacher model provides each client with some rich, high-level representation that a client can easily adapt to its local model, which overcomes the statistical heterogeneity present at different clients. We evaluate PersFL on CIFAR-10 and MNIST datasets using three data-splitting strategies to control the diversity between clients' data distributions.

We empirically show that PersFL outperforms `FedAvg` and three state-of-the-art personalization methods, `pFedMe`, `Per-FedAvg` and `FedPer` on majority data-splits with minimal communication cost. Further, we study the performance of PersFL on different distillation objectives, how this performance is affected by the equitable notion of fairness among clients, and the number of required communication rounds. We also build an evaluation framework with the following modules: `Data Generator`, `Federated Model Generation`, and `Evaluation Metrics`. We introduce new metrics for the domain of personalized `FL`, and split these metrics into two perspectives: `Performance`, and `Fairness`. We analyze the performance of all the personalized algorithms by applying these metrics to answer the following questions: *Which personalization algorithm performs the best in terms of accuracy across all the users?*, and *Which personalization algorithm is the fairest amongst all of them?* Finally, we make the code for this work available at https://tinyurl.com/1hp9ywfa for public use and validation.

1. INTRODUCTION

Federated Learning (FL) is a distributed collaborative learning paradigm that does not require centralized data storage in a single location. Instead, a joint global predictor is learned jointly by a network of participating users [1]. This paradigm is useful when the clients have private data that they cannot share with the participating entities due to privacy concerns. Recently, FL has found widespread applications in domains ranging from healthcare, finance to predictive keyboards. However, in spite of its widespread applications, `FL` faces different challenges such as expensive communication, systems heterogeneity, statistical heterogeneity, and privacy concerns [2]. Among these, statistical data heterogeneity has recently gained attraction, which means that clients' data are unbalanced and non-identical and independently distributed (non-IID). These challenges are discussed in more detail below.

Expensive Communication. In the `FL` setting, there are potentially a massive number of devices present in the network, and as a result the network communication in comparison to the local computation can be slower by many orders of magnitude. Therefore, it becomes imperative to develop communication-efficient methods wherein the communications are composed of small payloads as opposed to sharing the entire data over the network.

Systems Heterogeneity. Since there are a massive number of devices present in the `FL` setting, these devices may differ in a variety of aspects such as hardware, network connectivity and power ratings, etc. In addition to this, only a small fraction of all devices may be active at any given point of time. It is also a common assumption in such settings that each device may be unreliable, and in some cases an active device can even drop out of the system at any given point of time say due to the device's battery being dead or even the device being out of network coverage area, etc. Such characteristics of the device are what lead to the issue of systems heterogeneity.

Privacy Concerns. In `FL`, sharing the model updates over each global aggregation round rather than sharing the raw data leads to a step towards protecting the user data. However, several works have shown that it is possible to extract sensitive information from such updates communicated in the network, either to the central server or to a third party. Recently, techniques such as differential privacy are being used to ensure privacy in `FL`. However, these privacy guarantees come at the cost of reduced system performance. Hence, understanding these trade-offs theoretically and empirically becomes very important to realizing `FL` systems with privacy.

Statistical Heterogeneity. The data in `FL` is distributed across millions of users. Each user has their own characteristics when it comes to device usage. The number of data points per user can also vary significantly, and in addition to this, there may be an underlying structure in the global data distribution that captures the relationship amongst devices and the local device data distributions. This is in violation of the frequently-used I.I.D assumption in centralized machine learning as well as distributed optimization. This type of a heterogeneity leads to an increased complexity in terms of modeling, analysis, and evaluation.

Thus, the global model trained on clients' non-IID data restricts it from delivering good generalization on each client's local data. Each client gets a common model, irrespective of their data distribution. For instance, consider the next word prediction engine that outputs what word comes next suggestions on a smartphone that enables users to express themselves faster. A common model learned collaboratively among clients fails to give each user useful suggestions, particularly when they have a unique way of expressing themselves in mobile applications such as in writing texts or emails. On the other hand, learning without client collaboration leads to a poor generalization of local clients due to a lack of data. Personalized learning schemes proposed for FL aim to address this problem by finding a personalized model for each client that benefits from other clients' data while overcoming the statistical heterogeneity problem.

Personalization Approaches. There have been a few different approaches that proposed learning schemes through meta-learning, local fine-tuning, multi-task learning, model regularization, contextualization, and model interpolation to build personalized models. For instance, `Per-FedAvg` [3] uses Meta-Learning to learn a common initialization point for all the users, which is then adapted to each user with a couple of steps of gradient descent. Another approach, `pFedMe` [4], re-formulates the FL objective as a bi-level optimization problem and modifies the minimized loss function with the inclusion of a regularization term. Lastly, `FedPer` [5] splits a deep neural network into base and personalization layers, where the base layers are learned collaboratively, and personalization layers are specific to each user. However, some of these approaches incur high computational and algorithmic complexity. For instance, `pFedMe` and `Per-FedAvg` require a higher number of global communication rounds compared to `FedPer`. Model Agnostic Meta-Learning based methods (`MAML`) [6] (used in `Per-FedAvg`) require the computation of the Hessian matrix, which significantly adds computational complexity to each client.

Evaluation in `FL`. In [7], the authors introduce an evaluation framework for large-scale `FL`. They introduce several evaluation metrics such as the number of network nodes, the size of the datasets, the number of communication rounds, etc. Though these metrics are also relevant for personalized FL, they do not cover aspects such as measures of performance and fairness across the users, and more importantly do not answer the following questions: *Which personalization algorithm performs the best in terms of accuracy across all the users?*, and *Which personalization algorithm is the fairest amongst all of them?*

Contributions. In this paper, we introduce PersFL, a new discrete two-stage personalization algorithm, which distills each client's optimal teacher model into each client's local model. In the first stage, each client participates in the FL training and sets the global model across all the aggregation rounds that gives the least error as an optimal teacher. Teacher models contain useful information unique to each client that can be readily adapted into the local models. At the end of the first stage, each client obtains a separate teacher model and proceeds to the second stage. In the second stage, each client distills the information from the optimal teacher model into their local model to learn a personalized model. PersFL controls the trade-off between optimal teacher and local model with temperature parameter that scales the class probability predictions from the teacher, and imitation parameter that balances how much a client imitates the teacher. At the end of PersFL algorithm, each client would have trained a local model based on their dataset and the useful knowledge extracted from other clients' datasets.

We empirically demonstrate the effectiveness of PersFL using CIFAR-10 and MNIST datasets, which are widely used to evaluate personalized models' performance. We compare

PersFL with `FedAvg` and three recent approaches for personalization in FL, a transfer-learning based algorithm (`FedPer`), a bi-level optimization based algorithm (`pFedMe`), and a meta-learning based algorithm (`Per-FedAvg`). To have a fair comparison, we use three different data-splitting strategies to control how each client's local dataset differs from other clients' data.

Our extensive experiments demonstrate that PersFL outperforms `FedAvg`, and outperforms or yields comparable results with the `FedPer`, `pFedMe` and `Per-FedAvg` in local accuracy. For example, compared to the best performing methods on CIFAR-10 data-splits, PersFL improves the accuracy of users on average by `4.7`%, `0.6`% and `3.9`% over `FedPer`, `pFedMe`, and `FedPer`, respectively. We perform additional experiments to characterize the equitable notion of fairness–the deviation among per-user accuracy, study the performance of variants of distillation objectives and investigate the number of communication rounds for convergence. For instance, PersFL reduces the deviation of per-user accuracy distributions on average by `1.5x` and `1.67x` compared to `Per-FedAvg` and `FedPer` algorithms on two different data-splits on MNIST. For the number of global communication rounds, PersFL takes `0.03x` and `0.5x` less communication rounds than `pFedMe` and `FedPer` on two different data-splits on CIFAR-10. We show that these results challenge the existing objectives of personalized learning schemes and motivate new problems in personalization for the research community.

We also introduce a framework for the domain of personalized `FL`. We have three modules part of this framework, namely: `Data-Generator`, `Federated Model Generation`, and `Evaluation Metrics`, respectively. The `data generator's` responsibilities are two-fold: selecting the dataset based on the user's choice and the subsequent generation of a data split for this chosen dataset. The `federated model generation` module is responsible for training the global `FedAvg` model and personalized algorithms. Lastly, the `evaluation metrics` module applies the metrics that we introduce to answer the following questions: (1) *Which personalization algorithm performs the best in terms of accuracy across all the users?*, and (2) *Which personalization algorithm is the fairest amongst all of them?*

We introduce new metrics in addition to the average accuracy used in the evaluation of personalized `FL` models. We split these metrics into two *perspectives* based on their utility:

`Performance`, and `Fairness`. We employ these metrics to find the best performing algorithm and the fairest solutions across different data splits of the CIFAR-10 dataset as well as provide an analysis of the performance and fairness of the personalized `FL` algorithms. Interestingly, we note that the best performing algorithm need not necessarily be the fairest algorithm as well. There can be two different algorithms for each of these *perspectives*, respectively, and we show that this is indeed the case on **DS-2** of CIFAR-10.

2. RELATED WORK

Several prior works have explored techniques for personalization in FL instead of using a common model for all users. These works can be broadly grouped into the following categories based on the techniques adapted to improve clients' model performance, such as local fine-tuning, multi-task learning, contextualization, model regularization based personalization, and model interpolation-based personalization. Table 2.1 shows the surveyed works classified into the different categories mentioned previously. Below, we review these approaches in more detail.

2.1 Local Fine-tuning

This is the most widely used form of personalization, where each user receives a copy of the global `FedAvg` model. Then they adapt it to their local data distribution by taking a couple of steps of gradient descent using their local data. This kind of adaptation is employed in gradient-based meta-learning methods, transfer learning [8] and domain-adaptation [9] methods. Model agnostic meta-learning (`MAML`) has an adaptation phase, wherein the common initialization point is adapted to each task by taking a couple of steps of gradient descent. This adaptation ensures that we find a reasonably good set of parameters for a particular task.

`Per-FedAvg` uses `MAML` to learn a common initialization point for each user during the training phase, which is then subsequently adapted to each user's local data distribution. This local adaptation to each user is what causes the models learned this way to be personalized to each user. In `Personalized FedAvg` [10], the authors opine that the following objectives: *improved personalized performance*, *solid initial model*, and *fast convergence*, must all be addressed simultaneously. They also interpret `FedAvg` as a meta-learning algorithm, and combine `FedAvg` and a meta-learning algorithm `Reptile` [11], to personalize the local models of each user.

In `FedMeta` [12], the authors combine meta-learning, more specifically `MAML` and `Meta-SGD` [13] with `FedAvg` to learn a common parameterized algorithm (a meta-learner) across the users. In `FedML` [14], the authors propose a personalized algorithm in FL using meta-

Table 2.1. The evaluation analysis of studied related Personalized FL methods.

#	Method	Only `FL` metrics†	Fairness Analysis	Datasets‡	Use of Custom Datasplit*	Comparison
`LOCAL FINE-TUNING`						
1	**APFL** [15]	✓	✗	(1), (2), (3), (6)	✓	FedAvg, SCAFFOLD, Per-FedAvg, pFedMe
2	**pFedMe** [4]	✓	✗	(1), (6)	✓	FedAvg, Per-FedAvg
3	**Per-FedAvg** [3]	✓	✗	(1), (3)	✓	FedAvg
4	**FedPer** [5]	✓	✗	(3), (4), (5)	✓	FedAvg
5	**Three Approaches for Personalization** [16]	✓	✗	(1), (2)	✓	FedAvg, AGNOSTIC
6	**Personalized FedAvg** [10]	✓	✗	(2), (7)	✓	FedAvg
7	**FedMeta** [12]	✓	✓	(7), (8), (9), (10)	✓	FedAvg
`MULTI-TASK LEARNING`						
8	**MOCHA** [17]	✓	✗	(11), (12), (13)	✓	FedAvg
`CONTEXTUALIZATION`						
9	**LG-FedAvg** [18]	✓	✗	(6), (1), (3), (14)	✓	FedAvg, FEDPROX
`MODEL REGULARIZATION BASED PERSONALIZATION`						
10	**FedAMP** [19]	✓	✗	(1), (15), (2), (4)	✓	SCAFFOLD, APFL, FedAvg, FEDPROX
11	**FML** [20]	✓	✗	(1), (3), (4)	✓	FedAvg, FEDPROX
`MODEL INTERPOLATION BASED PERSONALIZATION`						
12	**LotteryFL** [21]	✓	✗	(1), (3), (2)	✓	FedAvg, LG-FedAvg

† Whether the personalization method only reports metrics in the `FL` domain.
‡ (1) MNIST, (2) EMNIST, (3) CIFAR-10, (4) CIFAR-100, (5) FLICKR-AES, (6) Synthetic, (7) Shakespeare, (8) FEMNIST, (9) Sentiment 140, (10) Industrial recommendation task, (11) Google Glass (GLEAM), (12) Human Activity Recognition (HAR), (13) Vehicle Sensor, (14) Mobile Assessment for Prediction of Suicide (MAPS), (15) FMNIST.
∗ Whether the personalization method uses a custom datasplit technique.

learning, and the formulation of the problem is very similar to that studied in `Per-FedAvg`, with the only difference being that they perform the study only for strongly convex functions. In `Per-FedAvg`, the authors study non-convex functions and address the issue of gradient stochasticity.

The following methods use non-`MAML` based meta-learning techniques to achieve personalization in FL. Average Regret-Upper-Bound Analysis (`ARUBA`) [22] is a meta-learning algorithm that is inspired from online convex optimization and improves the performance of `FedAvg` when applied to it. In Differentially-Private Gradient-Based Meta-Learning (`DP-GBML`) [23], the authors use a similar idea to design personalized algorithms combining `FedAvg` and meta-learning, with the application of differential privacy [24].

In `FedPer` [5], the authors view a deep network as a combination of base and personalization layers, with the base layers being learned collaboratively and the personalized layers being specific to each user. In q-Fair Federated Learning `(q-FFL)` [25], the authors study a different combination of a `MAML`-type method combined with the FL architecture from an empirical point of view. They introduce a novel optimization objective that is inspired by fair resource allocation from wireless networks. The formulation of this optimization objective encourages a more uniform (and thereby fair) distribution of accuracy across users in the federated environment.

