Technometrics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/utch20

Personalized Federated Learning via Domain Adaptation with an Application to Distributed 3D Printing

Naichen Shi & Raed Al Kontar

To cite this article: Naichen Shi & Raed Al Kontar (2023): Personalized Federated Learning via Domain Adaptation with an Application to Distributed 3D Printing, Technometrics, DOI: 10.1080/00401706.2022.2157882

To link to this article: https://doi.org/10.1080/00401706.2022.2157882

TECHNOMETRICS
2023, VOL. 00, NO. 0, 1–12
https://doi.org/10.1080/00401706.2022.2157882

Taylor & Francis
Taylor & Francis Group

Personalized Federated Learning via Domain Adaptation with an Application to Distributed 3D Printing

Naichen Shi and Raed Al Kontar

Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI

ABSTRACT

Over the years, Internet of Things (IoT) devices have become more powerful. This sets forth a unique opportunity to exploit local computing resources to distribute model learning and circumvent the need to share raw data. The underlying distributed and privacy-preserving data analytics approach is often termed federated learning (FL). A key challenge in FL is the heterogeneity across local datasets. In this article, we propose a new personalized FL model, `PFL-DA`, by adopting the philosophy of domain adaptation. `PFL-DA` tackles two sources of data heterogeneity at the same time: a covariate and concept shift across local devices. We show, both theoretically and empirically, that `PFL-DA` overcomes intrinsic shortcomings in state of the art FL approaches and is able to borrow strength across devices while allowing them to retain their own personalized model. As a case study, we apply `PFL-DA` to distributed desktop 3D printing where we obtain more accurate predictions of printing speed, which can help improve the efficiency of the printers.

ARTICLE HISTORY
Received December 2021
Accepted November 2022

KEYWORDS
Concept shift; Covariate shift; Federated inference; Internet of Things; 3D Printing

1. Introduction

Traditional Internet of Things (IoT) enabled systems mainly follow a centralized approach for data analytics (Atzori, Iera, and Morabito 2010). In IoT, data from connected devices is regularly uploaded to a cloud/central server, where models are learned and then deployed back to the edge devices. However, the vast amount of data nowadays necessitates alternative data analytics paradigms. For instance, the need to upload large amounts of data to a central server incurs high communication and storage costs, demands large internet bandwidth, and leads to latency in deployment as models need to be transferred back to in-service connected devices. In addition, the centralized paradigm of IoT where data is agglomerated in a central server breeds concerns about privacy and security.

However, connected devices have become much more powerful in their computational capabilities over the years. AI chips are rapidly infiltrating the market. The processing power of mobile phones and wearable devices has seen significant advances. Small local computers such as Raspberry Pis have become commonplace in many industries such as manufacturing (Richardson and Wallace 2012). *This poses a significant opportunity to rethink traditional IoT so that most of the data processing happens where it is created, instead of uploading it to the cloud*.

In light of these challenges, federated learning (FL) has caught a lot of attention in recent years (McMahan et al. 2017; Li et al. 2020b). In FL, connected devices perform data analytics at the edge and transmit only focused updates needed for modeling learning to the cloud. As such, the cloud becomes an orchestrator of the training process that integrates updates from the local devices. With the orchestration of a central server, devices exploit their compute resources, borrow strength from each and retain local data privacy. Adding to that, within IoT, FL can reduce storage and communication costs as only minimum information needed to learn a model is shared.

Though the idea of FL dates back decades ago, to the early work of Mangasarian and Solodov (1994), it was only brought to the forefront of data analytics after the paper by McMahan et al. (2017) which proposed federated averaging (FedAverage) for distributed learning of a neural network. In FedAverage, edge devices receive the network architecture and initial weights from the cloud, then perform multiple steps of stochastic gradient descent to minimize the local empirical loss and send the updated model weights back to the cloud. The cloud then simply takes the average of all received weights and broadcasts the averaged weights for the next round. The process repeats until convergence. Despite its simplicity, FedAverage has seen a lot of success in many applications. In addition, recent years have brought many critical advances in FL, such as enabling faster convergence (Reddi et al. 2021), improving model aggregation schemes on the cloud (Acar et al. 2019; Karimireddy et al. 2020), and promoting fairness (Du et al. 2021; Yue, Nouiehed, and Kontar 2021), amongst many others. An in-depth overview of the current state of FL can be found in Kontar et al. (2021); where the future IoT system characterized by FL as the underlying data analytics approach is termed the Internet of Federated Things (IoFT).

CONTACT Raed Al Kontar *alkontar@umich.edu* Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI.
Supplementary materials for this article are available online. Please go to *www.tandfonline.com/r/TECH*.

Despite those many advances, data heterogeneity remains a significant challenge in FL. Unlike centralized systems, in a decentralized paradigm, data partitions are fixed as they reside at the edge and cannot be changed, shuffled, or randomized. Further, edge devices collect local datasets under diverse operational, environmental, cultural, and socio-economic conditions; thus, contain distinguished patterns. Traditional wisdom is to use a strong global model to fit all the patterns among datasets. However, in FL, such an approach becomes problematic. First, when the local datasets have different or even conflicting patterns, using a single model leads to slow convergence in training and degenerated performance in inference. Indeed, many papers have shown the wide gap in a global model's performance across different devices when heterogeneity exists (Hard et al. 2018; Wang et al. 2019a). Second, FL methods usually require the server to average the updated models with weights determined by dataset size. Therefore, devices with limited data, bandwidth/memory or unreliable connection may not be favored during training. This often results in fairness concerns where performance is satisfactory only on devices with large datasets while it degrades on other devices.

Personalized FL has been proposed to address the heterogeneity issue. Instead of using one global model, edge devices retain their own customized models. By personalization, models combine knowledge both from common trends and individualized characteristics. One group of popular personalization methods encourage the weights of personalized models to stay in a small region in the parameter space of the global model to balance shared knowledge and unique characteristics (Dinh, Tran, and Nguyen 2020; Li et al. 2021). Though such methods obtain improved results compared with global FL, they only handle a concept shift in the input–output relationship across clients, and it is unclear how the closeness in parameter space is related to incorporating unique client's input–output relationships.

In practice, we always want to design personalized FL systems based on the structure of data heterogeneity. For example, besides a change in the input–output relationship, heterogeneity in the input distribution across clients in FL is very common. Clients may observe unique inputs unseen to other clients, while in engineering settings, defects or failures types may vary across devices based on their operational conditions (Kontar et al. 2017). With the above objective in mind, we delve into the data generating process and classify heterogeneity of labeled datasets into heterogeneity in the input space and heterogeneity in the input–output relationship. We adopt a domain adaptation route to handle the input space shift and design a flexible, personalized model with strong local representation power. The model we propose is fully personalized and yet insusceptible to overfitting. We also provide statistical examples whereby traditional models can fail while our approach excels. Extensive simulations from simple curve fitting to classification on large datasets and real-life applications show that our model improves performance, promotes fairness compared with the state-of-the-art, and addresses different types of heterogeneity across client data.

We briefly summarize our contributions below:

- We propose `PFL-DA`: A personalized FL model that exploits domain adaptation to handle heterogeneity in both the input space (a covariate shift) and input–output relationship (a concept shift).
- We theoretically analyze the convergence of our algorithm. We also provide a counterexample showing how traditional personalized FL approaches can fail in the presence of large covariate shifts and highlight the benefit of our model in such instances.
- Experiments show that our method achieves higher average testing accuracy on a wide range of benchmark datasets compared with state-of-the-art personalized models. Our results also indicate that `PFL-DA` promotes fairness across clients.
- We test our approach on a prototype featuring geographically dispersed and connected 3D printers. The case study showcases the advantageous features of our approach and highlights its potential in enabling collaborative model building in manufacturing.