2.2 Multi-task Learning

Another way of looking at the problem of personalization in FL is from the perspective of a multi-task learning (`MTL`) setting. In `MOCHA` [17], each client is analogous to a task. Therefore the adaptation to a task is analogous to personalizing to each user in the FL setting. It also requires all the clients to be online during the training phase. The connections between FL and `MTL` and meta-learning are elaborated in [2], [26]. Another approach that is also an open problem, and is discussed in [27], is to cluster clients (users) based on features such as region or other similar characteristics as similar tasks. work [16].

2.3 Contextualization

Contextualization is a closely related notion to personalization, in the sense that the model learned in the federated setting should be able to work under different contexts. For example, the context can differ between users as their characteristics differ. On the other hand, this can be true even in a single user's case, i.e., the ability of a single model on a user to work under different contexts. This problem is studied in the form of the next character recognition task [28]. To obtain any solution in such a setting, we would need access to features about the context during the training phase. In fact, evaluating models proposed in such settings has been analyzed in a closely related work [29]. In this work, the authors propose an evaluation technique to measure the extent of on-device personalization. They also explore the conditions under which personalization yields desirable models. In Local-Global Federated Averaging (`LG Fed-Avg`) [18], the authors propose to learn compact local representations on each client and a global model across all the clients, i.e., an ensemble of local and global models. Models learned this way are better at dealing with heterogeneous data and learning fair representations effectively, thereby obfuscating protected attributes.

2.4 Model Regularization-based Personalization

Each user's model can be personalized by employing the regularization of the differences between the global and the local models. `pFedMe` [4] uses Moreau envelopes [30] as a regularization term to learn personalized models and the global FL model parallelly. Clients can pursue their models in different directions, albeit staying not too far away from the global `FedAvg` model ($\mathtt{w}$). They compare their method to `Per-FedAvg`, and claim that their method is better as they directly optimize for $\mathtt{f_i}(.)$ (loss function at client $\mathtt{i}$) and also that `Per-FedAvg` requires the computation or the estimation of the Hessian matrix whereas this is not the case in `pFedMe`.

In `FedAMP` [19], the authors propose a personalized method employing federated attentive message passing to facilitate similar clients to collaborate more. They achieve this by having a regularization term in the FL objective. The regularization term improves client collaborations' effectiveness through an attention-inducing function $\mathtt{A}(.)$. In a closely related work [31], the authors add a regularization term to the global FL objective. This regularization term is similar to an L2-penalty term on the distances between the local and the global models. They also use a mixing parameter λ to control the optimization degree of the local and the global models.

Federated Mutual Learning (`FML`) [20] uses the non-IID nature of the data as a feature to learn more personalized models for each user. In this work, the authors combine the idea of Knowledge Distillation [32] and Deep Mutual Learning (`DML`) [33]. They apply regularization in the form of $\mathcal{KL}$-divergence between the local models' predictions and the global model to achieve personalization.

2.5 Model Interpolation-based Personalization

Another class of personalization techniques that, broadly speaking, focus on the mixture of the local and the global models. In Adaptive Personalized Federated Learning (`APFL`) [15], the authors try to address the following question: *what degree of personalization is best for each client?* They learn a personalized model for each client, a mixture of the local and global models. The optimal mixing parameter, which controls the local and the global models' ratio, is integrated into the learning problem. In a recent work [16] the authors propose the following three different approaches with generalization guarantees for personalization in FL: *user clustering*, *data interpolation*, and *model interpolation*.

In *user clustering*, the clients are clustered into groups, and subsequently, a model is trained for each group. Thus, the model is analogous to one that is between the purely local models and the purely global model. This introduces a trade-off between the generalization capability of the model learned and the distribution divergence. In *data interpolation*, the authors try to answer the following question: *how to use the auxiliary data (i.e., the global data distribution) to improve the accuracy of the model learned on the local data distribution?* They relate this problem to domain adaptation. In the last approach *model interpolation*, the authors propose to learn a personalized model that is a mixture of the local and the global models. This formulation is very similar to the problem formulation of `APFL`. Out of these three, the first two approaches are not suitable for FL since they require meta-feature information from the clients, which raises privacy concerns.

In `LotteryFL` [21], the authors adopt a Lottery Ticket Network (`LTN`) through the application of the Lottery Ticket Hypothesis (`LTH`) [34] to learn personalized models for each user. Each client learns a `LTN` by pruning the base model parameters on their local data. Rather than communicating the entire base model, only the parameters of the `LTN`s of each user are communicated between the clients and the server. Subsequently, the server then aggregates over all these `LTN`s, and the updated parameters are sent back to the clients. The clients then repeat the same process of learning an `LTN`. This process continues until convergence.

3. PRELIMINARIES

3.1 Federated Averaging

`Federated Averaging` (`FedAvg`) is an algorithm in which $\mathtt{n}$ users along with a central global aggregator participate together to learn a joint collaborative (global) model. The data of each user does not ever leave their device. The users train the shared model on their local data and then share their model weights to the central aggregator. The central aggregator then aggregates (averages) the model updates from all the participating users and shares the new global model's updated weights. This process continues till convergence. In the end, each user obtains the same global model. The server aggregation over all the received client models is shown in Equation 3.1.

$$\mathtt{w}_{\mathtt{t+1}} \leftarrow \sum_{\mathtt{k=1}}^{\mathtt{K}} \frac{\mathtt{n}_{\mathtt{k}}}{\mathtt{n}} \mathtt{w}_{\mathtt{t}}^{\mathtt{k}} \tag{3.1}$$

In Equation 3.1, $\mathtt{w}_{\mathtt{t+1}}$ refers to the global model being learned in the global aggregation round $\mathtt{t+1}$, $\mathtt{K}$ is the number of users indexed by $\mathtt{k}$, $\mathtt{n}_{\mathtt{k}}$ is the number of training data samples that user $\mathtt{k}$ has, and $\mathtt{n}$ is the number of training data samples across all the $\mathtt{K}$ users, and $\mathtt{w}_{\mathtt{t}}^{\mathtt{k}}$ refers to the global model adapted to the user $\mathtt{k}$'s local training data distribution in the global aggregation round $\mathtt{t}$.

3.2 Distillation

Model compression [35] or distillation [32] are techniques to reduce the size and complexity of machine learning models. Distillation compresses a large complex model or an ensemble of models $\mathtt{f}_{\text{large}}(\mathtt{x})$ (teacher model) into a smaller and less complex model $\mathtt{f}_{\text{small}}(\mathtt{x})$ (student model), which mimics the predictions of the complex model. There are scenarios in which the teacher model and ensemble models are too complicated from a computational perspective. Thus, distilling a large model into a simpler model makes it easier to run on limited computational resources such as on edge and mobile devices. Remarkably, model distillation achieves model compression with no or minimal loss in accuracy.

Put in math, given a large model $\mathtt{f}_{\text{large}}$ that has been already learned, a small model $\mathtt{f}_{\text{small}}$ is learned by minimizing

$$\mathtt{f}_{\mathtt{s}} = \mathop{\arg\min}_{\mathtt{f}_{\mathtt{small}} \in \mathtt{F}_{\mathtt{s}}} \frac{\mathtt{1}}{\mathtt{n}} \sum_{\mathtt{i}=\mathtt{1}}^{\mathtt{n}} [(\mathtt{1} - \lambda)\mathcal{L}(\{(\mathtt{x}_{\mathtt{i}}, \mathtt{y}_{\mathtt{i}})\}_{\mathtt{i}=\mathtt{1}}^{\mathtt{n}}, \mathtt{f}_{\text{small}}) + \lambda\mathcal{L}(\{\mathtt{x}_{\mathtt{i}}, \mathtt{s}_{\mathtt{i}}\}_{\mathtt{i}=\mathtt{1}}^{\mathtt{n}}, \mathtt{f}_{\text{small}})] \tag{3.2}$$

where $\mathtt{f}_{\mathtt{small}}$ is the candidate student model from the class capacity measure of the student model $\mathtt{F}_{\mathtt{s}}$ (i.e., hypothesis space of models for the student model), $\mathtt{f}_{\mathtt{s}}$ is the optimal student model learned, and $\lambda \in [\mathtt{0}, \mathtt{1}]$ is the imitation parameter which controls the extent to which $\mathtt{f}_{\mathtt{small}}$ imitates $\mathtt{f}_{\mathtt{large}}$ compared to directly learning from the data.

In Equation 3.2, there are two datasets on which $\mathtt{f}_{\mathtt{small}}$ is trained, $(\mathtt{x}_{\mathtt{i}}, \mathtt{y}_{\mathtt{i}})_{\mathtt{i}=\mathtt{1}}^{\mathtt{n}}$ and $(\mathtt{x}_{\mathtt{i}}, \mathtt{s}_{\mathtt{i}})_{\mathtt{i}=\mathtt{1}}^{\mathtt{n}}$. $\mathtt{s}_{\mathtt{i}}$ is called the soft label of the teacher ($\mathtt{f}_{\mathtt{large}}$) model computed as $\mathtt{s}_{\mathtt{i}} = \mathtt{f}_{\mathtt{large}}(\mathtt{x}_{\mathtt{i}})/\mathtt{T}$ and $\mathtt{y}_{\mathtt{i}}$ is the ground truth label. $\mathtt{T} > \mathtt{0}$ is called the temperature parameter, which softens the teacher model's class probabilities. The softened class-probability predictions reveal dependencies among the labels that are otherwise hidden as either extremely small or large numbers.

4. PROPOSED PERSONALIZATION ALGORITHM

We introduce PersFL, a new personalization algorithm, which unifies distillation with personalization to improve the generalization of each user's model accuracy instead of using the same global `FedAvg` model for each user (Section 4.1). Subsequently, in Section 4.2 we provide an intuition into why PersFL works, and in Section 4.3 perform a convergence and complexity analysis of PersFL.

4.1 PersFL Algorithm

Our idea of distillation for personalization is inspired by the discrete phase local adaptation technique called the greedy local fine-tuning method [15]. In this approach, a global `FedAvg` model is first learned during the training phase. During the subsequent adaptation phase, users perform several gradient descent steps to adapt the global `FedAvg` model's weights to users' local data distribution. However, a crucial question that needs to be answered is: *why do all users adapt the same global model when the goal is personalization for each user?* This question inspires us to integrate a separate optimal teacher model into the local model of each user. To obtain the optimal teacher, each user iteratively validates the global `FedAvg` model on their local data during each aggregation round of the FL training phase. Subsequently, each user identifies the best global `FedAvg` model as an optimal teacher based on the accuracy. Each user then incorporates the optimal teacher into their local model through distillation. Algorithm 1 details the steps of PersFL, which is a discrete two-stage algorithm.

Stage-1: Finding the Optimal Teacher Models. In the first stage, each user participates in the training phase of FL and receives the current copy of the global model (`FedAvg`) in each round. In this step, each user locally stores the global model's current copy before updating the local version of the global model and sending it to the server for aggregation.

Algorithm 1 PersFL Algorithm

```
Stage-1: Finding the optimal teacher models
Notation:
K: Clients indexed by k, $E_G$: Number of global aggregation rounds, $\mathcal{G}_e$: Global FedAvg model during the global aggregation round e, $O_k$: Optimal teacher model for user k, $l_k$: Loss of $O_k$ on the user k's validation data (initialized to an arbitrarily large value), $x_k^{val}$: Validation data of user k.
for global aggregation round $e = 1$ to $E_G$ do
    for user $k = 1$ to K do
        $l_k \leftarrow \mathcal{L}_{cross}(\sigma(\mathcal{G}_e(x_k^{val})), y_k^{val})$
        if $l_k < l_k$ then
            $l_k \leftarrow l_k$
            $O_k \leftarrow \mathcal{G}_e$
        end if
    end for
end for
```

```
Stage-2: Distilling from the optimal teacher models
Notation:
K: Clients indexed by k, B: Local mini-batch size, $E_L$: Number of local epochs, T: Temperature parameter, $\lambda$: Imitation parameter, $O_k$: Optimal teacher model for user k, $A_k$: Local personalized model of user k, $|T|_c$: Search space of temperature values, $|\boldsymbol{\lambda}|_c$: Search space of imitation parameter values.
Client Side:
  • For each client k, initialize $A_k$ with the weights of $O_k$.
  • Find the optimal values of $\lambda$ and T by performing the following steps.
for user $k = 1$ to K do
    $B \leftarrow$ Obtain mini-batches of data from the user k's training data
    for local epoch $i = 1$ to $E_L$ do
        for batch $b \in B$ do
            $\mathcal{L}(A_k, b) \leftarrow (1 - \lambda) * hardLoss + (\lambda T^2) * softLoss$
            $hardLoss \leftarrow \mathcal{L}_{cross}(y_b, \sigma(A_k(b)))$
            $softLoss \leftarrow \mathcal{KL}(s_b, \sigma(\frac{A_k(b)}{T}))$
            $s_b \leftarrow \sigma(\frac{O_k(b)}{T})$
            $w \leftarrow w - \eta \nabla \mathcal{L}(A_k, b)$
        end for
    end for
end for
```

After the termination of the FL training phase, each user finds the optimal teacher model $\mathtt{O_k}$ by minimizing:

$$\mathtt{O_k} = \underset{\mathtt{O_{k \in |G|_E}}}{\arg\min}\, \mathcal{L}_{\mathtt{cross}}(\sigma(\mathtt{O_k}(\mathtt{x_k^{val}})), \mathtt{y_k^{val}}) \tag{4.1}$$

where $|\mathtt{G}|_{\mathtt{E}}$ is the `FedAvg` model learned in each global aggregation round, $\mathtt{E}$ is the total number of global aggregation rounds of FL, $\mathtt{x_k^{val}}$ is the validation data of user $\mathtt{k}$ and $\mathtt{y_k^{val}}$ is the ground truth of the validation data of user $\mathtt{k}$. Thus, the optimal teacher model represents the global `FedAvg` model across the aggregation rounds that achieves the minimum loss on the validation data of user $\mathtt{k}$.

Stage-2: Distilling from the Optimal Teacher Models. The second stage takes place locally for each user after the FL training phase, independent of FL (i.e., no client collaboration). We call this stage local adaptation. We first initialize the personalized model, $\mathtt{A_k}$ with the weights of $\mathtt{O_k}$ for each user $\mathtt{k}$. Following this, each user distills the information from the optimal teacher ($\mathtt{O_k}$) into their local model to learn $\mathtt{A_k}$. Specifically, each user computes $\mathtt{A_k}$ by distilling hard-loss and soft-loss by minimizing:

$$\begin{aligned}\mathtt{A_k} = \underset{\substack{\lambda \in |\lambda|_{\mathtt{C}}, \mathtt{T} \in |\mathtt{T}|_{\mathtt{C}} \\ \mathtt{A_k} \in |\mathtt{A_k}|_{\mathtt{C}}}}{\arg\min} & \overbrace{(1-\lambda)(\mathcal{L}_{\mathtt{cross}}(\sigma(\mathtt{A_k}(\mathtt{x_k^{train}})), \mathtt{y_k^{train}}))}^{\text{hard-loss}} \\ & + \overbrace{(\lambda \mathtt{T}^2) \times \mathcal{KL}(\sigma(\frac{\mathtt{A_k}(\mathtt{x_k^{train}})}{\mathtt{T}}), \sigma(\frac{\mathtt{O_k}(\mathtt{x_k^{train}})}{\mathtt{T}}))}^{\text{soft-loss}}\end{aligned} \tag{4.2}$$

The hard-loss refers to the loss of the student model ($\mathtt{A_k}$) on the hard-labels ($\mathtt{y_i}$), whereas soft-loss refers to the loss of the student model on the soft labels ($\mathtt{s_i}$). Soft labels are the teacher model's ($\mathtt{O_k}$) scaled predictions. $|\lambda|_{\mathtt{C}}$ refers to the complexity of the search space of the imitation parameter, $|\mathtt{T}|_{\mathtt{C}}$ denotes the search space complexity of the temperature parameter, and $|\mathtt{A_k}|_{\mathtt{C}}$ refers to the hypothesis space of the personalized models for $\mathtt{k}$. Here, each user performs a grid-based search to find the optimal values for distillation parameters temperature ($\mathtt{T}$) and imitation parameter (λ) while simultaneously learning the personalized model.

We use Kullback-Leibler (KL) divergence between the output of the teacher model ($\mathtt{O_k}$) and the student model ($\mathtt{A_k}$). Enforcing the logits of $\mathtt{O_k}$ and $\mathtt{A_k}$ to be similar yields a regularizing effect, which in turn improves the generalization ability of $\mathtt{A_k}$. We additionally multiply the soft-loss by $\mathtt{T}^2$ since the gradients of the term $\sigma(\mathtt{A_k(b)/T})$ scale as $\mathtt{1/T}^2$. This ensures that the relative contributions of the hard-labels ($\mathtt{y_i}$) and the soft-labels ($\mathtt{s_i}$) are roughly unchanged if the $\mathtt{T}$ value changes.

At the end of the **Stage-2**, each user learns a personalized model optimized for their local data distribution by distilling the useful knowledge from other users into the personalized model $\mathtt{A_k}$ through the optimal teacher $\mathtt{O_k}$ learned in **Stage-1**.

4.2 Why does PersFL work?

We introduce the learning under privileged information (LUPI) paradigm [36] and show that PersFL reduces to Generalized Distillation [37], an instance of the LUPI paradigm. Vapnik's LUPI paradigm assumes that feature-label pairs ($\mathtt{x_i}, \mathtt{y_i}$), and additional information $\mathtt{x}_\mathtt{i}^\star$ about ($\mathtt{x_i}, \mathtt{y_i}$) are available at training time and $\mathtt{x}_\mathtt{i}^\star$ is not available at test time. Here, $\mathtt{x}_\mathtt{i}^\star$ is called the privileged information. For example, consider the problem of identifying cancerous biopsy images. $\mathtt{x}$ is the biopsy image of a patient in pixel space. An oncologist may describe the biopsy image relevant to cancer in a specialized language space different than the pixel space. The descriptions are called privileged information ($\mathtt{x}^\star$), which contain useful information to classify the biopsy images, however this information is not available at test time.

Generalized distillation develops an objective to learn from multiple data representations as follows. Firstly, it learns a teacher model $\mathtt{f_t}$ on the feature-target set $\{\mathtt{x}_\mathtt{i}^\star, \mathtt{y_i}\}_{\mathtt{i}=1}^{\mathtt{n}}$. Second, it computes teacher soft labels $\mathtt{s_i} = \mathtt{f_t}(\mathtt{x}_\mathtt{i}^\star)/\mathtt{T}$ using a temperature parameter $\mathtt{T} > \mathtt{0}$. Lastly, it learns a student model $\mathtt{f_s}$ from $\{\mathtt{x_i}, \mathtt{y_i}\}_{\mathtt{i}=1}^{\mathtt{n}}$, $\{\mathtt{x_i}, \mathtt{s_i}\}_{\mathtt{i}=1}^{\mathtt{n}}$.

In PersFL, the optimal teacher model $\mathtt{O_k}$ of each user $\mathtt{k}$ is analogous to the privileged information $\mathtt{x}^\star$. Since each user's data distributions are not exactly the same, each user's

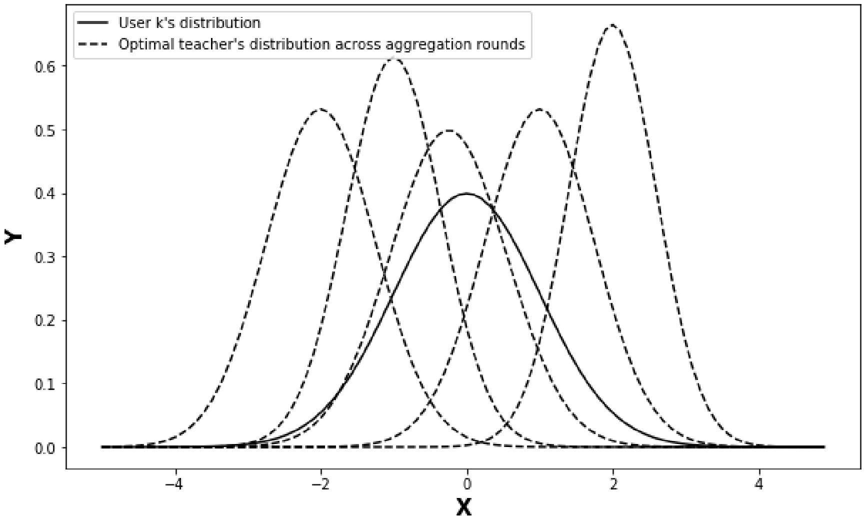


Figure 4.1. Users `k`'s distribution vs. global `FedAvg` model's distribution across the global aggregation rounds.

optimal teacher model is unique and identified with the best performing `FedAvg` model on the user `i`'s validation data during the FL communication rounds. A user can obtain the intricate patterns from other users' large amounts of data through the weights of $\mathtt{O_k}$ as privileged information, only available to each user during FL training.

Once each user identifies the teacher model, the soft-labels of the teacher model ($\mathtt{s_i}$) are computed on the data of each user. PersFL learns the student model from the teacher through distillation by choosing optimal values of the parameters, λ and `T`. We argue that distilling information from the teacher model to the student model overcomes the *catastrophic forgetting* problem [38], which is the tendency of a model to forget the information learned in the previously trained tasks when it is trained on new tasks. PersFL mitigates this problem by first initializing the student model with the teacher model's weights and then distilling the teacher model's information to the student model.

Figure 4.1 shows the training data distribution of a user `k` (solid line) and the distribution of the `FedAvg` model in the global communication rounds (dashed lines). The global model's distribution comes close to approximating the user's distribution and starts moving away from it. This divergence of distributions is caused due to the non-IID data distribution across users. PersFL's teacher model is unique to each user, and there is maximal overlap with their data distribution, which helps address the statistical heterogeneity problem and mitigate its negative transfer effect. Our extensive experiments in Section 6 validate this hypothesis.

4.3 Convergence and Complexity Analysis

Convergence of PersFL. Convergence of an algorithm is defined as the algorithm's ability to converge to the global optimum, which is defined as the region in the loss landscape with the lowest possible loss (global minima). Equation 4.3 represents the core objective of the FL setting. Here, $\mathtt{f_i}(.)$ denotes the loss over user $\mathtt{i}$'s data distribution when there are $\mathtt{N}$ users, and $\mathtt{w}$ refers to the weights of the `FedAvg` model being learned.

$$\min_{\mathtt{w} \in \mathcal{R}^{\mathtt{d}}} \left\{ \mathtt{f}(\mathtt{w}) := \frac{1}{\mathtt{N}} \sum_{\mathtt{i}=1}^{\mathtt{N}} \mathtt{f_i}(\mathtt{w}) \right\} \tag{4.3}$$

Unlike other personalized FL methods such as `pFedMe` and `FedPer`, the convergence analysis for PersFL is not needed since PersFL does not modify the FL core objective. To detail, PersFL is a two-stage discrete algorithm in which the users join the FL training in the first stage to learn a separate optimal teacher model. Each user then independently distills the optimal teacher to their local dataset. Therefore, the convergence of PersFL is the same as the convergence of the `FedAvg` algorithm [39].

Complexity of PersFL. We analyze the complexity of PersFL in terms of the number of epochs executed locally at each user during the training phase. The worst-case complexity of PersFL when all users are computationally involved in every round is given by $\mathcal{O}(\mathtt{E_G E_L} + |\mathtt{T}|_{\mathtt{C}}|\lambda|_{\mathtt{C}}\mathtt{E})$. $\mathtt{E_G}$ is the number of global communication rounds, $\mathtt{E_L}$ is the number of local epochs, $|\mathtt{T}|_{\mathtt{C}}$ and $|\lambda|_{\mathtt{C}}$ are the class complexity measures of the search space and $\mathtt{E}$ is the number of epochs that we set to distill information from the optimal teacher model into the personalized model.

5. EVALUATION METRICS FOR PERSONALIZED FL

5.1 Need for Alternative Metrics in `FL`

From our study of personalization methods in Chapter 2, we identified two problems in the evaluation of personalized models. Below, we provide an example scenario and present the issues through this scenario.

Motivating Example. Consider that `9` users collaboratively learn a common model for the task of the next-character prediction on the keypad of their mobile phones. The dataset is distributed to each user according to a particular data-split strategy to mimic the non-IID nature of the real world's data distributions. We refer to **DS-1**, **DS-2**, and **DS-3** as different strategies for splitting the dataset, and distributing them among all the users. Each user learns a local model, a global (common) model (`FedAvg`), and a personalized model using four different personalization approaches. These personalized algorithms aim to learn a model specific to each user and better than both the local and the global `FedAvg` model. Table 5.1 presents the accuracy of the per-user local models, `FedAvg` model, and four personalized models on the data-splits. With this example scenario, we present the problems below.

Problem 1 - Incomplete Results. The personalization approaches solely report the personalized models' average accuracy across all the users to measure model effectiveness. Turning to Table 5.1, we ask the question, *Which personalization algorithm performs the best in terms of accuracy across all the users?*, and *Which personalization algorithm is the fairest amongst all of them?* To answer this question, the research community uses the average accuracy. In this example, `Alg.4` is relatively better than the other approaches, and it gives the highest average accuracy of `79.3`%. However, upon closer inspection, we observe that with respect to the local/`FedAvg` model, `Alg.4` only leads to an increase in performance for 2 out of 9 users. This observation also points to the lack of fairness evaluation of the personalized methods, despite the tremendous interest in the machine learning community to develop fair methods. We define fairness from an equitable notion, i.e., the concept of users getting similar improvements [25]. An ideal personalization algorithm is one which leads to a similar `QoI` (defined in Section 5.2.3) for all users compared to the local/global `FedAvg` models. Although `Alg.2` is the best performing algorithm, `Alg.1` is the fairest among all

Table 5.1. Example scenario to help motivate the need for alternative metrics in Personalized FL.

	Local Model	**FedAvg**	**Alg.1**	**Alg.2**	**Alg.3**	**Alg.4**
Datasplit(s)	**DS-1**	**DS-1**	**DS-1**	**DS-2**	**DS-3**	**DS-1**
Users						
User0	73%	78%	82%	79%	76%	75%
User1	71%	75%	82%	74%	72%	72%
User2	61%	69%	82%	75%	68%	68%
User3	55%	71%	75%	79%	82%	96%
User4	69%	74%	74%	78%	85%	97%
User5	65%	77%	75%	89%	87%	75%
User6	74%	80%	77%	74%	78%	76%
User7	68%	82%	77%	76%	79%	78%
User8	75%	85%	78%	79%	79%	77%
Avg Acc	**67.89%**	**76.78%**	**78%**	**78.11%**	**78.44%**	**79.33%**

the algorithms. This leads to an interesting observation that an algorithm that is the best performing need not necessarily be the fairest.

Problem 2 - Inconsistent Datasets and Data-splits. A data-splitting strategy is used to split the dataset across participating users such that each of them receives a fraction of the data which is not identically distributed. This is termed as a non-IID distribution which is frequently the case in the real-world. The characteristics of the data in `FL` settings are important because if the data is distributed IID, personalization cannot offer any benefits [15], the global `FedAvg` model would perform the best in this case. On the other hand, if the data is distributed non-IID, then personalization helps combine the local and shared (global) information. We observe in Table 5.1 that all approaches use the same dataset yet the data-splits (abbreviated as **DS-x**) are different. For instance, `Alg.1`, and `Alg.4` are trained on **DS-1**, whereas `Alg.2` is on **DS-2** while `Alg.3` is on **DS-3**. The use of different data-splits on the same dataset makes it hard to compare personalized models. Additionally, we observe that when personalized models are compared with each other, each paper often reports results on a new (different) data-split than the one used in the compared paper. For example, `Alg.1` and `Alg.2` cannot be directly compared as they have been trained on different data-splits, and their performance might be different on another data-split.

Study of Related Works. We have studied 12 recent personalization methods to identify the datasets and data-splits they use, whether they report metrics other than those commonly used in the `FL` settings, which approaches they are compared to, and whether a fairness analysis is performed. Table 2.1 presents our analysis results in five different personalization categories, as elaborated in Chapter 2.

First, we found that all the personalized *FL* approaches either use all or a subset of the metrics used in the `FL` settings, such as *training loss*, *average validation accuracy*, *prediction error*, *number of communication rounds* to measure the communication complexity. This means that none of the surveyed works report a per-user accuracy for the personalization algorithms, as depicted in the *Only `FL` metrics* column of Table 2.1. Additionally, we found that none of the surveyed works except for `FedMeta` perform a fairness analysis. `FedMeta` analyzes the fairness of the algorithm's final performance distribution averaged over multiple runs.