The rest of the article is organized as follows: In Section 1.1, we discuss current literature related to our work. Our personalized modeling approach is presented in Section 2. In Section 3 we provide theoretical guarantees for our model along with counterexamples showing that our model can excel in the presence of heterogeneity while traditional methods can fail. Section 4 provides case studies on various simple and large-scale datasets. We also apply our approach to a distributed and connected 3D printer dataset. Finally, we end with a brief conclusion and a discussion on an interesting open direction in Section 5. The proof of the main theorem and details of numerical simulations are relegated to the supplementary materials.

1.1. Related Work

We first provide an overview of related work. The overview is by no means an exhaustive list. For an in depth overview of FL we refer readers to Kontar et al. (2021).

Global modeling. Currently, most FL methods focus on learning a global model that aims to maximize utility across all devices (McMahan et al. 2017; Li et al. 2018; Acar et al. 2019; Karimireddy et al. 2020; Reddi et al. 2021). One can view the global model as one model that fits all clients, where the goal is to yield better performance in expectation across all devices as opposed to each device learning a separate model using its own data. This approach has seen success in large scale mobile applications (Yang et al. 2018; Wang et al. 2019b; Lim et al. 2020). However, in engineering settings, devices are operated under different environmental conditions, such as different levels of speed, load and temperature (Fang, Paynabar, and Gebraeel 2017). As a result, data will exhibit highly heterogeneous trends and behaviors across devices.

Personalized modeling. A few algorithms have been proposed for personalized modeling in FL (Smith et al. 2017; Arivazhagan et al. 2019; Dinh, Tran, and Nguyen 2020; Liang et al.

2020; Yu, Bagdasaryan, and Shmatikov 2020; Li et al. 2021). One line of work seeks to encourage model weights to stay close to each other in the parameter space. For example, in the train-then-personalize approach, one trains a global model and then fine-tunes it to individual devices (Yu, Bagdasaryan, and Shmatikov 2020). Also, by similar motivations, `pedme` (Dinh, Tran, and Nguyen 2020), and `Ditto` (Li et al. 2021) allow each model to have individualized weights and penalizes the digression from individual weights to central weights by regularization. One representative work is `Ditto`, which adds regularization terms as the L_2 norm of the distance between individualized weights and global weight trained by FedAverage. Though attaining improved performance, these models usually ignore the structure of data and only aim at tackling concept shifts. *We will show a counterexample where none of these algorithms can extract useful shared knowledge from distributed datasets*.

Another popular personalization technique is parameter sharing (Arivazhagan et al. 2019; Liang et al. 2020). Modern deep neural networks contain multiple layers. One can personalize some of them and keep others the same among clients. Among them, LG-Fedavg (Liang et al. 2020) trains personalized networks for local feature representation and shared networks for global decision-making. Multi-task federated learning (Smith et al. 2017) personalizes top layers for different tasks. Such models pose strong structural assumptions on data which limit representation power. With such structural assumptions, coercive weight sharing may result in models underperforming compared to separately learning using their own data. This phenomenon is commonly referred to as negative transfer of knowledge (Kontar, Raskutti, and Zhou 2020).

Fairness in federated learning. Current approaches to Fair FL aim to reduce the discrepancy in model performance among clients, that is, encourage models to perform similarly among all clients. One useful technique to promote fairness is to design special loss functions that attach higher importance to low-performance agents (Li et al. 2020a; Yue, Nouiehed, and Kontar 2021).

For instance, Du et al. (2021) propose a minimax optimization framework called agnostic federated learning (AFL) to optimize the worst weighted combination of local devices. Li et al. (2019) propose q-Fair federated learning (q-FFL). q-FFL reweights loss functions such that devices with poor performance will be given relatively higher weights in the weighted sum of global objectives. Yue, Nouiehed, and Kontar (2021) estimates the weighting coefficients by statistical ordering and promotes both group level and individual level fairness. The approaches above establish a global model. In contrast, recent work has tried to improve fairness through personalization. Through careful personalization, `Ditto` (Li et al. 2021) achieves better fairness and robustness at the same time. Improved fairness through personalization is also observed in Wang et al. (2019a). The rationale is that personalization may help the performance of clients where the global model usually performs poorly. Our work also investigates along the line of exploiting personalization for improved fairness.

Domain adaptation. Domain adaptation seeks to extract common features from inputs in different domains to be used for prediction (Ganin and Lempitsky 2015; Tzeng et al. 2017; Gulrajani and Lopez-Paz 2020). Common techniques involve training a function that extracts features from different domains, and then predictions are obtained through the training of a decoder on the features. Such models are trained by minimizing the summation of empirical loss on all domains. Since the decoder is global, ideally, it does not contain information unique to domains, and featurizers should fully adapt to different domains. Therefore, to ensure featurizers learn useful common data representations, several tricks have been proposed. For example, some works use adversarial training to guarantee that the learned features are domain-invariant (Ganin and Lempitsky 2015). Similarly, CORAL (Sun and Saenko 2016) matches the feature mean and variance of different domains to make these features similar. Since federated datasets are usually from the same task but collected by geographically separated devices, it is natural to assume they are from different domains.

2. Model Description

We consider a supervised learning setting. Suppose we have N edge devices, each with a local labeled dataset $\boldsymbol{D}_i = \{(x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2}) \dots (x_{i,n_i}, y_{i,n_i})\}$, where n_i is the size of the ith dataset. The total number of observations is denoted by $n = \sum_{i=1}^{N} n_i$.We assume that local data are examples from space $\mathcal{X}_i \times \mathcal{Y}$, where $\mathcal{X}_i$ is the input space for client i and $\mathcal{Y}$ is $\mathbb{R}$ for regression and categorical for classification. We also let $f_i(x) : \mathcal{X}_i \to \mathcal{Y}$ denote the personalized model to be learned for client i.

For client i, the joint distribution of input–output tuple (x, y) is $\mathbb{P}^i_{x,y} = \mathbb{P}^i_x \mathbb{P}^i_{y|x}$. As aforementioned, current literature focuses predominantly on a concept shift where $x_i \sim \mathbb{P}_x$ is the same for all clients, yet the conditional distribution $\mathbb{P}^i_{y|x}$ varies across clients. Typically, this is modeled as $y_i = f_{\boldsymbol{\theta}_i}(x_i)$ where clients use f from the same function class (linear models, neural networks) yet with different parameters. Here $f_i \triangleq f(x_i) \triangleq f_{\boldsymbol{\theta}_i}(x_i)$.

However, a covariate shift where $\mathbb{P}^i_x$ is different across clients is not uncommon in real applications (Kontar et al. 2021). Indeed, FL's decentralized nature and its ability to reach diverse clients with different external conditions can lead to datasets with both a concept and covariate shift. Take a simple engineering example whereby a product design can differ from one manufacturer to another. In this article, we adopt domain adaptation to personalized FL. We first start by a simple domain adaptation model that tackles a covariate shift and then use this model to build `PFD-DA` which has increased representation power and tackles both types of data heterogeneity.

2.1. Simple Domain Personalization

We start by tackling a covariate shift. The fundamental idea is to exploit a representation function $\Phi_i : \mathcal{X}_i \to \mathcal{H}$ to map inputs into the same feature space $\mathcal{H}$. Then a globalized decoder g obtains predictions based on the features. More specifically, the mapping from input space $\mathcal{X}_i$ to output space $\mathcal{Y}$ is decomposed into the composition of a featurerizer $\Phi_i : \mathcal{X}_i \to \mathcal{H}$ from the

input into a feature space, and a decoder $g : \mathcal{H} \to \mathcal{Y}$ from the feature to the output space.

More formally, the model is given as

$$f_{\boldsymbol{\theta}_i}(x) = g_{\boldsymbol{w}}(\Phi_{\boldsymbol{\beta}_i}(x)), \tag{1}$$

where $\boldsymbol{\theta}_i = \{\boldsymbol{w}, \boldsymbol{\beta}_i\}$ denotes the set of parameters that parameterize g and Φ_i, respectively. Notice here that $\boldsymbol{w}$ is a set of global shared parameters across all clients while each client retains their own $\boldsymbol{\beta}_i$.

To learn $g_{\boldsymbol{w}}$ and $\Phi_{\boldsymbol{\beta}_i}$, FL minimizes the risk over all clients:

$$\min_{\boldsymbol{w},\,\boldsymbol{\beta}_i} \sum_{i=1}^{N} p_i F_i^{\text{pop}}(\boldsymbol{w}, \boldsymbol{\beta}_i) \triangleq \min_{\boldsymbol{w},\,\boldsymbol{\beta}_i} \sum_{i=1}^{N} p_i \mathbb{E}_{(x,y)\sim\mathbb{P}_{x,y}^{i}} \ell\big(y, g_{\boldsymbol{w}}(\Phi_{\boldsymbol{\beta}_i}(x))\big), \tag{2}$$

where $F_i^{\text{pop}}(\boldsymbol{w}, \boldsymbol{\beta}_i)$ is the population risk function for client i, ℓ is some loss function such as mean-square error for regression tasks or cross-entropy for classification tasks and p_i is a client weight given as $p_i = \frac{1}{N}$. Clearly, since $\mathbb{P}_{x,y}^{i}$ is not known, F_i^p is approximated by the empirical risk $F_i = \frac{1}{n_i}\sum_{(x_{i,j},y_{i,j})\in\boldsymbol{D}_i} \ell\left(y_{i,j}, g_{\boldsymbol{w}}(\Phi_{\boldsymbol{\beta}_i}(x_{i,j}))\right)$. Ideally, if trained effectively $\Phi_{\boldsymbol{\beta}_i}$ can capture the covariate shift in the data and $g_{\boldsymbol{w}}$ learns the mapping from the feature space to the output. Hereon we refer to the model in (1) as simple domain adaptation—`Simple-DA`.

2.2. Flexible Domain Personalization

Notice that (1) only models a domain shift. We bridge the gap by also individualizing g as $g_{\boldsymbol{\gamma}_i}$ to characterize concept shifts. $\boldsymbol{\gamma}_i$ denotes client specific weights that parameterize the mapping $g_i \triangleq g_{\boldsymbol{\gamma}_i} : \mathcal{H} \to \mathcal{Y}$. This is given as

$$f_{\boldsymbol{\theta}_i}(x) = g_{\boldsymbol{\gamma}_i}(\Phi_{\boldsymbol{\beta}_i}(x)) \tag{3}$$

The model in (3) is fully personalized for increased flexibility in modeling heterogeneity. Ideally, $\Phi_{\boldsymbol{\beta}_i}(x)$ should handle the covariate shift and $g_{\boldsymbol{\gamma}_i}$ should handle the concept shift.

With this goal in mind, we use a bi-level optimization formulation to solve $\Phi_{\boldsymbol{\beta}_i}(x)$ and $g_{\boldsymbol{\gamma}_i}$ separately. Client i optimizes the objective:

$$\min_{\boldsymbol{\gamma}_i} \widetilde{F}_i(\boldsymbol{\gamma}_i, \boldsymbol{\beta}_i, \boldsymbol{w}) \tag{4}$$

under the constraint that

$$\left\{\boldsymbol{w}, \{\boldsymbol{\beta}_i\}_{i=1}^{N}\right\} \in \arg\min_{\widetilde{\boldsymbol{w}},\{\widetilde{\boldsymbol{\beta}}_j\}_{j=1}^{N}} \sum_{j=1}^{N} p_j F_j(\widetilde{\boldsymbol{\beta}}_j, \widetilde{\boldsymbol{w}}) \tag{5}$$

where p_i is still set to $\frac{1}{N}$. F_i is a local empirical risk of function (1) on client i:

$$F_i(\boldsymbol{\beta}_i, \boldsymbol{w}) = \frac{1}{n_i} \sum_{(x_{i,j},y_{i,j})\in\boldsymbol{D}_i} \ell\left(y_{i,j}, g_{\boldsymbol{w}}(\Phi_{\boldsymbol{\beta}_i}(x_{i,j}))\right)$$

and $\widetilde{F}_i$ is a regularized local empirical risk that learns (3) while penalizing deviations from parameters $\boldsymbol{w}$ of the global decoder $g_{\boldsymbol{w}}$.

$$\widetilde{F}_i(\boldsymbol{\gamma}_i, \boldsymbol{\beta}_i, \boldsymbol{w}) = \frac{1}{n_i} \sum_{(x_{i,j},y_{i,j})\in\boldsymbol{D}_i} \ell\left(y_{i,j}, g_{\boldsymbol{\gamma}_i}(\Phi_{\boldsymbol{\beta}_i}(x_{i,j}))\right) + \lambda_1 \left\|\boldsymbol{\gamma}_i - \boldsymbol{w}\right\|^2.$$

We will briefly explain how encoder and decoder functions are trained to handle covariate and concept shift, respectively. In (5), the encoders $\Phi_{\boldsymbol{\beta}_i}$'s learn to map inputs across different domains into the same feature space with the help of $g_{\boldsymbol{w}}$. Since in (5) the decoder $g_{\boldsymbol{w}}$ is shared by all clients, the mapping from the feature space to the output is the same for all clients. Thus, to fit training data, the encoders $\Phi_{\boldsymbol{\beta}_i}$'s are forced to output features in similar distributions (i.e., domain invariant distributions). When the encoders are properly trained, features encoded by encoder functions follow similar distributions. Then to map these features into outputs and to handle concept shifts, we train personalized decoder functions as well. Since the features are now domain invariant, we can use regularization-related personalization schemes to handle the concept shifts. $g_{\boldsymbol{w}}$ learned from (5) represents the common patterns of decoder functions, thus, the ridge penalty term facilitates knowledge sharing of the decoder function. In the inference stage, we use the personalized $g_{\boldsymbol{\gamma}_i}$'s.

The hyperparameter λ_1 controls the level of knowledge transfer: a larger λ_1 penalizes the deviation between $\boldsymbol{\gamma}_i$ and $\boldsymbol{w}$ more severely, thus, making the model closer to simple domain adaptation in (1). While a smaller λ_1 allows the decoder function $g_{\boldsymbol{\gamma}_i}$ to have greater individuality. Our model thus balances between a completely individualized model and `Simple-DA`.