Second, we observe that 9 out of the 12 surveyed works use standard datasets including (1), (2), (3), (4), (6), and (7). The other 3 works use a subset of the following additional datasets: (5), (8), (9), (10), (11), (12), (13), (14), and (15).

Third, we found that all works use different custom data-splits on these datasets. We show this in the *Use of Custom Datasplit* column of Table 2.1. Further, we describe some of the data-splits used in the literature, which we henceforth refer to as **DS-#**, in Section 5.2.1. `FedPer` uses **DS-1**, `pFedMe` uses **DS-3**, `Per-FedAvg` uses **DS-4**, and all the other works use other kinds of data-splits.

Lastly, we found that 8 out of the 12 works compare their algorithm to the base `FedAvg` algorithm, and not other existing personalization algorithms, as depicted in the *Comparison* column of Table 2.1. This makes it difficult to evaluate how different personalization algorithms perform with respect to each other. `APFL`, `pFedMe`, `FedAMP`, and `LotteryFL` compare their method with other existing personalization algorithms whereas the others do not. Even though some works compare their method with other approaches such as AGNOSTIC [40], SCAFFOLD [41], and FEDPROX [42], these are not personalization algorithms in `FL`; hence, they do not fall under the purview of our discussion.

Subsequently, we introduce a framework for personalized `FL`, and also propose new metrics to alleviate the aforementioned problems.

5.2 System Architecture

In this section, we explain the proposed system framework in more detail. Figure 5.1 provides an overview of the system components. Initially, the system selects the dataset and divides it according to the chosen data-split strategy among the chosen number of clients (Section 5.2.1). The system then trains the `FedAvg` model and the other personalization algorithm-based models with the chosen model architecture (Section 5.2.2). Finally, the performance of the trained models from the previous step are evaluated on a defined set of metrics (Section 5.2.3). These metrics are divided into the following two groups: a *performance* perspective and a *fairness* perspective.

5.2.1 Data Generator

The `Data Generator` module is responsible for two tasks: (1) selecting the dataset to be used based on the user's choice and then (2) subsequently generating a data-split on this dataset, also based on the user's choice. The user also provides the number of clients to simulate, and after this process, the data is now split to mimic the `FL` setting.

Datasets. We currently support the CIFAR-10 and MNIST datasets, widely used in `FL` training in the literature. MNIST is a dataset of 28×28 images of handwritten digits from `0-9` consisting of `10` labels and $70,000$ instances. CIFAR-10 is a dataset of 32×32 color images with `10` classes and $60,000$ instances. Other well-known datasets such as CIFAR-100 and Imagenet, and custom datasets can be easily integrated to the framework.

Data-splitting Strategies. The data splitting module supports the following four different data-splitting techniques.

In data-split 1 (**DS-1**), each user has the same total number of samples but may have different classes and a different number of samples per class. The statistical heterogeneity is varied by controlling the parameter `k`, which controls the number of overlapping classes between each user. For example, a highly non-IID partition will arise when $k = 4$ compared to a highly IID partition when $k = 10$. In our experiments, we set `k` to 4 to have non-IID data across users.

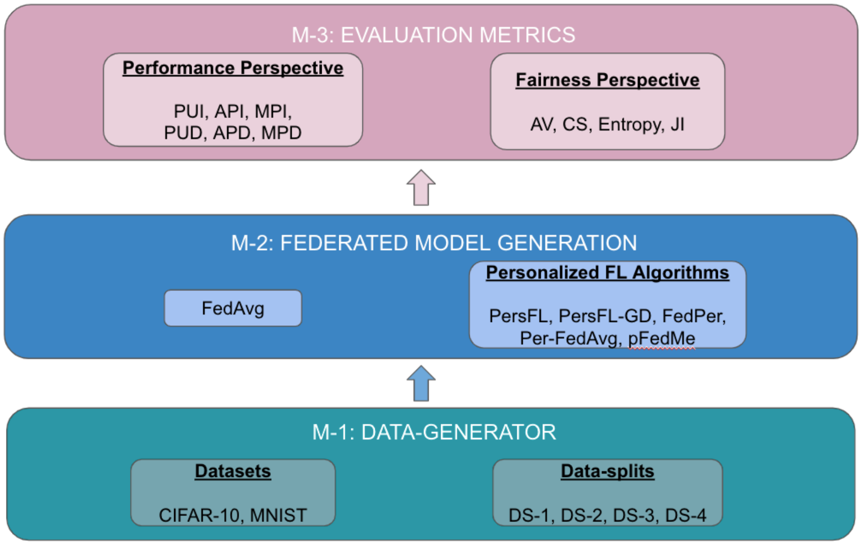


Figure 5.1. System Architecture Diagram.

In **DS-2**, all users have samples from all classes, but the number of samples per class they have is different, and hence the total number of samples per user is also different across users. In order to replicate a non-IID distribution, we assign samples from each class to the users following a Dirichlet distribution with $\alpha = 0.9$, following the previous work [43]. Each class is parameterized by a vector $\mathtt{q}$ where $\mathtt{q} \geq 0, \mathtt{i} \in [1, \mathtt{N}]$, where $\mathtt{q}$ is sampled from a Dirichlet distribution with parameters α and $\mathtt{p}$. The parameter $\mathtt{p}$ refers to the prior class distribution among all the classes, and α refers to the concentration parameter that controls the data similarity among the users. If $\alpha \rightarrow \infty$, all users have an identical distribution to the prior. If $\alpha \rightarrow 0$, each user only has samples from one class randomly chosen.

For **DS-3**, each user has two of the ten class-labels. Additionally, the total number of samples per user is different, i.e., all the users do not have the same number of total samples. The samples assigned to users are drawn from a log-normal distribution with the parameters $\mu = 0$ and $\sigma = 2$. A variable $\mathtt{u}$ has a log-normal distribution if $\mathtt{log}(\mathtt{u})$ is normally distributed. The probability density function for the log-normal distribution is computed as:

$$\mathtt{p}(\mathtt{u}) = \frac{1}{\sigma \mathtt{u} \sqrt{(2\pi)}} \mathtt{e}^{-(\frac{(\ln(\mathtt{u}) - \mu)^2}{2\sigma^2})} \tag{5.1}$$

These parameters correspond to the underlying normal distribution from which we draw the samples.

DS-4 is introduced in `Per-FedAvg` [3]. In this technique, assuming there are `n` users, the data is split as explained below. Half of all the users ($\frac{\texttt{n}}{2}$), have `k` images from each of the first five classes, and the remaining users ($\frac{\texttt{n}}{2}$), have $\frac{\texttt{k}}{2}$ images from only *one* among first five classes and `2k` images from only *one* among the remaining five classes. The parameter `k` is set as $\texttt{k} = \texttt{68}$ for CIFAR-10, and $\texttt{k} = \texttt{196}$ for MNIST.

5.2.2 Federated Model Generation

In this module, the `FedAvg` model and the personalized `FL` algorithms supported (Section 5.2.4) are trained, and their inferences on the test set are obtained for evaluation and comparison.

Model Architectures. Currently, we support the following model architectures commonly used in the literature. For the CIFAR-10 dataset, we use a CNN-based model with two 2-D convolutional layers separated by a MaxPool layer between them and followed by three fully connected (FC) layers. The fully connected layers have 400, 120, and 84 hidden neurons. We use ReLu activations after every layer excluding the last fully connected layer. For the MNIST dataset, the architecture used is a two-layer deep neural network (DNN) with 100 hidden-layer neurons, and with a ReLu activation applied on the hidden layer. The output layer has 10 nodes with a softmax function to get the class probabilities. The architectures we use for both datasets are similar to those in `pFedMe` for conformity. Extensions to support other architectures is straightforward.

5.2.3 Evaluation Metrics

In this module, we introduce new metrics to evaluate the personalized `FL` algorithms from the `performance` and `fairness` perspective. To quantify the performance of the algorithms, we compute the metrics on the Quantum of Improvement (`QoI`), defined in Equation 5.2. `P`, `G`, and `L` refer to the performances of the personalized algorithm, the global `FedAvg` model, and the local model, respectively for a user `i`. Subsequently, all metrics introduced in the following sub-sections are applied on the `QoI` to answer the questions raised previously.

$$\mathtt{QoI_i} = \mathtt{P_i} - \mathtt{max}(\mathtt{G_i}, \mathtt{L_i}) \tag{5.2}$$

Performance Metrics

The following metrics come under the purview of performance. These metrics help us answer which personalization algorithm is better from a performance perspective on model accuracy.

Percentage of User-models Improved (PUI). `PUI` is the percentage of all the users who have experienced an improvement compared to their local and global models. Ideally, a good personalization algorithm can improve the per-user accuracy of a maximal set of users.

$$\mathtt{PUI} = \frac{\mathtt{COUNT}(\mathtt{QoI_i} > 0)}{\mathtt{COUNT}(\mathtt{U})} \times 100, \mathtt{i} \in \mathtt{U}(\mathtt{U} : \mathtt{Users}) \tag{5.3}$$

Percentage of User-models Decreased (PUD). `PUD` is the percentage of all the users who have experienced a decrease compared to their local and global models. Ideally, a good personalization algorithm does not lead to any decrease in performance for any user. If this is not possible, it leads to a minimal set of users whose performance decreases.

$$\mathtt{PUD} = \frac{\mathtt{COUNT}(\mathtt{QoI_i} < 0)}{\mathtt{COUNT}(\mathtt{U})} \times 100, \mathtt{i} \in \mathtt{U}(\mathtt{U} : \mathtt{Users}) \tag{5.4}$$

Median Percentage of Improvement (MPI). `MPI` is computed as $\mathtt{Median}(\mathtt{U}^{+})$ where $\mathtt{Median}(.)$ is the function that returns the median of its input, and $\mathtt{U}^{+}$ is the `QoI` of the set of users who obtained an increase in their performance. A good personalization algorithm has a high median performance value among the users that obtain a performance improvement from a performance perspective.

Average Percentage of Improvement (API). `API` is the average percentage improvement among the users who obtained an increase in their performance ($\mathtt{U}^{+}$). In a normally distributed distribution, the mean is the best measure of central tendency. However, when this is not the case, the median might be a better measure of central tendency. Since we do not know how the QoI distributions will look like, we have both the measures.

$$\mathtt{API} = \frac{\sum_{\mathtt{i} \in \mathtt{U}^+} \mathtt{QoI}_\mathtt{i}}{\mathtt{len}(\mathtt{U}^+)} \tag{5.5}$$

Median Percentage of Decrease (MPD). MPD is computed as $\mathtt{Median}(\mathtt{U}^-)$ where $\mathtt{Median}(.)$ is the function that returns the median of its input, and $\mathtt{U}^-$ is the QoI of the set of users whose performance decreased with the application of the personalization algorithm(s).

Average Percentage of Decrease (APD). APD is the average percentage decrease among the users whose performance decreased ($\mathtt{U}^-$).

$$\mathtt{APD} = \frac{\sum_{\mathtt{i} \in \mathtt{U}^-} \mathtt{QoI}_\mathtt{i}}{\mathtt{len}(\mathtt{U}^-)} \tag{5.6}$$

Fairness metrics

In this section, we introduce metrics to evaluate personalization algorithms from a fairness perspective.

Average Variance (AV). For two personalization techniques $\mathtt{t}$ and $\mathtt{t}'$, the performance distribution among $\mathtt{K}$ users represented by $\{\mathtt{F}_1(\mathtt{t}), \ldots, \mathtt{F}_\mathtt{K}(\mathtt{t})\}$ is more fair (uniform) under technique $\mathtt{t}$ than $\mathtt{t}'$ if the following holds:

$$\mathtt{AV}(\mathtt{F}_1(\mathtt{t}), ...\mathtt{F}_\mathtt{K}(\mathtt{t})) < \mathtt{AV}(\mathtt{F}_1(\mathtt{t}'), ...\mathtt{F}_\mathtt{K}(\mathtt{t}')) \tag{5.7}$$

where, $\mathtt{AV}(\mathtt{F}_1(\mathtt{t}), ...\mathtt{F}_\mathtt{K}(\mathtt{t}))$ is computed as follows.

$$\mathtt{AV} = \frac{1}{\mathtt{K}} \sum_{\mathtt{i}=1}^{\mathtt{K}} (\mathtt{F}_\mathtt{i}(\mathtt{t}) - \bar{\mathtt{F}}(\mathtt{t}))^2 \tag{5.8}$$

$\bar{\mathtt{F}}(\mathtt{t})$ in Equation 5.8 refers to the average model performance across all the users and is computed as $\mathtt{F}\bar{(\mathtt{t})} = \frac{1}{\mathtt{K}} \sum_{\mathtt{i}=1}^{\mathtt{K}} (\mathtt{F}_\mathtt{i}(\mathtt{t}))$.