2.3. Training Algorithm

So far we have introduced the general model in (3). In this section we provide a federated algorithm for learning the global parameters $\boldsymbol{w}$ and local parameters $\{\boldsymbol{\beta}_i, \boldsymbol{\gamma}_i\}$, via an alternating descent approach. Our approach only requires sharing the updated weights $\boldsymbol{w}$ from a global decoder $g_{\boldsymbol{w}}$ with the cloud. The general framework is shown in Algorithm 1, while the functions within Algorithm 1 are presented in Algorithms 2 and 3.

Our approach uses T communication rounds, where a communication round is the process that the central server/cloud sends an updated set of weights $\boldsymbol{w}^t$ for $t \in [T]$ to the edge devices, which update the weights and send them back to the server. In communication round t, the cloud broadcasts global weights $\boldsymbol{w}^t$ to all edge devices. The edge devices then take $\boldsymbol{w}^t$ as their initialization and update $\boldsymbol{w}$ and $\boldsymbol{\beta}_i$ on the local empirical loss F_i by performing several steps of gradient descent or adaptive stepsize methods. After obtaining an updated solution $(\boldsymbol{w}_i^{t+1}, \boldsymbol{\beta}_i^{t+1})$, $\boldsymbol{\gamma}_i^t$ is updated using regularized loss $\widetilde{F}_i$. At the

Algorithm 1 Model Training

Input: Client datasets $\{\boldsymbol{D}_i\}_{i=1}^N$, T, initialization for $\{\boldsymbol{\beta}_i^0\}_{i=1}^N$, $\{\boldsymbol{\gamma}_i^0\}_{i=1}^N$ and $\boldsymbol{w}^0$
for $t = 1, 2, \ldots T$ **do**
  Server broadcasts model weight $\boldsymbol{w}^t$ to clients C_q
  for each client i **do**
    $\boldsymbol{\beta}_i^{t+1}$, $\boldsymbol{w}_i^{t+1}$ = `local_update`$[\boldsymbol{w}^t, \boldsymbol{D}_i]$
    $\boldsymbol{\gamma}_i^{t+1}$ = `local_update_regularized`$[\boldsymbol{w}^t, \boldsymbol{\beta}_i^t, \boldsymbol{D}_i]$
    Client i sends $\boldsymbol{w}_i^{t+1}$ to server
  end for
  $\boldsymbol{w}^{t+1}$ =`server_update` $[\boldsymbol{w}_1^{t+1}, \ldots, \boldsymbol{w}_N^{t+1}]$.
end for

Algorithm 2 `local_update`

1: **Input:** $\{\boldsymbol{D}_i\}_{i=1}^N$, η_t, $\boldsymbol{w}_i^t$, and $\boldsymbol{\beta}_i^t$.
2: Initialize $\boldsymbol{w}_i^{t,0} = \boldsymbol{w}_i^t$, and $\boldsymbol{\beta}_i^{t,0} = \boldsymbol{w}_i^t$
3: **for** $q = 0, 1, \ldots E-1$ **do**
4: $\quad \boldsymbol{w}_i^{t,q+1} = \boldsymbol{w}_i^{t,q} - \eta_t \nabla_{\boldsymbol{w}} F_i(\boldsymbol{w}_i^{t,q}, \boldsymbol{\beta}_i^{t,q})$
5: $\quad \boldsymbol{\beta}_i^{t,q+1} = \boldsymbol{\beta}_i^{t,q} - \eta_t \nabla_{\boldsymbol{\beta}_i} F_i(\boldsymbol{w}_i^{t,q}, \boldsymbol{\beta}_i^{t,q})$
6: **end for**
7: Return $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^{t,E}$, and $\boldsymbol{\beta}_i^{t+1} = \boldsymbol{\beta}_i^{t,E}$

Algorithm 3 `local_update_regularized`

1: **Input:** Client datasets $\{\boldsymbol{D}_i\}_{i=1}^N$, stepsize η_t, $\boldsymbol{w}^t$, $\boldsymbol{\gamma}_i^t$, and $\boldsymbol{\beta}_i^t$.
2: Set initialization $\boldsymbol{\gamma}_i^{(t,0)} = \boldsymbol{\gamma}_i^t$
3: **for** $q = 0, 1, \ldots E-1$ **do**
4: $\quad \boldsymbol{\gamma}_i^{t,q+1} = \boldsymbol{\gamma}_i^{t,q} - \eta_t \nabla_{\boldsymbol{\gamma}_i} \widetilde{F}_i(\boldsymbol{w}_i^t, \boldsymbol{\beta}_i^t, \boldsymbol{\gamma}_i^{t,q})$
5: **end for**
6: Return $\boldsymbol{\gamma}_i^{t+1} = \boldsymbol{\gamma}_i^{t,E}$

end of local training, only global weights $\boldsymbol{w}_i^{t+1}$ are sent back to the central server. The server then uses `server_update` to calculate the aggregated global weights $\boldsymbol{w}^{t+1}$.

In Algorithm 1, `local_update` is the clients update function to optimize (5). An implementation is shown in Algorithm 2. Similarly, `local_update_regularized` is designed to optimize (4) and an implementation is shown in Algorithm 3.

Further, in this article for `server_update` we exploit FedAverage (McMahan et al. 2017) as the aggregation scheme for global parameters $\boldsymbol{w}$: $\boldsymbol{w}^{t+1} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{w}_i^{t+1}$. However, any alternative aggregation scheme can be readily plugged in.

Also, Algorithm 1 is under full-device participation to simplify notations. However, our training procedure can be readily extended to partial device participation.

In the next section, we show that Algorithm 1 converges to fixed points of our bi-level objective with sufficient communication rounds. We then provide counterexamples where existing personalized models fail while `PFL-DA` can handle such instances.

3. Theoretical Analysis

In this section, we start by showing the convergence of Algorithm 1 under a general nonconvex objective. Then we introduce some simple examples to demonstrate our model's strengths.

3.1. Convergence

Though local solvers `local_update` and `local_update_regularized` can use any optimization algorithm, we study gradient descent to understand the convergence behavior of Algorithm 1. Specifically, at communication round t, client i runs E steps of GD on F_i to update $\boldsymbol{w}$ and $\boldsymbol{\beta}_i$, then the clients runs E steps of GD on $\widetilde{F}_i$.

We use $\boldsymbol{w}_i^{t,q}$ to denote the global weight update for client i, at communication round t, and local GD iterate q. The average of $\boldsymbol{w}_i^{t,q}$ is denoted as $\hat{\boldsymbol{w}}^{t,q} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{w}_i^{t,q}$. For each communication round t, local update index q can range from 0 to $E-1$. As an initialization, at the 0th iterate, $\boldsymbol{w}_i^{t,0} = \hat{\boldsymbol{w}}^{t,0}$ for every client i. At the Eth iterate all weights are averaged again $\hat{\boldsymbol{w}}^{t+1,0} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{w}_i^{t,E}$. Notice that the variable $\hat{\boldsymbol{w}}$ is actually calculated only when $q = 0$. Local variables $\boldsymbol{\beta}_i^{t,q}$, and $\boldsymbol{\gamma}_i^{t,q}$ are defined similarly. We use $\nabla_{\boldsymbol{v}} F(\boldsymbol{v})$ to denote the gradient of F with respect to vector $\boldsymbol{v}$, keeping all other variables constant.

The following theorem shows that our algorithm converges under very general conditions.

Theorem 3.1. Under assumptions:

1. All local objectives F_i are gradient Lipschitz continuous with constant L,
2. The norm of gradient of F_i and $\widetilde{F}_i$ over $\boldsymbol{w}$, $\boldsymbol{\beta}_i$'s and $\boldsymbol{\gamma}_i$'s are all bounded by G^2
3. The function values of $\sum_{i=1}^N F_i$ and $\sum_{i=1}^N \widetilde{F}_i$ are lower bounded by F^* and $\widetilde{F}^*$.

Then when we Algorithms 1–3 with a diminishing stepsize $\eta_t = \frac{\eta_1}{\sqrt{t}}$, the following holds:

$$\min_{t\in\{1,\ldots T\}, q\in\{0,\ldots,E-1\}} \left[\left\| \sum_{i=1}^N p_i \nabla_{\boldsymbol{w}} F_i(\hat{\boldsymbol{w}}^{t,q}, \boldsymbol{\beta}_i^{t,q}) \right\|^2 + \sum_{i=1}^N p_i \left\| \nabla_{\boldsymbol{\beta}_i} F_i(\hat{\boldsymbol{w}}^{t,q}, \boldsymbol{\beta}_i^{t,q})) \right\|^2 \right] \leq O\left(\frac{\log T}{\sqrt{T}}\right). \tag{6}$$

And

$$\min_{t\in\{1,\ldots T\}, q\in\{0,\ldots,E-1\}} \left[p_i \sum_{i=1}^N \left\| \nabla_{\boldsymbol{\gamma}_i} \widetilde{F}_i(\hat{\boldsymbol{w}}^{t,q}, \boldsymbol{\beta}_i^{t,q}, \boldsymbol{\gamma}_i^{t,q})) \right\|^2 \right] \leq O\left(\frac{\sqrt{\log T}}{T^{\frac{1}{4}}}\right). \tag{7}$$

Theorem 3.1 shows that global weights $\boldsymbol{w}$ converge to critical points of the global loss $\sum_{i=1}^N p_i F_i(\boldsymbol{w}, \boldsymbol{\beta}_i)$, and local weights $(\boldsymbol{\beta}_i, \boldsymbol{\gamma}_i)$ converge to critical points of the local (regularized and unregularized) losses F_i and $\widetilde{F}_i$. This matches our intuition that the global model should learn common knowledge, while local models should also learn from local datasets. The detailed proof of Theorem 3.1 is relegated to the supplementary materials. We also note that our results are more general than current personalized FL (Li et al. 2021) results as we do not assume the convexity of the objective. The $O\left(\frac{\log T}{\sqrt{T}}\right)$ convergence rate is standard for nonconvex optimization with diminishing stepsize algorithms (Ghadimi and Lan 2013; Yue, Nouiehed, and Kontar 2021). The $O\left(\frac{\sqrt{\log T}}{T^{\frac{1}{4}}}\right)$ rate is slightly slower, which is a price we pay for personalization.

3.2. Mathematical Proof of Concept

In this section, we use a simple mathematical example to demonstrate the strengths of our model in datasets with covariate shifts. We assume client i's ground truth is a sine function,

$f_i(x) = \alpha_i \sin(2\pi(x+\theta_i))$, where α_i and θ_i are independent random variables that model concept and covariate shifts, respectively. We assume θ_i follows a uniform distribution on $[0, 1]$ and the distribution of α_i has mean μ_α and variance σ_α^2. Basically, devices have similar trends yet with a phase shift (often common in vibration signals). Though sine functions are simple, they are the basis of many periodic functions. If we train a global model to minimize the population risk $\min_w \mathbb{E}_i[||f_{\boldsymbol{w}} - f_i||_2^2]$, where $f_{\boldsymbol{w}}$ is a global model with $\boldsymbol{w}$ as weights and $||\cdot||_2$ is a functional norm defined as $||f||_2^2 = \int_0^1 f(x)^2 dx$. Then $f_{\boldsymbol{w}}$ should minimize:

$$\begin{aligned}
&\arg\min_{f_w} \mathbb{E}_{\alpha_i,\theta_i}\left[\int_0^1 \left(f_{\boldsymbol{w}}(x) - \alpha_i \sin(2\pi x + 2\pi\theta_i)\right)^2 dx\right] \\
&= \arg\min_{f_w} \mathbb{E}_{\alpha_i,\theta_i}\left[\int_0^1 f_{\boldsymbol{w}}(x)^2 - 2f_{\boldsymbol{w}}(x)\alpha_i \sin(2\pi x + 2\pi\theta_i) dx\right] \\
&= \arg\min_{f_w} \mathbb{E}_{\alpha_i,\theta_i}\left[\int_0^1 f_{\boldsymbol{w}}(x)^2 dx\right] = \arg\min_{f_w}\left[\int_0^1 f_{\boldsymbol{w}}(x)^2 dx\right].
\end{aligned}$$

Therefore, the unique minimizer is $f_{\boldsymbol{w}}(x) = 0$ for every x in $[0, 1]$. Clearly, a global model does not learn anything from such a dataset where there is a phase shift in the sine function. Now suppose there exists a set of weights $\boldsymbol{w}_{\text{zero}}$ such that $f_{\boldsymbol{w}_{\text{zero}}} = 0$. We then examine the performance of several personalized FL models on this problem.

On the sine regression problem, `Ditto`'s local objective becomes a local risk plus a regularization: $\int_0^1 \left[f_{\boldsymbol{\gamma}_i}(x) - \alpha_i \sin(2\pi x + 2\pi\theta_i)\right]^2 dx + \lambda||\boldsymbol{\gamma}_i - \boldsymbol{w}_{\text{zero}}||^2$, $f_{\boldsymbol{w}_{\text{zero}}} = 0$. Then the optimal solution is the same as training a completely individualized model with L_2 regularization. Not only no useful shared information is learned. The regularization will further exacerbate the problem by forcing $\boldsymbol{\gamma}_i$ close to $\boldsymbol{w}_{\text{zero}}$ which is clearly unsatisfactory as it predicts a zero everywhere. Another intuitive and popular personalization method is train-then-optimize. For instance in Yu, Bagdasaryan, and Shmatikov (2020), clients collaboratively train one global model, then adapt it to local datasets for personalization. However, since the optimal global model is a constant zero function, the personalization starts from a possibly bad initialization. As seen above, the phase shift across sinusoidal patterns prevents central training from extracting useful information leading to a wrong and uninformative global model. This invalidates the justification for local training via distance-based regularization from such an uninformative global model.

On the other hand, `PFL-DA` can address this challenge simply by its construction where the featurizer $\Phi_i \triangleq \Phi_{\boldsymbol{\beta}_i}(x)$ only needs to learn a phase shift which, here, can be fully represented in just one parameter θ_i. Mathematically, we can consider the parametric model where $\Phi_{\boldsymbol{\beta}_i}(x) = x + \boldsymbol{\beta}_i$ and the decoder function $g_{\boldsymbol{w}}$ is a square integrable function parameterized by $\boldsymbol{w}$. We assume the learned $\boldsymbol{\beta}_i$ is only a function of θ_i, $\boldsymbol{\beta}_i = \boldsymbol{\beta}(\theta_i)$, since encoder functions should handle the covariate shifts.