Cosine Similarity (CS). For two personalization techniques $\mathtt{t}$ and $\mathtt{t}'$, the performance distribution among $\mathtt{K}$ users represented by $\{\mathtt{F_1(t)}, \ldots, \mathtt{F_K(t)}\}$ is more fair (uniform) under technique $\mathtt{t}$ than $\mathtt{t}'$ if the following holds:

$$\mathtt{CS}[(\mathtt{F_1(t)}, ...\mathtt{F_K(t)}), \mathtt{1}] \geq \mathtt{CS}[(\mathtt{F_1(t')}, ...\mathtt{F_K(t')}), \mathtt{1}] \tag{5.9}$$

where, $\mathtt{CS}(\mathtt{F_1(t)}, ...\mathtt{F_K(t)})$ is computed as follows.

$$\mathtt{CS} = \frac{\frac{1}{\mathtt{K}} \sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i(t)}}{\sqrt{\frac{1}{\mathtt{K}} \sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i^2(t)}}} \tag{5.10}$$

Entropy. For two personalization techniques $\mathtt{t}$ and $\mathtt{t}'$, the performance distribution among $\mathtt{K}$ users represented by $\{\mathtt{F_1(t)}, \ldots, \mathtt{F_K(t)}\}$ is more fair (uniform) under technique $\mathtt{t}$ than $\mathtt{t}'$ if the following holds:

$$\mathtt{Entropy}(\mathtt{F_1(t)}, ...\mathtt{F_K(t)}) \geq \mathtt{Entropy}(\mathtt{F_1(t')}, ...\mathtt{F_K(t')}) \tag{5.11}$$

where $\mathtt{Entropy}$ is defined as follows.

$$\mathtt{Entropy} = -\sum_{\mathtt{i}=1}^{\mathtt{K}} \frac{\mathtt{F_i(t)}}{\sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i(t)}} \mathtt{log}\left(\frac{\mathtt{F_i(t)}}{\sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i(t)}}\right) \tag{5.12}$$

Jain's Index (JI). $\mathtt{JI}$ [44] is one of the earliest proposed and most widely studied fairness measures in the domain of computer networks and resource allocation. It is defined as follows:

$$\mathtt{JI} = \frac{[\sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i(t)}]^2}{\mathtt{K} \sum_{\mathtt{i}=1}^{\mathtt{K}} \mathtt{F_i(t)}^2} \tag{5.13}$$

The performance distribution among $\mathtt{K}$ users is represented by $\{\mathtt{F_1(t)}, \ldots, \mathtt{F_K(t)}\}$. Also, $\mathtt{JI}$ is bounded, i.e., $0 \leq \mathtt{JI} \leq 1$. A large value of $\mathtt{JI}$ represents a fairer distribution from the system's perspective.

It is important to note that the $\mathtt{QoI}$ can have negative values. This means that the personalization algorithm has led to a decrease in that particular user's performance rather than an expected increase. In such cases, the direct application of the fairness metrics will lead to an error. Instead split the `QoI` into two sets: a set of `QoI` values made up of users who have experienced an increase ($\mathtt{U}^{+}$), and a set of absolute `QoI` values made up of users who have experienced a decrease ($\mathtt{U}^{-}$). Then the fairness metrics can be applied to both these sets independently, and interpreted accordingly.

Generalization of Metrics to FL. The evaluation metrics introduced in this section are not restricted to personalized `FL`. They can also be applied in the `FL` setting. There have been diverse federated optimizers proposed in `FL` as an alternative to the base `FedAvg`, such as `FedBoost` [45], `STC` [46], `FedAT` [47], `Overlap-FedAvg` [48].

In these works, a new optimizer's performance is compared against `FedAvg` and other `FL` algorithms using only average accuracy. The evaluation metrics introduced for personalized `FL` approaches can be used to evaluate fairness across the users and also measure the performance of federated optimizers other than solely reporting the average accuracy.

5.2.4 Framework Implementation

The `Data Generator` module of our framework currently supports two datasets (CIFAR-10 and MNIST), and four data-splits (**DS-1**, **DS-2**, **DS-3**, and **DS-4**). In the `Evaluation Metrics` module, we implemented the metrics from a `performance` perspective and a `fairness` perspective. We integrated the following `FL` algorithms into our framework: PersFL, PersFL-GD, `FedPer`, `pFedMe`, and `Per-FedAvg`. We are limited by the non-availability of the code and the hyper-parameter details to replicate the other personalization algorithms' reported results. Thus, we have not included them in the current work. Our framework is implemented on Python `3.7`, and a PyTorch version of `v1.3.1` and runs on an NVIDIA Tesla T4 GPU with `16GB` memory with a CUDA version of 10.0 and a driver version of 410.104.

6. EXPERIMENTATION

We evaluate the performance of PersFL using two datasets, each with three data-splits, and compare its results with `FedAvg` and three recent personalization approaches, `FedPer`, `pFedMe`, and `Per-FedAvg`. Table 6.1 describes the compared approaches with PersFL, including the datasets used in their evaluation, data-splits (detailed below), and the algorithms that they compare with their techniques. In Section 6.1 the degree of personalization of each personalization algorithm as well as `FedAvg` is analyzed. In Section 6.2, we study the equitable notion of fairness among users across all personalization algorithms. The impact of optimal teachers on performance is studied in Section 6.3. The selection of optimal parameters for the distillation process is presented in Section 6.4. We study a variant of PersFL, namely PersFL-GD, and compare the difference in performances between the two formulations in Section 6.5. In Section 6.6, we study the number of global communication rounds required for all the personalized algorithms. In Section 6.7 we visualize the averaged values of the distillation parameters over the different experimental runs, across all users.

Subsequently, in Section 6.8 and Section 6.9 we first evaluate the personalized `FL` algorithms from a *performance* perspective and then from a *fairness* perspective, respectively. Using the per-user accuracies, we apply the metrics that we introduced in Section 5.2.3 to answer the following questions:

1. *Which algorithm performs the best in terms of accuracy across all the users?*
2. *Which algorithm is the fairest amongst all of them?*

We conduct all the experiments with 10 users. We make three assumptions in line with the assumptions made in compared approaches. First, we assume that all users are active during the entire training phase to speed up the model convergence. Second, the data of each user does not change between the global aggregations. Lastly, the hyper-parameters, batch-size (`B`), and local epochs (`E`), are invariant among the participant users. We conduct all experiments with a `60%-20%-20%` train-validation-test split on an NVIDIA Tesla T4 GPU with `16GB` memory.

Table 6.1. The details of the personalized federated learning methods compared with PersFL.

#	Method	Datasets	Comparison	Datasplit
1	**Per-FedAvg** [3]	MNIST, CIFAR10	`FedAvg`	Custom
2	**pFedMe** [4]	MNIST, Synthetic dataset	`FedAvg`, `Per-FedAvg`	DS-3
3	**FedPer** [5]	FLICKR-AES, CIFAR-10, CIFAR-100	`FedAvg`	DS-1

6.1 Degree of Personalization across Users

To study the degree (extent) of personalization of PersFL and to compare it with the other personalization approaches, we average experiments of CIFAR-10 and MNIST datasets over 5 experimental runs for each data-split. Table 6.2 and Table 6.3 show the per-user accuracy of `Fed-Avg`, PersFL, `FedPer`, `pFedMe`, and `Per-FedAvg` on different datasplits of CIFAR-10 and MNIST. Below, we present the performance of PersFL with the `FedAvg` model and the best performing algorithm on the average accuracy across users.

Our analysis of CIFAR-10 in Table 6.2 shows that PersFL performs better than other personalization techniques across all data-splits. In comparison to the `FedAvg` model, PersFL leads to a percentage increase of 82%, 22.3%, and 76.6% on **DS-1**, **DS-2**, and **DS-3**. For the compared approaches, for all data splits, PersFL improves the on average accuracy by 4.7%, 0.6% and 3.9% across users compared to `FedPer`, `pFedMe`, and `FedPer` respectively. The absolute improvement of PersFL over other techniques on **DS-1**, **DS-2** and **DS-3** is 3.7%, 0.4% and 3.1%, compared to `FedPer`, `pFedMe`, and `FedPer`.

The results of our experiments on MNIST in Table 6.3 show that, in comparison to the `FedAvg` model, PersFL improves the accuracy on average by 7.1%, 2.9%, and 3.1% on **DS-1**, **DS-2**, and **DS-3**. PersFL performs similar to the other approaches. In **DS-1**, the best-performing approach `Per-FedAvg` gives 98.9% accuracy, which performs slightly better than 98.6% accuracy of PersFL. In **DS-2**, PersFL gives a 0.4% increase in accuracy across users compared to `Per-FedAvg`. In **DS-3**, both PersFL and `FedPer` gives 99.4% accuracy.

Table 6.2. Accuracy per user for the different personalization algorithms on the different data-splits on the CIFAR-10 dataset.

	FedAvg			**PersFL**			**FedPer**			**pFedMe**			**Per-FedAvg**		
Users	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**
User 0	43.6	50.8	48.2	85.5	61.3	94.5	83.2	57.2	93.1	74.3	61.7	94.2	69.2	58.2	92.5
User 1	50.9	45.3	40.8	78.2	56.9	79.9	74.5	51.4	77	64.1	57.3	79.2	65	56.1	73.7
User 2	44.5	49.4	31.2	82.2	57.3	68.9	78.4	53.2	64.7	69.6	57.3	64.4	67.9	57.2	64.6
User 3	51.3	46.5	31.5	82.1	60.1	82.5	77.9	55.4	77.5	69.4	58.9	72.5	67.2	58.8	77
User 4	45.3	50.8	49.4	79.4	59.1	82.5	76.1	54.4	78.6	67.2	59.5	80	65.8	59.4	82.5
User 5	44.2	50.7	47.8	77.1	61.9	79.9	72.1	57.6	76.9	62.1	60.6	77.5	62.7	59.3	77.9
User 6	35.8	46.5	56.8	75.6	58.9	90.3	70.9	53.3	88.5	59.9	58.6	88.3	58.2	57.7	89.1
User 7	37.9	49.5	58.1	79.7	61.2	87.6	75.6	56.9	84.6	65.8	60.2	84	64.3	58	83.7
User 8	47.7	48.7	49	87.7	60	76.7	84.4	57.5	73.5	75.5	59.6	66.9	72.5	55.8	64.1
User 9	48.6	49.1	53.5	91	58.8	80.3	88.5	54.3	77.9	81.7	58.3	73.8	76.6	55.3	72.7
Avg Acc	**45**	**48.7**	**46.6**	**81.9**	**59.6**	**82.3**	**78.2**	**55.1**	**79.2**	**69**	**59.2**	**78.1**	**66.9**	**57.6**	**77.8**
Std Dev	**5.1**	**2**	**9.4**	**4.9**	**1.7**	**7.2**	**5.6**	**2.1**	**7.9**	**6.7**	**1.4**	**9.2**	**5.1**	**1.5**	**9.5**

Table 6.3. Accuracy per user for the different personalization algorithms on the different data-splits on the MNIST dataset.

	FedAvg			**PersFL**			**FedPer**			**pFedMe**			**Per-FedAvg**		
Users	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**
User 0	91.2	84.2	98.3	98.3	88.6	99.6	98.2	84.2	99.9	94.8	88.2	99	97.3	87.5	98.6
User 1	92.2	85.2	97.9	98.8	86.3	99.3	98.8	85.2	99.3	95.8	86.1	98.3	99	86.1	97.6
User 2	88	83.5	96.2	99.5	86	98.7	98.3	83.5	98.2	93.2	85.4	97.5	99.7	86.2	96.8
User 3	93.2	84.4	94.8	98.2	87.9	99.6	97.8	84.4	99.6	94.3	86.7	97.6	98.7	87.5	96
User 4	92.3	86.6	96.3	98.7	88.1	99.6	98.2	86.6	99.6	93.7	87.4	98.1	99	88	97.1
User 5	92.2	84.5	97.3	98.3	86.4	99.2	97.8	84.5	99.1	94.2	85.3	98.5	98.8	86.8	97.7
User 6	93.8	87	97.7	98.7	88.8	99.8	98.5	87	99.9	95	88	98.8	99.2	87.9	98.2
User 7	91.5	86.8	95.7	98.7	88.7	99.6	98.5	86.8	99.4	95.8	87.7	98	99.3	88.2	96.9
User 8	94.3	85.8	94.3	98.3	89.1	99.2	98.3	85.8	98.9	94.5	87.9	97.8	98.8	88.5	96.5
User 9	91.3	86.7	95.9	98.5	89.7	99.7	98.5	86.7	99.6	96.2	88.6	98.3	99.2	89.1	97.4
Avg Acc	**92**	**85.5**	**96.4**	**98.6**	**88**	**99.4**	**98.3**	**85.5**	**99.4**	**94.8**	**87.1**	**98.2**	**98.9**	**87.6**	**97.3**
Std Dev	**1.7**	**1.3**	**1.3**	**0.4**	**1.3**	**0.3**	**0.3**	**1.3**	**0.5**	**1**	**1.2**	**0.5**	**0.6**	**1**	**0.8**

Table 6.4. Average Accuracy across all the users for personalized models initialized with the weights of FedAvg model vs. Optimal Teacher model for each user on (a) CIFAR-10 and (b) MNIST datasets.

	DS-1		**DS-2**		**DS-3**	
Users	**FedAvg**	**Opt Teacher**	**FedAvg**	**Opt Teacher**	**FedAvg**	**Opt Teacher**
User 0	84.5	85.5	59.5	61.3	94.6	94.5
User 1	77.6	78.2	55.9	56.9	78.7	79.9
User 2	81.2	82.2	55.7	57.3	68.2	68.9
User 3	81.4	82.1	58.6	60.1	81	82.5
User 4	78.7	79.4	58.1	59.1	82	82.5
User 5	75.2	77.1	60.9	61.9	79.6	79.9
User 6	74.4	75.6	56.6	58.9	90.1	90.3
User 7	79.4	79.7	59.6	61.2	86.9	87.6
User 8	86.6	87.7	58.4	60	75.8	76.7
User 9	90.4	91	56.3	58.8	79.5	80.3
Avg Acc	**80.9**	**81.9**	**58**	**59.6**	**81.6**	**82.3**
Std Dev	**5**	**4.9**	**1.8**	**1.7**	**7.5**	**7.2**

(a) CIFAR-10

	DS-1		**DS-2**		**DS-3**	
Users	**FedAvg**	**Opt Teacher**	**FedAvg**	**Opt Teacher**	**FedAvg**	**Opt Teacher**
User 0	98.3	98.3	89	88.6	99.6	99.6
User 1	98.8	98.8	86.3	86.3	99.3	99.3
User 2	99.3	99.5	86.3	86	98.7	98.7
User 3	98.2	98.2	87.9	87.9	99.6	99.6
User 4	98.7	98.7	88.1	88.1	99.6	99.6
User 5	98.3	98.3	86.7	86.4	99.2	99.2
User 6	98.7	98.7	89	88.8	99.8	99.8
User 7	98.7	98.7	89	88.7	99.6	99.6
User 8	98.3	98.3	89	89.1	99.2	99.2
User 9	98.5	98.5	89.6	89.7	99.7	99.7
Avg Acc	**98.6**	**98.6**	**88.1**	**88**	**99.4**	**99.4**
Std Dev	**0.3**	**0.4**	**1.2**	**1.3**	**0.3**	**0.3**

(b) MNIST

6.2 Equitable Notion of Fairness among Users

To understand the distribution of the personalized models' performance across users, we compute the standard deviation (`SD`) of the per-user accuracy. From an equitable notion of fairness, this helps us understand how fair the personalization improvements are across users [25]. We use `AV` defined in Section 5.2.3 for this.