Under such parameterization, we can show that the optimal solution to (5) is $\boldsymbol{\beta}(\theta_i) = \theta_i$ and $g_{\boldsymbol{w}}(\phi) = \mu_a \sin(2\pi\phi)$. *Therefore, the global decoder function $g_{\boldsymbol{w}}$ is a sine function whose amplitude is the average amplitude μ_a over different clients. Hence, it can be regarded as the "center" of all decoder functions.* Accordingly, if we parameterize $g_{\boldsymbol{\gamma}_i}$ also to be sine functions: $g_{\boldsymbol{\gamma}_i}(\phi) = \boldsymbol{\gamma}_i \sin(2\pi\phi)$, the optimal solution to (4) is

$$\boldsymbol{\gamma}_i = \frac{\alpha_i + 2\lambda_1\mu_a}{1 + 2\lambda_1},$$

which is a weighted average of α_i and μ_α. Here one can observe the interplay in $\widetilde{F}_i$ where the first term encourages model fit and the regularization encourages resemblance to the global decoder ($g_{\boldsymbol{w}}$). This is indeed reminiscent of the prior/posterior interplay in Bayesian statistics where $g_{\boldsymbol{w}}$ plays the role of a prior for $g_{\boldsymbol{\gamma}_i}$.

Therefore, the analysis of sine functions confirms our motivations that (4) and (5) can handle both covariate and concept shift. The detailed derivations are relegated to the supplementary materials. In Section 4, we revisit sinusoidal functions and highlight the above ideas via simulations.

4. Numerical Experiments

4.1. Numerical Proof of Concept

We examine the performance of `PFL-DA` on the sine data proposed in Section 3.2. Our target functions are shifted sine functions with amplitudes uniformly sampled from $[0.8, 1.2]$. We simulate 500 clients, with 490 clients containing 100 training points each, and 10 clients containing 5 training points each. The ground truth of every client is a sine function with a unique phase shift sampled from $[-\frac{1}{2}, \frac{1}{2}]$. To recover the encoder and decoder functions from data, we define the encoder function to be $\Phi_{\boldsymbol{\beta}_i}(x) = \Pi(x + \text{NeuralNet}_1(1; \boldsymbol{\beta}_i))$, where $\text{NeuralNet}_1(\cdot; \boldsymbol{\beta}_i)$ is a 4-layer fully connected neural network parameterized by $\boldsymbol{\beta}_i$. The operator Π is defined as $\Pi(\phi) = \phi - [\phi]$, where $[\phi]$ is the closest integer to ϕ. We use Π to remove the periodicity of the sine function by mapping the feature ϕ onto the interval $[-\frac{1}{2}, \frac{1}{2}]$. Also, we define the decoder function as a 6-layer neural network $g_{\boldsymbol{w}}(\phi) = \text{NeuralNet}_2(\phi; \boldsymbol{w})$. $g_{\boldsymbol{\gamma}_i}$ has the same architecture as $g_{\boldsymbol{w}}$, but the parameters (weights and biases) are different. Ideally, $\text{NeuralNet}_1(1; \boldsymbol{\beta}_i)$ should learn the domain shift of each client, and $\text{NeuralNet}_2(\phi; \boldsymbol{w})$ should learn the sine function.

To evaluate performance, we benchmark with `Ditto`, `PFL-DA` and `indiv`. In `indiv` we train models on different devices separately without any interaction among devices. To evaluate the testing error, we first calculate the mean square error between ground truth and model prediction on 100 equally spaced points on [0,1] for each client with five training points, then average the error on all of them. We randomly generate the sine functions and perform the simulation for six times to estimate the deviation of average testing error. To visualize the fittings of different algorithms, we plot training data and model predictions, as well as $\Phi_{\boldsymbol{\beta}_i}$'s in Figure 1.

From the first row of Figure 1, we can see that `Ditto` cannot fit data well, possibly due to the negative transfer of knowledge as we discussed in Section 3.2. `Indiv` can fit training data, but cannot learn from other clients thus the prediction on data-sparse region has an irregular shape. `PFL-DA` is able to fit data and learn the patterns of the sine function. The second row of Figure 1 shows the decoder functions on different clients. If the modeling is appropriate, all the decoder functions should learn the same sine function. This is indeed the case for `PFL-DA`, where decoders from all clients exhibit very similar sinusoidal patterns. *Such patterns are learned purely from data*. However,

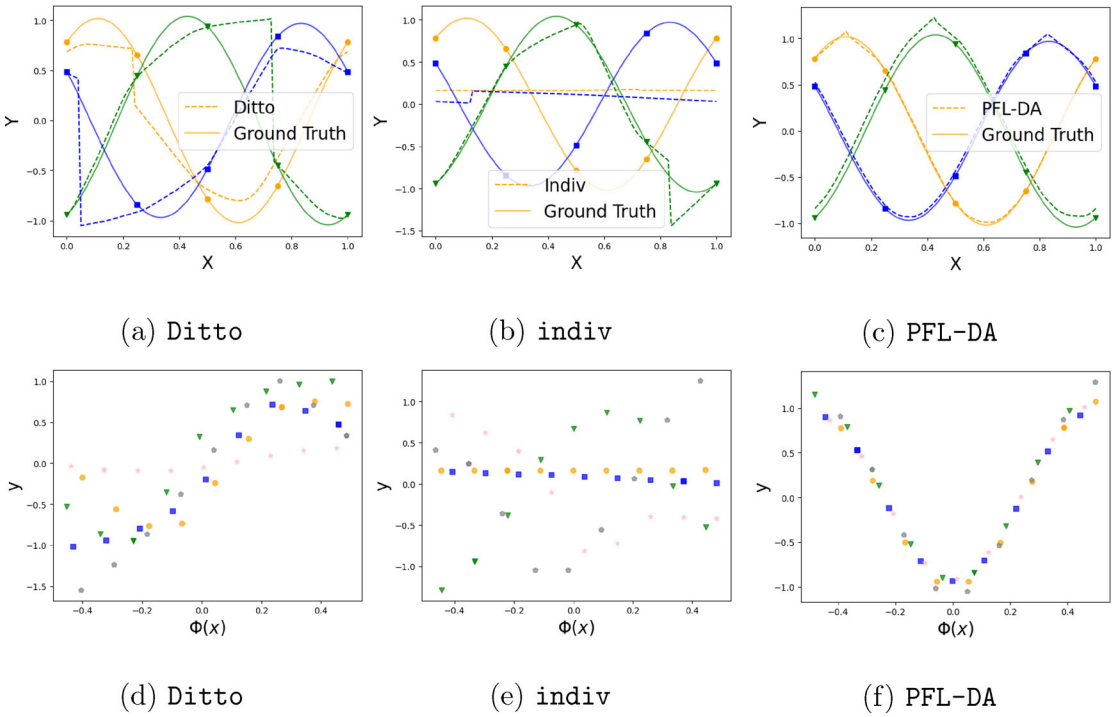


Figure 1. Regression of sine functions by three algorithms. In the first row, we randomly select three clients and show the training data as dots and model prediction as dashed lines. Different colors and markers denote different clients. In the second row, we illustrate the learned decoder functions by plotting output y versus feature $\Phi(x)$.

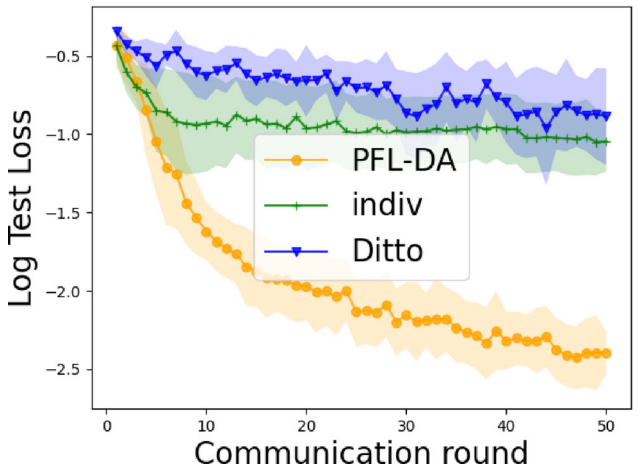


Figure 2. The learning curves of sine functions. We run six independent experiments for each algorithm and plot the mean and 95% confidence interval of logarithm of the test loss.

for `Ditto` and `indiv`, the learned decoders do not have clear collective trends, suggesting that no useful information is shared. The decoder patterns confirm our analysis that `PFL-DA` can learn useful information among clients while `Ditto` and `indiv` cannot for the sine function regression.

We also plot the learning curves of the three algorithms in Figure 2 to visualize the training process. After around 20 communication rounds, the test loss of `PDA-FL` decreases steadily, while that of `indiv` and `Ditto` remains high during the course of training. `PFL-DA` achieves the lowest test error after sufficient communication rounds. The results confirm the superior performance of `PFL-DA` and its ability to borrow strength from other clients while retaining individualized features.

4.2. Case Studies

Next we test our model on multiple datasets. A central directory for the datasets can be found in IoFT datasets (2021). Our benchmark algorithms are:

- `FedAverage` (McMahan et al. 2017): Used to benchmark with the performance of global modeling.
- `Indiv`: Completely individualized training where clients only train on their local data. This model will shed light on the extent of knowledge sharing across clients.
- `TP` (Yu, Bagdasaryan, and Shmatikov 2020): A train-then-personalize approach where a global model using `FedAverage` is learned and then it is fine-tuned on the clients' local data.
- `Ditto` (Li et al. 2021): `Ditto` is an iterative train-then-personalize procedure.
- `pfedme` (Dinh, Tran, and Nguyen 2020): `pfedme` is a personalized FL approach that uses Moreau envelope to update global parameters and personalize.
- `Simple-DA`: The simple domain adaptation model introduced in (1). We note, in essence the model introduced by Liang et al. (2020) can be posed in manner similar to `Simple-DA`.
- `PFL-DA`: Our proposed personalized domain adaptation model for addressing both concept and covariate shifts in FL.

4.2.1. Setup

We consider five popular image datasets: ColoredMNIST, RotatedMNIST, VLCS, PACS, and FEMNIST. These datasets contain images whose labels we should classify. Unlike ordinary image classification datasets like CIFAR or MNIST, the images

Table 1. Sample images from case study datasets.

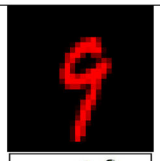
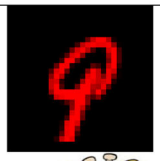
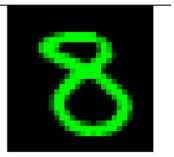
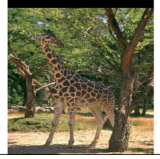
Dataset	Client 1	Client 2	Client 3	Client 4
ColoredMNIST				
PACS				

Table 2. Average testing accuracies.

Dataset	FedAverage	Indiv	TP	Ditto	Simple-DA	pfedme	PFL-DA
CMNIST	68.8±0.2	75.6±0.2	54.5±0.1	71.8±0.2	**75.6**±0.2	56.6 ± 1.4	**75.8**±0.1
RMNIST	93.8±0.4	**98.4**±0.1	93.9±0.2	93.9±0.2	**98.4**±0.1	91.4 ± 0.6	**98.4**±0.1
FEMNIST	77.9±0.3	61.7 ±0.3	77.7±0.3	**80.2**±0.3	46.0±3	78.6 ± 9.6	**80.8**±0.2
VLCS	82.8±0.3	82.5±0.3	82.7±0.3	82.4±0.3	82.6±0.1	43.8 ± 0.1	**83.7**±0.1
PACS	84.4±0.8	93.9±0.4	85.0±1.9	92.7±0.2	**94.4**±0.5	91.7 ± 0.2	**95.6**±0.1

NOTE: Bold values indicates the highest accuracy results that are statistically significantly better than the rest.

Table 3. Standard deviation of accuracies among clients.

Dataset	FedAverage	Indiv	TP	Ditto	Simple-DA	pfedme	PFL-DA
CMNIST	**1.1**±0.1	12.8±0.1	7.6±0.4	18.0±0.2	13.0±0.2	5.1 ± 0.8	12.8±0.5
RMNIST	5.5±0.4	**0.3**±0.1	5.4±0.2	5.7±0.2	**0.2**±0.1	8.0±0.2	**0.2**±0.1
FEMNIST	11.0±0.1	11.0±0.1	11.1±0.1	**8.3**±0.1	15.5±0.1	11.0 ± 0.1	8.7±0.1
VLCS	11.8±0.2	10.8±0.2	11.9±0.1	11.7±0.1	11.0±0.3	**2.6**±0.1	10.0±0.2
PACS	8.2±0.1	**1.6**±0.3	6.7±0.8	**1.1**±0.2	1.9±0.1	3.0±0.1	**1.3**±0.3

NOTE: Bold values indicates the highest accuracy results that are statistically significantly better than the rest.

from these datasets come from heterogeneous domains. Among them, ColoredMNIST, RotatedMNIST, VLCS, and PACS are datasets widely used for domain adaptation tasks, and FEMNIST is a popular FL dataset. Below we provide a brief overview of the datasets along with an illustrative image of the data and different domains in Table 1. In depth details on the setup are deferred to the supplementary material.

- ColoredMNIST: We separate the popular hand-written digit dataset MNIST into 100 clients and color pictures on one client with green or red according to the labels.
- RotatedMNIST: We separate MNIST into six clients. On each client, we rotate all images by a certain angle in $[0, \alpha_{\max}]$. The domains are different as images have different orientations. Intuitively, when $\alpha_{\max}$ becomes larger, domain shifts grow.
- VLCS VLCS (Albuquerque et al. 2019) is a realistic dataset from four domains. Each domain contains images from five classes. We assume there are four clients, with each one corresponding to one unique domain.
- PACS Similarly, PACS (Asadi, Hosseinzadeh, and Eftekhari 2019) is a dataset with four domains. Each domain has images from seven classes. We also assume that one domain corresponds to one client.
- FEMNIST FEMNIST (Caldas et al. 2019) is an FL dataset with images of digits (0–9) and English characters (A-Z, a-z) with 62 classes. It is naturally split over clients where each client has data representing their handwritten digits and characters.

As shown in Table 1, the images come from dissimilar domains, but contain common information than can be exploited for a stronger predictive model. As such, we can naturally use PFL-DA to borrow strength across clients. To model domain shifts, we set $\Phi_{\boldsymbol{\beta}_i}$'s to be convolutional neural networks (CNN). The convolutional layers have different parameters on different clients, thus, can potentially extract features from different domains. In ColoredMNIST, RotatedMNIST, and FEMNIST we train a 4-layer CNN from initialization. For PACS and VLCS, we start from a pretrained Resnet50. We also use a CNN for the decoder function $g_{\boldsymbol{\gamma}_i}$'s. Due to dataset size, we use Adam (Kingma and Ba 2015) as our stochastic optimizer for `local_update` and `local_regularized_update` in Algorithm 1. The learning rate, weight decay, and batch size are set to recommended values from literature. In the supplementary materials, we provide recommended practices from literature to calculate the gradient of a regularized loss, such as the one in `local_regularized_update`.