We compare PersFL with the best performing method in terms of average accuracy. In CIFAR-10 experiments (Table 6.2), PersFL yields a `SD` of `4.9` among the average accuracy of users on **DS-1**, which is the least deviation compared to the other approaches. This yields a reduction of `1.14x` in `SD` compared to `FedPer`. For **DS-2**, `pFedMe` yields an `SD` of `1.4`, slightly better than `1.7` `SD` of PersFL. For **DS-3**, PersFL achieves a reduction of `1.10x` in `SD` compared to `FedPer`.

For MNIST (Table 6.3), we observe similar results compared to the experiments on CIFAR-10. For **DS-1**, although `Per-FedAvg` with an average `98.9`% accuracy is better than PersFL's `98.6`% accuracy, PersFL leads to a `1.5x` reduction in `SD` compared to it. For **DS-2**, we find that the `SD` of `Per-FedAvg` at `1.0`, is lower than the `SD` of PersFL at `1.3`. On **DS-3**, the reduction factor for PersFL in `SD` stands at `1.67x` compared to `FedPer`. Overall, PersFL often leads to a reduction in the SD of the average accuracy across users, thus leading to a more uniform (fair) distribution than the compared methods.

6.3 Impact of Optimal Teachers on Accuracy

We claim that each user's optimal teacher model ($\mathtt{O_k}$) has maximal overlap with their local data distribution. To understand the impact of optimal teachers on the degree of personalization of the personalized models, we conduct the following experiment. We compare two different ways of choosing the teacher model, i.e., global `FedAvg` and optimal teacher models used in **Stage-**`2` of PersFL as the teacher for distillation. We observe that choosing the optimal teacher model as the teacher and performing distillation leads to more personalized solutions and a lower standard deviation in the per-user accuracy.

Table 6.4a and Table 6.4b show the results of using `FedAvg` and optimal teacher for CIFAR-10 and MNIST datasets. We make the following observations in terms of the av-

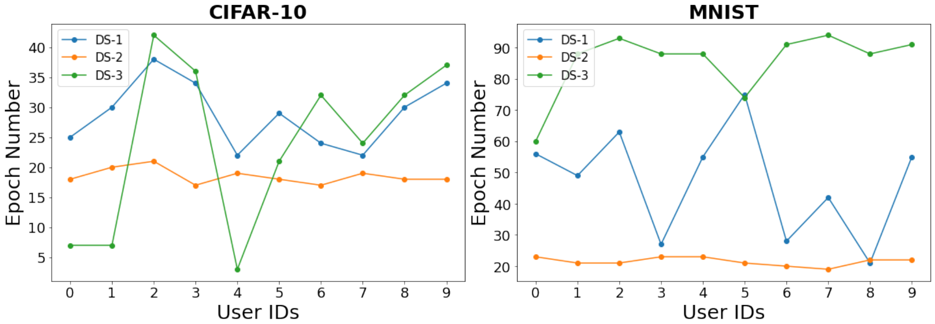


Figure 6.1. Average epoch numbers when users' optimal teacher models are selected across data-splits.

erage per-user accuracy. On **DS-1**, initializing with the optimal teacher model leads to a percentage increase of 1.2% and an absolute increase of 1%. On **DS-2**, the optimal teacher initialization leads to a percentage increase of 2.7% and an absolute increase of 1.6%. In the case of **DS-3**, there is a 0.8% percentage and 0.7% absolute increase attributed to initialization with optimal teachers. Additionally, we observe that PersFL yields slightly lower standard deviations of per-user accuracy. We conduct the same experiments on the MNIST dataset and present the results in Table 6.4b. The average accuracy across users and the standard deviations of the average accuracy are very similar across all the methods. We argue that this variation can be attributed to the lack of inter-class variations in the MNIST dataset.

Figure 6.1 shows the averaged epoch numbers across the different experimental runs for both CIFAR-10 and MNIST, across all data-splits. The users choose their optimal teacher model after participating in the global communication rounds. We observe that the averaged epoch numbers in which the optimal teacher models are chosen are predominantly different. This observation validates our hypothesis that each user has a unique optimal teacher model, and all users should not use the same teacher model. Turning to Figure 4.1, the dashed curve closest to the solid curve corresponds to the optimal teacher model for a particular user `k`. The averaged epoch number across all experimental runs at which this optimal teacher model is chosen is shown in Figure 6.1.

Table 6.5. Average optimal values of λ and $\mathtt{T}$ for each user across different experimental runs on CIFAR-10 and MNIST.

	CIFAR-10						**MNIST**					
	DS-1		**DS-2**		**DS-3**		**DS-1**		**DS-2**		**DS-3**	
Users	**T**	λ	**T**	λ	**T**	λ	**T**	λ	**T**	λ	**T**	λ
User 0	12.2	0.45	8.2	0.6	6.6	0.25	1	0	11.4	0.45	1	0
User 1	8.2	0.4	25	0.7	12.2	0.5	1	0	5.8	0.6	1	0
User 2	7.4	0.45	12.2	0.65	2.6	0.3	1.8	0.1	15.4	0.4	1	0
User 3	17	0.3	21	0.6	17	0.35	1	0	10.6	0.45	1	0
User 4	17	0.5	11.4	0.75	11.4	0.15	1	0.05	11.4	0.6	1	0
User 5	12.2	0.55	16.2	0.6	20.2	0.35	1	0.05	1.8	0.65	5.8	0.05
User 6	20.2	0.6	17	0.7	7.4	0.2	1	0.05	11.4	0.4	1	0
User 7	12.2	0.4	2.6	0.65	4.2	0.1	1	0	11.4	0.45	1	0
User 8	12.2	0.5	21	0.75	2.6	0.35	1	0.1	3.4	0.6	1	0
User 9	20.2	0.3	17	0.75	8.2	0.6	1	0	6.6	0.5	1	0

6.4 Optimal Parameter Selection

We conduct experiments to understand distillation's effectiveness by investigating the values of the optimal distillation parameters chosen by each user's student model. Table 6.5 shows the average values of distillation parameters, imitation (λ) and temperature ($\mathtt{T}$), averaged out over the five experimental runs on CIFAR-10 and MNIST datasets. For CIFAR-10, we observe that the λ parameter values are non-zero across users and data-splits. This observation means that distillation is effective, and the teacher model's knowledge is useful in learning the student model. In contrast, in the case of MNIST, we observe that λ parameter values are mostly very close to $\mathtt{0}$ in the case of **DS-1** and **DS-3**. We believe that this is because of the nature of the MNIST dataset i.e., due to the lack of inter-class variations. We show a detailed analysis of the variation of the classification accuracy on the test-set across users depending on the values of the distillation parameters λ and $\mathtt{T}$ in Section 6.7.

6.5 Variants of PersFL

The distillation objective of PersFL in Equation 4.2 can be implemented with different objective functions defining how to distill the information from the optimal teacher model into the user's local model. To demonstrate the generalization of PersFL to different distillation objectives, we perform the distillation with a different loss function and compare it with Equation 4.2.

We define the following objective function between the soft labels and the student's soft predictions:

$$\mathtt{A_k} = \underset{\substack{\lambda\in|\lambda|_{\mathtt{C}},\mathtt{T}\in|\mathtt{T}|_{\mathtt{C}}\\ \mathtt{A_k}\in|\mathtt{A_k}|_{\mathtt{C}}}}{\arg\min} \overbrace{(\mathtt{1}-\lambda)(\mathcal{L}_{\mathtt{cross}}(\sigma(\mathtt{A_k}(\mathtt{x_k^{train}})),\mathtt{y_k^{train}}))}^{\text{hard-loss}} \\ + \overbrace{(\lambda \mathtt{T}^2)\times\mathcal{L}_{\mathtt{cross}}(\sigma(\frac{\mathtt{A_k}(\mathtt{x_k^{train}})}{\mathtt{T}}),\sigma(\frac{\mathtt{O_k}(\mathtt{x_k^{train}})}{\mathtt{T}}))}^{\text{soft-loss}} \quad (6.1)$$

The soft predictions of a student are defined as the predictions of the student model, which are scaled by the temperature parameter, $\sigma(\mathtt{A_k}(\mathtt{x_k^{train}})/\mathtt{T})$. Equation 6.1 is different from the original formulation of PersFL in Equation 4.2. Here we do not use KL-divergence to enforce the similarity of logits between the teacher and student model. In contrast, we compute the cross-entropy between the models. We refer to this variant of PersFL to `PersFL-GD`. We note that other distillation methods can be easily integrated into PersFL.

Table 6.6 shows the results of both variants of PersFL across all data-splits on both CIFAR-10 and MNIST datasets. We make the following three observations. First, in MNIST, both variants of PersFL on **DS-1** and **DS-3** have a very similar performance in terms of average accuracy across all users. In **DS-2** PersFL has an absolute performance improvement of $\mathtt{0.7}\%$ compared to PersFL-GD. Second, the performance difference is relatively more in CIFAR-10 experiments. PersFL has an absolute improvement of $\mathtt{1.4}\%$ on **DS-2** of CIFAR-10 and an improvement of $\mathtt{0.6}\%$ on **DS-3**. In all cases, PersFL performs better than or at least equal to PersFL-GD. Lastly, in terms of the standard deviations between the per-user accuracy, both PersFL and PersFL-GD yield very similar performance.

Table 6.6. Performance comparison of variants of PersFL. ❶ is for PersFL, and ❷ is for PersFL-GD objective functions.

	CIFAR-10						**MNIST**					
	DS-1		**DS-2**		**DS-3**		**DS-1**		**DS-2**		**DS-3**	
Users	❶	❷	❶	❷	❶	❷	❶	❷	❶	❷	❶	❷
User 0	85.5	85.5	61.3	60	94.5	94.1	98.3	98.3	88.6	88.1	99.6	99.8
User 1	78.2	77.8	56.9	56.4	79.9	78.8	98.8	99	86.3	85.3	99.3	99.4
User 2	82.2	81.6	57.3	56.2	68.9	68.2	99.5	99.2	86	85.5	98.7	98.7
User 3	82.1	81.5	60.1	58.4	82.5	82	98.2	98.2	87.9	87.2	99.6	99.8
User 4	79.4	79.2	59.1	58.2	82.5	82.4	98.7	98.5	88.1	87.5	99.6	99.7
User 5	77.1	76.3	61.9	61	79.9	78.8	98.3	98.7	86.4	85.6	99.2	99.3
User 6	75.6	75.5	58.9	57.9	90.3	90.1	98.7	98.5	88.8	88.3	99.8	99.9
User 7	79.7	79.5	61.2	58.8	87.6	87	98.7	99	88.7	88.2	99.6	99.7
User 8	87.7	87.1	60	58.4	76.7	75.6	98.3	98.8	89.1	88.4	99.2	99.3
User 9	91	90.5	58.8	56.3	80.3	79.5	98.5	98.8	89.7	88.7	99.7	99.8
Avg Acc	**81.9**	**81.5**	**59.6**	**58.2**	**82.3**	**81.7**	**98.6**	**98.7**	**88**	**87.3**	**99.4**	**99.5**
Std Dev	**4.9**	**4.9**	**1.7**	**1.6**	**7.2**	**7.4**	**0.4**	**0.3**	**1.3**	**1.3**	**0.3**	**0.4**

Table 6.7. Comparison of # of global communication rounds

	CIFAR-10			**MNIST**		
Method	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**
FedAvg	50	25	100	100	50	100
PersFL	50	25	50	100	25	100
FedPer	50	25	100	100	50	100
pFedMe	800	1000	800	800	800	800
Per-FedAvg	800	800	1000	800	800	800

6.6 Global Communication Rounds

In this set of experiments, we compare the number of global communication rounds taken by each personalization algorithm. Table 6.7 shows the number of global communication rounds for each algorithm. Below, for each dataset's data-splits, we compare the number of global communication rounds required by PersFL with the best performing method. We observe that, in CIFAR-10 for **DS-1**, PersFL takes 50 communication rounds similar to `FedPer`. For **DS-2**, PersFL takes `0.03x` less communication rounds than `pFedMe`, and for **DS-3**, PersFL takes `0.5x` less communication rounds than `FedPer`. In MNIST experiments, we observe that for **DS-1** PersFL needs `0.13x` less communication rounds required by `Per-`

`FedAvg`. In **DS-2**, PersFL requires `0.03x` less communication rounds compared to `Per-FedAvg`, and PersFL needs the same number of communication rounds as `FedPer` in **DS-3**.

6.7 Optimal Distillation Parameters

We present a detailed analysis of the selection of the optimal parameters introduced in 6.4. Figures 6.2, 6.3, and 6.4 show the classification accuracy on the test set of each user for a combination of values of λ and `T` averaged out over the five experimental runs on CIFAR-10. Figures 6.5, 6.6, and 6.7 show the classification accuracy in the case of MNIST. The x-axes in these plots represent different values of the imitation parameters, and the y-axes represent the classification accuracy. We do not include the imitation parameter of `1` in these figures because it is never the case in our experiments that the optimal value of λ turns out to be `1`.

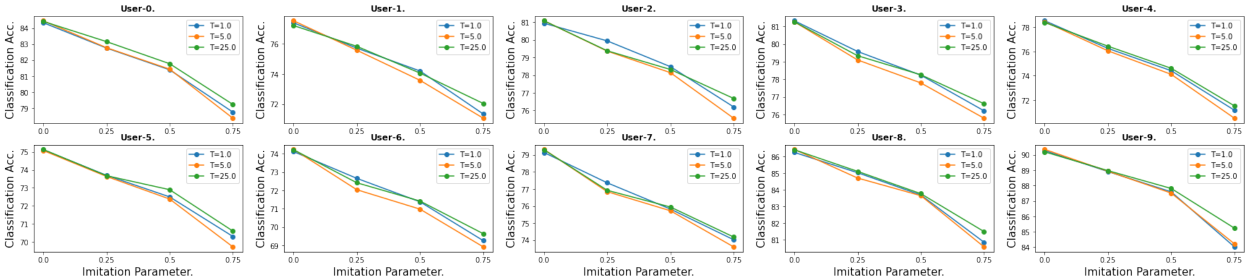


Figure 6.2. Per-user interaction plots between λ and `T` on **DS-1** of CIFAR-10 averaged over the experimental runs.

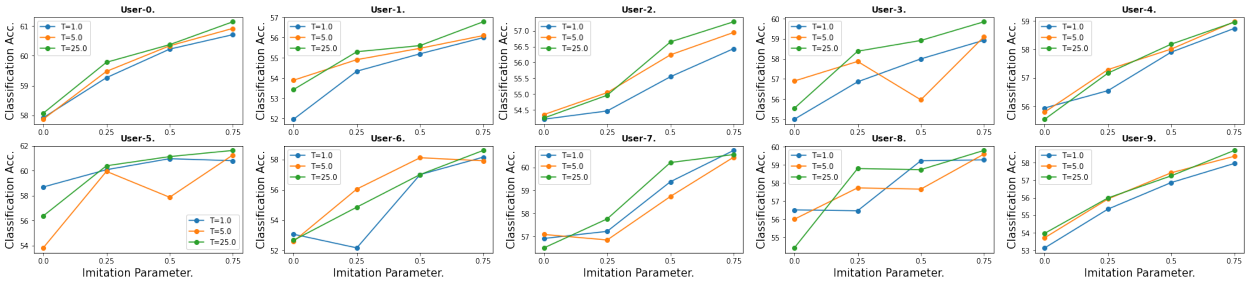


Figure 6.3. Per-user interaction plots between λ and `T` on **DS-2** of CIFAR-10 averaged over the experimental runs.

6.8 Performance Perspective Analysis

Instead of solely using average accuracy, we use average accuracy in conjunction with other metrics to make a more informed decision on evaluating personalized approaches. Table 6.8 shows the performance metrics applied to the `QoI` of the different personalization

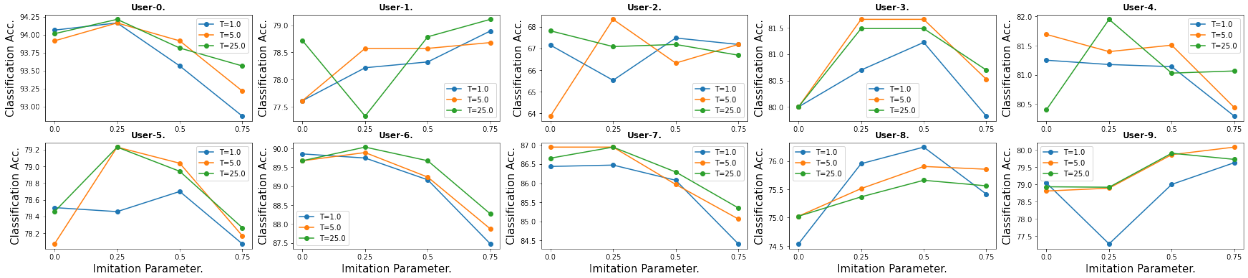

User-0.
User-1.
User-2.
User-3.
User-4.
User-5.
User-6.
User-7.
User-8.
User-9.

Figure 6.4. Per-user interaction plots between λ and `T` on **DS-3** of CIFAR-10 averaged over the experimental runs.

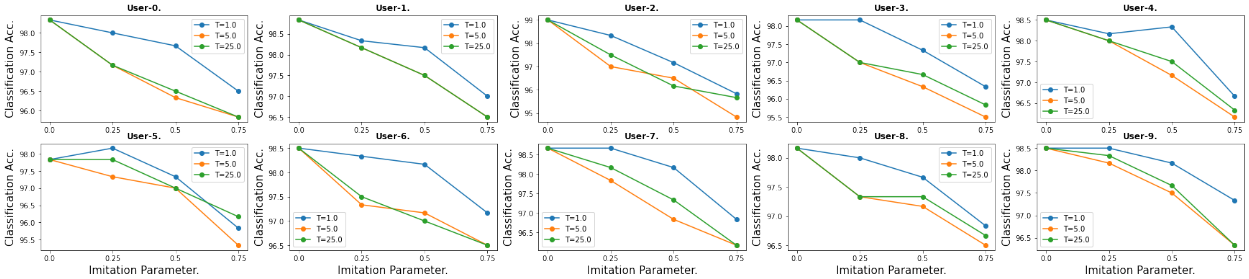

User-0.
User-1.
User-2.
User-3.
User-4.
User-5.
User-6.
User-7.
User-8.
User-9.

Figure 6.5. Per-user interaction plots between λ and `T` on **DS-1** of MNIST averaged over the experimental runs.

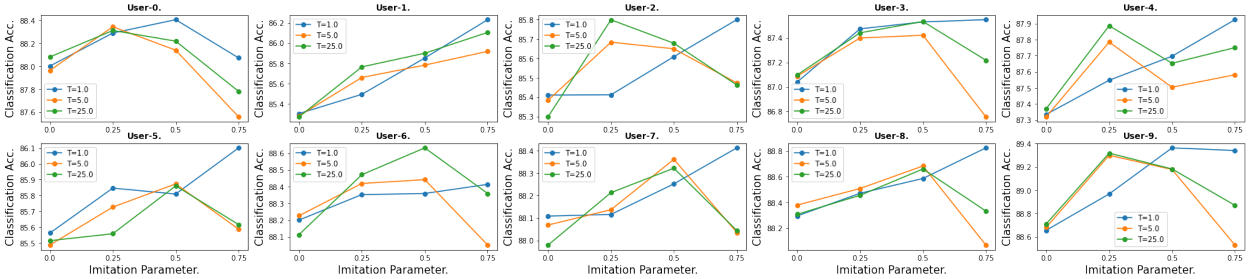

User-0.
User-1.
User-2.
User-3.
User-4.
User-5.
User-6.
User-7.
User-8.
User-9.

Figure 6.6. Per-user interaction plots between λ and `T` on **DS-2** of MNIST averaged over the experimental runs.

Table 6.8. A subset of the applicable performance metrics applied to the QoI of different personalization algorithms on CIFAR-10.

	PersFL			FedPer			pFedMe			PerFed		
Metrics	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**	**DS-1**	**DS-2**	**DS-3**
`PUI`	100	100	100	100	100	100	100	100	100	100	100	100
`MPI`	38.74	11.23	33.29	34.53	6.59	30.43	24.58	10.81	31.06	22.9	8.55	32.65
`API`	36.87	10.83	35.67	33.18	6.41	32.59	23.98	10.47	31.44	21.95	8.85	31.17
`AA`[†]	81.85	59.56	82.29	78.16	55.13	79.21	68.95	59.19	78.07	66.93	57.57	77.79

`AA`[†] is the average accuracy across all users.

algorithms across different data-splits of CIFAR-10. Not all the metrics are applicable in this case, and hence we only apply a subset of the metrics for performance. Metrics such as `PUD`, `MPD`, and `APD` are not applicable since all the personalization algorithms lead to an increase in the per-user performance. Therefore, the `PUI` for each personalization algorithm is `100`%.

In terms of `MPI` and `API`, on **DS-1**, PersFL and PersFL-GD perform the best at `38.74`% and `36.87`% for PersFL, and `38.23`% and `36.47`% for PersFL-GD, respectively. On **DS-2**, PersFL and `pFedMe` yield the highest `MPI` at `11.23`% and `10.81`%, respectively. Similarly these algorithms lead to an `API` of `10.83`% and `10.47`%, respectively. We observe that on **DS-3**, PersFL has the highest `MPI` and `API`, at `33.29`% and `35.67`%, respectively.

In terms of `average accuracy`, we observe that, PersFL and PersFL-GD perform the best at `81.85`%, and `81.45`% on **DS-1**. On **DS-2**, PersFL and `pFedMe` are the best perform-

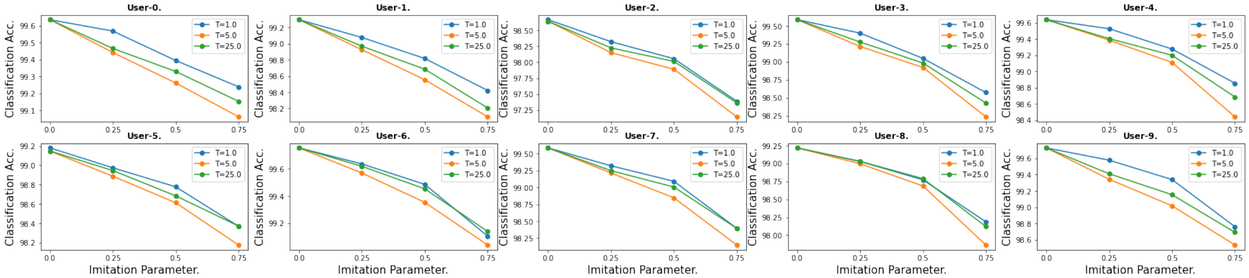
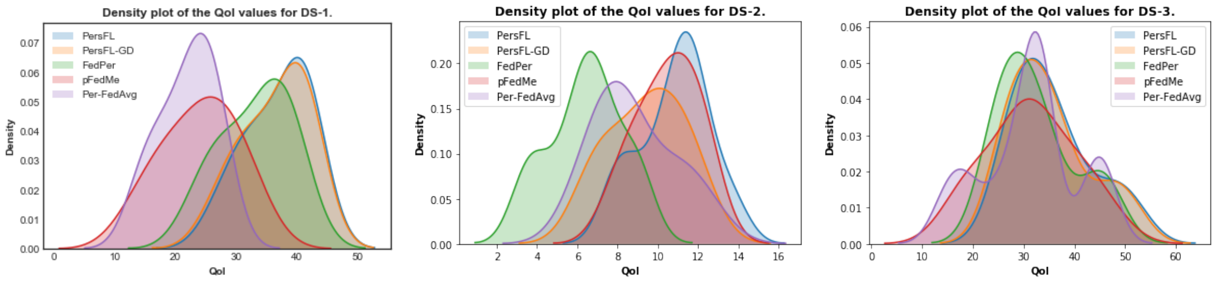


Figure 6.7. Per-user interaction plots between λ and `T` on **DS-3** of MNIST averaged over the experimental runs.



(a) **DS-1** (b) **DS-2** (c) **DS-3**

Figure 6.8. Density plot of `QoI` values for CIFAR-10.

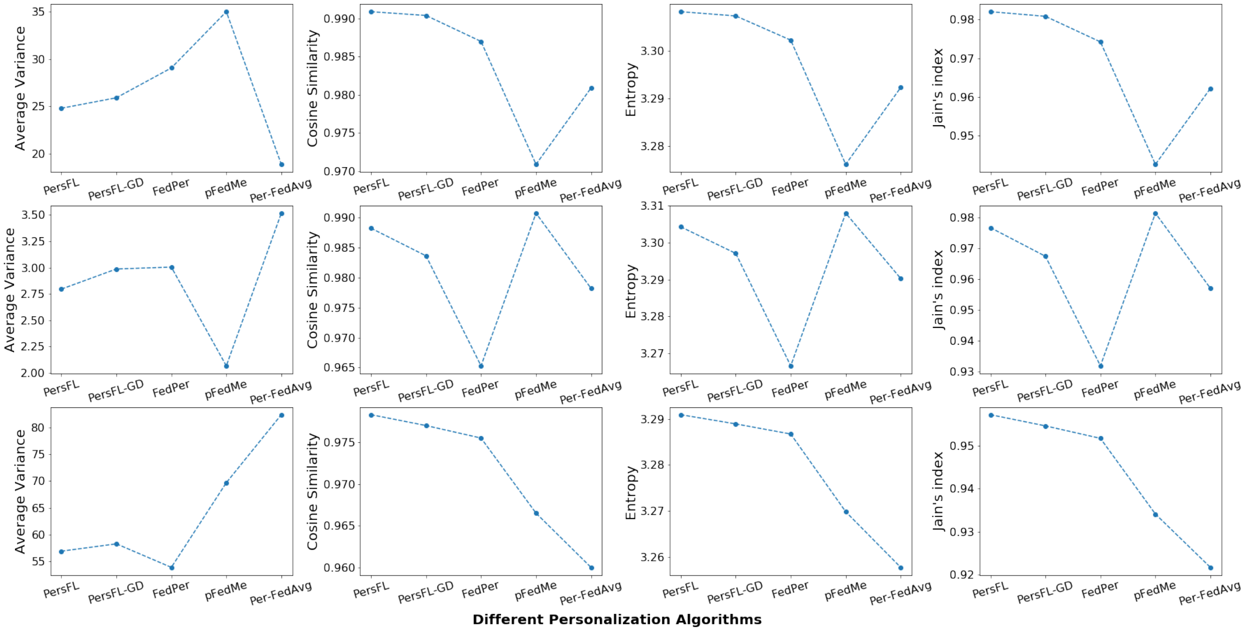


Figure 6.9. Fairness metrics applied to the personalized algorithms on different data-splits of CIFAR-10. The first row is for **DS-1**, second row for **DS-2**, and the last row for **DS-3**.

ing at `59.56`%, and `59.19`%, respectively, and on **DS-3**, PersFL and PersFL-GD are the top-performing techniques at `82.29`% and `81.64`%.

From `PUI`, `MPI`, `API`, and the `average accuracy`, we conclude that PersFL is indeed the best performing algorithm on **DS-1**. Similarly, from these metrics, we observe that `pFedMe` and PersFL are the best performing algorithms on **DS-2** and **DS-3**, respectively.

Remark 1. *We observe that the algorithm that has the best performance need not necessarily be the fairest algorithm for a particular data-split. This is indeed the case on **DS-2**, where* PersFL *is the better performing algorithm, whereas* `pFedMe` *is the fairer solution.*

Figure 6.8 shows the density plots of the `QoI` for all clients across three data-splits of the CIFAR-10 dataset. The peaks in a density plot help display where the values are concentrated over an interval. The x-axes of the plots denote the `QoI` intervals, whereas the y-axes denote the density. We observe that in the case of **DS-1**, the peaks as well as the distribution of both PersFL and PersFL-GD are very similar, concentrated around a `QoI` interval of `40`. In the case of **DS-2**, we observe that the peak of PersFL is associated with a higher `QoI` interval compared to `pFedMe`, but the distribution of `pFedMe` is more normal compared to PersFL as PersFL has an additional peak corresponding to the `QoI` interval of `8`. On **DS-3**, we observe that the `QoI` interval corresponding to the peaks of PersFL, PersFL-GD, `pFedMe`, and `Per-FedAvg`, are all almost the same value indicating that the `QoI` of these algorithms is concentrated around that value across all users.

6.9 Fairness Perspective Analysis

Figure 6.9 shows the different fairness metrics applied to **DS-1**, **DS-2**, and **DS-3** of CIFAR-10, respectively. In these figures, the x-axis represents the different personalization algorithms, and the y-axis refers to the different fairness metrics applied to these algorithms. We note that for an algorithm to be fair, in addition to being fair, it also needs to have good performance; just being fair without a good performance is not fair!

From Figure 6.9 on **DS-1**, we observe that for `AV`, `Per-FedAvg` has the lowest `AV` amongst all the personalization algorithms. A lower variance means that the performance distribution because of that algorithm's application is more uniform than the other algorithms. However, we observe that `Per-FedAvg` has a low value across the other fairness metrics. In the case of the other three metrics `CS`, `Entropy`, and `JI`, we find that PersFL and PersFL-GD are the best performing. For these metrics, the higher the value of the metric, the fairer the distribution. Between PersFL and PersFL-GD, PersFL has a slightly higher value than PersFL-GD across the other fairness metrics. Hence, for **DS-1**, we conclude that PersFL is the fairest amongst all the personalization algorithms as a majority of the fairness metrics show.

Remark 2. *It is not always the case that a personalized* `FL` *algorithm is the best across all the fairness metrics unanimously. An example of this observation is on* ***DS-1****, where* `Per-FedAvg` *has the lowest* `AV`*, but it is not the best across the other fairness metrics.*

In the case of **DS-2**, from Figure 6.9, we observe that `pFedMe` has the least `AV`. Clearly, it also has the highest value for the other three metrics `CS`, `Entropy`, and `JI`. We confirm that it has a good performance across all the users. On **DS-2** in Table 6.8, we see that `pFedMe` is very close to PersFL in terms of the performance metrics, and hence it has a good performance while at the same time being relatively fairer than PersFL. Therefore, `pFedMe` is the fairest algorithm on **DS-2**.

On **DS-3**, from Figure 6.9, we find that `FedPer` has the lowest `AV`. However, it is not the fairest algorithm according to the other fairness metrics. In fact, among the other three metrics, PersFL has the highest value. Hence, it is the fairest solution amongst all personalization algorithms, as shown by most fairness metrics. We also observe that in terms of performance, PersFL is the best performing algorithm on **DS-3**. From these observations, we conclude that PersFL is the fairest algorithm on **DS-3**.

Remark 3. *We also find that* PersFL *is both the best performing as well as the fairest solution among all the algorithms in two cases, i.e.,* ***DS-1*** *and* ***DS-3****.*

7. DISCUSSION AND LIMITATIONS

7.1 Personalization for New Participants

PersFL can easily adapt new users participating in the FL framework to learn personalized models. To detail, consider that a new user `k` joins the framework after the training phase of PersFL in search of a personalized solution. Since each user chooses their optimal teacher model at the end of the PersFL training phase, the `FedAvg` model for user `k` may serve as the teacher model i.e., $\mathtt{O_k}$ (assuming that only the `FedAvg` model in the final global aggregation round is stored). If the central aggregator stores the `FedAvg` model across the global aggregation rounds, user `k` can then choose $\mathtt{O_k}$ to be the most optimal `FedAvg` model across all the global aggregation rounds that has the lowest error on the validation data of user `k`. Following this, user `k` can then easily perform a search over λ and `T` to find the optimal values within the search space according to Equation 4.2. Subsequently, the user can perform distillation with $\mathtt{O_k}$ to learn a personalized model $\mathtt{A_k}$. However, the global `FedAvg` model learned over the aggregation rounds may not be the most optimal teacher model for the new participants when the number of new participants joining the framework increases. In future work, we will study the trade-off between the number of new participants and the accuracy of personalized models with respect to the shift in their data distribution.

7.2 Dual Optimization of PersFL

Our work raises some new important questions, such as how to unify distillation with personalization in FL such that personalized models are jointly learned for all users and how to incorporate feedback from the student models to learn more optimal teacher models? Future work will explore the joint learning of the global model and optimal distillation parameters for each user, i.e., joint optimization rather than the discrete formulation of PersFL. In this way, we plan to incorporate feedback from the student model after distillation to improve each user's optimal teacher model in the subsequent distillation steps.

7.3 Extension to `FL`

Apart from the currently used metrics in `FL` such as *number of communication rounds*, *number of data nodes* etc., we can additionally apply the metrics that we introduce to incorporate the notion of *per-user performance* and also *fairness*, which the metrics currently used in `FL` do not capture. This would further help in the evaluation of the `FL` algorithms.

7.4 Support for Personalization Algorithms

In the future, we will introduce alternative metrics to accuracy such as F1-score, precision, and recall which is currently not the case. We will also extend the framework to support more personalization algorithms and datasets and other types of data-split strategies.

8. CONCLUSIONS

We present PersFL[1], a personalized FL algorithm , which addresses the statistical heterogeneity issue between different clients' data to improve the FL performance. PersFL finds the optimal teacher model of each client during the FL training phase and distills the useful knowledge from optimal teachers into each user's local data after the training phase. We evaluate the effectiveness of PersFL on CIFAR-10 and MNIST datasets using three different data-splitting strategies. Experimentally, we show that PersFL outperforms the `FedAvg` and three state-of-the-art personalized FL methods, `pFedMe`, `Per-FedAvg` and `FedPer` on the majority of data-splits with minimal communication cost. We additionally provide a set of numerical experiments to demonstrate the performance of PersFL on different distillation objectives, how this performance is affected by the equitable notion of fairness among clients, and the number of communication rounds between clients and server.

We also introduce a framework for the domain of personalized `FL`. We have a `data-generator`, `federated model generation`, as well as `evaluation metrics` module in this framework. Each module has its specific utility in the framework, ranging from selecting the dataset and splitting it according to a data-split strategy to applying the different metrics introduced in this work. We introduce additional metrics alongside the `average accuracy` used to evaluate personalized `FL` models. We split these metrics into two *perspectives* based on their utility: `Performance`, and `Fairness`. We employ these metrics to find the best performing algorithm and the fairest solutions across different data-splits of the CIFAR-10 dataset.

---

[1] ↑PersFL code is available at https://tinyurl.com/1hp9ywfa.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, “Federated learning of deep networks using model averaging,” *CoRR*, vol. abs/1602.05629, 2016. arXiv: 1602.05629. [Online]. Available: http://arxiv.org/abs/1602.05629.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020, ISSN: 1558-0792. DOI: 10.1109/msp.2020.2975749. [Online]. Available: http://dx.doi.org/10.1109/MSP.2020.2975749.

[3] A. Fallah, A. Mokhtari, and A. Ozdaglar, “Personalized Federated Learning: A Meta-Learning Approach,” *arXiv e-prints*, arXiv:2002.07948, arXiv:2002.07948, Feb. 2020. arXiv: 2002.07948 `[cs.LG]`.

[4] C. T. Dinh, N. H. Tran, and T. Dung Nguyen, “Personalized Federated Learning with Moreau Envelopes,” *arXiv e-prints*, arXiv:2006.08848, arXiv:2006.08848, Jun. 2020. arXiv: 2006.08848 `[cs.LG]`.

[5] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, “Federated learning with personalization layers,” *CoRR*, vol. abs/1912.00818, 2019. arXiv: 1912.00818. [Online]. Available: http://arxiv.org/abs/1912.00818.

[6] C. Finn, P. Abbeel, and S. Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks,” *arXiv e-prints*, arXiv:1703.03400, arXiv:1703.03400, Mar. 2017. arXiv: 1703.03400 `[cs.LG]`.

[7] L. Liu, F. Zhang, J. Xiao, and C. Wu, *Evaluation framework for large-scale federated learning*, 2020. arXiv: 2003.01575 `[cs.LG]`.

[8] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. [Online]. Available: https://doi.org/10.1109/TKDE.2009.191.

[9] Y. Mansour, M. Mohri, and A. Rostamizadeh, “Domain adaptation: Learning bounds and algorithms,” 2009. arXiv: 0902.3430. [Online]. Available: http://arxiv.org/abs/0902.3430.

[10] Y. Jiang, J. Konecný, K. Rush, and S. Kannan, “Improving federated learning personalization via model agnostic meta learning,” *CoRR*, vol. abs/1909.12488, 2019. arXiv: 1909.12488. [Online]. Available: http://arxiv.org/abs/1909.12488.

[11] A. Nichol, J. Achiam, and J. Schulman, "On First-Order Meta-Learning Algorithms," *arXiv e-prints*, arXiv:1803.02999, arXiv:1803.02999, Mar. 2018. arXiv: 1803.02999 `[cs.LG]`.

[12] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, vol. abs/1802.07876, 2018. arXiv: 1802.07876. [Online]. Available: http://arxiv.org/abs/1802.07876.

[13] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few shot learning," *CoRR*, vol. abs/1707.09835, 2017. arXiv: 1707.09835. [Online]. Available: http://arxiv.org/abs/1707.09835.

[14] S. Lin, G. Yang, and J. Zhang, *Real-time edge intelligence in the making: A collaborative learning framework via federated meta-learning*, 2020. arXiv: 2001.03229 `[cs.LG]`.

[15] Y. Deng, M. Mahdi Kamani, and M. Mahdavi, "Adaptive Personalized Federated Learning," *arXiv e-prints*, arXiv:2003.13461, arXiv:2003.13461, Mar. 2020. arXiv: 2003.13461 `[cs.LG]`.

[16] Y. Mansour, M. Mohri, J. Ro, and A. Theertha Suresh, "Three Approaches for Personalization with Applications to Federated Learning," *arXiv e-prints*, arXiv:2002.10619, arXiv:2002.10619, Feb. 2020. arXiv: 2002.10619 `[cs.LG]`.

[17] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017, pp. 4424–4434. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf.

[18] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. Morency, "Think locally, act globally: Federated learning with local and global representations," 2020. arXiv: 2001.01523. [Online]. Available: http://arxiv.org/abs/2001.01523.

[19] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, *Personalized cross-silo federated learning on non-iid data*, 2021. arXiv: 2007.03797 `[cs.LG]`.

[20] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, K. Kuang, F. Wu, and C. Wu, *Federated mutual learning*, 2020. arXiv: 2006.16765 `[cs.LG]`.

[21] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets," *arXiv e-prints*, arXiv:2008.03371, arXiv:2008.03371, Aug. 2020. arXiv: 2008.03371 `[cs.LG]`.

[22] M. Khodak, M. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," *CoRR*, vol. abs/1906.02717, 2019. arXiv: 1906.02717. [Online]. Available: http://arxiv.org/abs/1906.02717.

[23] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," *CoRR*, vol. abs/1909.05830, 2019. arXiv: 1909.05830. [Online]. Available: http://arxiv.org/abs/1909.05830.

[24] C. Dwork, "Differential privacy," in *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ser. ICALP'06, Venice, Italy: Springer-Verlag, 2006, pp. 1–12, ISBN: 3540359079. DOI: 10.1007/11787006_1. [Online]. Available: https://doi.org/10.1007/11787006_1.

[25] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," *CoRR*, vol. abs/1905.10497, 2019. arXiv: 1905.10497. [Online]. Available: http://arxiv.org/abs/1905.10497.

[26] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[27] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[28] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018. arXiv: 1811.03604. [Online]. Available: http://arxiv.org/abs/1811.03604.

[29] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," *CoRR*, vol. abs/1910.10252, 2019. arXiv: 1910.10252. [Online]. Available: http://arxiv.org/abs/1910.10252.

[30] J.-J. Moreau, "Propriétés des applications 'prox'," French, *Compte Rendus Acad. Sci.*, no. 256, pp. 1069–1071, 1963.

[31] F. Hanzely and P. Richtárik, *Federated learning of a mixture of global and local models*, 2021. arXiv: 2002.05516 `[cs.LG]`.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531.

[33] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” *CoRR*, vol. abs/1706.00384, 2017. arXiv: 1706.00384. [Online]. Available: http://arxiv.org/abs/1706.00384.

[34] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7.

[35] C. Bucila, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’06, Philadelphia, PA, USA: Association for Computing Machinery, 2006, pp. 535–541, ISBN: 1595933395. DOI: 10.1145/1150402.1150464. [Online]. Available: https://doi.org/10.1145/1150402.1150464.

[36] V. Vapnik and A. Vashist, “A new learning paradigm: Learning using privileged information.,” *Neural Networks*, vol. 22, no. 5-6, pp. 544–557, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/nn/nn22.html#VapnikV09.

[37] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, “Unifying distillation and privileged information,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.03643.

[38] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999, ISSN: 1364-6613. DOI: https://doi.org/10.1016/S1364-6613(99)01294-2. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364661399012942.

[39] X. Li, K.-x. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” *ArXiv*, vol. abs/1907.02189, 2020.

[40] M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” 2019. arXiv: 1902.00146. [Online]. Available: http://arxiv.org/abs/1902.00146.

[41] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, “SCAFFOLD: stochastic controlled averaging for on-device federated learning,” *CoRR*, vol. abs/1910.06378, 2019. arXiv: 1910.06378. [Online]. Available: http://arxiv.org/abs/1910.06378.

[42] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, “On the convergence of federated optimization in heterogeneous networks,” *CoRR*, vol. abs/1812.06127, 2018. arXiv: 1812.06127. [Online]. Available: http://arxiv.org/abs/1812.06127.

[43] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification," *arXiv e-prints*, arXiv:1909.06335, arXiv:1909.06335, Sep. 2019. arXiv: 1909.06335 `[cs.LG]`.

[44] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *CoRR*, vol. cs.NI/9809099, 1998. [Online]. Available: https://arxiv.org/abs/cs/9809099.

[45] J. Hamer, M. Mohri, and A. T. Suresh, "FedBoost: A communication-efficient algorithm for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 3973–3983. [Online]. Available: http://proceedings.mlr.press/v119/hamer20a.html.

[46] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–14, Nov. 2019. DOI: 10.1109/TNNLS.2019.2944481.

[47] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, *Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data*, 2020. arXiv: 2010.05958 `[cs.DC]`.

[48] Y. Zhou, Y. Qing, and J. Lv, *Communication-efficient federated learning with compensated overlap-fedavg*, 2020. arXiv: 2012.06706 `[cs.LG]`.