4.2.2. Results

The results from the simulations above are summarized in Tables 2 and 3. Table 2 shows the average of classification accuracies among clients and their corresponding standard deviations from three independent random experiments. Many interesting insights can be derived from the results. First, `PFL-DA` outperforms all benchmarks. This highlights `PFL-DA`'s ability to tackle heterogeneity in both the input space and input–output relationship. Second, it is worthwhile to zoom in on the

FEMNIST dataset. Unlike the other datasets, the domain shift in this dataset is small as most writers have data across different classes. Here, the improvement of PFL-DA over Ditto is rather marginal. This is intuitively expected as both methods can handle a concept shift in the input–output relationship. It is also not surprising that Simple-DA has a very bad performance on FEMNIST as it cannot inherently handle a concept shift. Third, we can observe that a domain shift has a strong negative effect on Ditto. Indeed, in the case where the domain shift is tangible (PACS), Ditto performs much worse than individualized training. This sheds light on the counterexample provided in Section 3.2 that shows that with excessive domain heterogeneity, current personalized approaches can hurt the personalization process by staying within the vicinity of an uninformative global model. Here we note that the TP approach did not suffer as much in PACS since it does not restrict the personalized parameters to stay close to the global model, yet it still under-performs compared to indiv as staring from a bad global model can have a negative effect on the final solution.

Table 3 shows the standard deviation of accuracies across clients. This in turn measures fairness in FL which is usually defined through accuracy disparity across clients, where the goal is to incur a uniformly good performance across all devices (Kairouz et al. 2019; Li et al. 2019). To better visualize the distribution of predictive performance of personalized models on different clients, we use a boxplot to see testing accuracy of FEMNIST in Figure 3. The results show that PFL-DA promotes fairness compared to benchmarked methods and can even improve upon the fairness of Ditto.

4.2.3. Sensitivity Analysis on Heterogeneity

To evaluate the effect of domain shifts on model performance, we change the heterogeneity in Rotated MNIST and Colored MNIST. Remember that in Rotated MNIST, we rotate all images on one client by a certain angle in $[0, \alpha_{\max}]$, where $\alpha_{\max}$ can represent the data heterogeneity. We change $\alpha_{\max}$ from 20 to 80 degrees and examine the performance of three representative algorithms: PFL-DA, FedAverage, and Ditto. Similarly, in Colored MNIST, we change the maximum correlation parameter $\rho_{\max}$ from 0.2 to 0.8, and run the three algorithms. Results are shown in Figure 4. From Figure 4, it is clear that Ditto and PFL-DA perform comparably on datasets with small domain shifts. However, when domain shifts grow larger, PFL-DA significantly outperforms other algorithms.

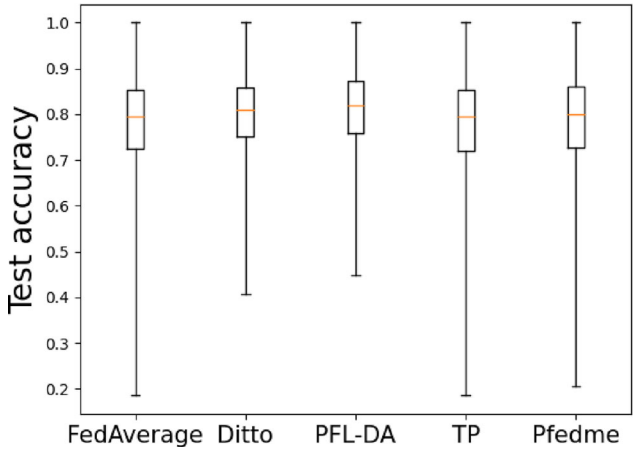


Figure 3. Boxplot of testing accuracy at the 100th communication round on FEMNIST.

4.3. Distributed and Connected Material Extrusion 3D Printers

The experiments done above are based on artificial datasets. Unfortunately, as FL is still in its infancy phase, real-life datasets (in engineering, etc.) are largely missing. The few that exist are largely based on mobile applications. To this end, there are some recent efforts to build engineering-based datasets to test FL and understand its domain-specific challenges (Kontar et al. 2021). One dataset is based on connected 3D printers (IoFT datasets 2021). In this dataset, multiple desktop 3D printers are installed in geographically separated locations. The goal of this test is to allow different clients to share knowledge and build better models for 3D printing while preserving their privacy and intellectual merit and reducing latency in model deployment.

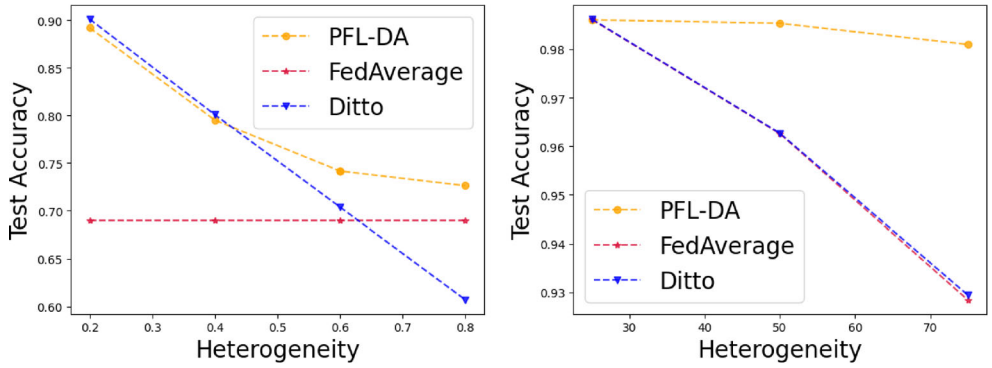
A typical 3D printer constructs 3D models by successive addition of materials. 3D printing has found successful applications in a wide range of fields, including the aerospace industry, automotive industry, food industry, and much more (Ngo et al. 2018; Shahrubudin, Lee, and Ramlan 2019). Different types of 3D printers are proposed to satisfy the diversified needs of users, such as binder jetting printers and materials extrusion printers.

Many popular desktop 3D printers use *material extrusion* technology. These printers use motors to control the movement of the printhead and printbed and usually stack thermoplastics layer by layer. Such printers are often inexpensive and easy to install. However, they can suffer from vibration-related issues. The vibration of the print head directly affects the precision of printed models: high vibration can lead to model distortions or even scrapped parts (Duan, Yoon, and Okwudire 2018). Reducing printing speed helps to mitigate vibration issues. However, lower printing speed can elongate printing time.

Our case study uses personalized FL methods to learn the dependence of printing speed and vibration in printing cubic objects. The findings can provide insights for balancing printing quality with printing speed for desktop 3D printers. Indeed, this balance is an active area of research (Duan, Yoon, and Okwudire 2018).

Data (Chou and Okwudire 2021) is collected from six Ender 3D printers located in different places. Notice that 3D printers are delivered by parts and assembled by the user. Thus, the environment, assembling, and adjusting process all introduce noise to printers. Each printer is also equipped with two accelerometers to measure the accelerations in the x and y axis. A central server communicates with every printer. Thus, one printer naturally represents one client, which is ideal for FL experiments. Figure 5(a) is an illustration of the federated system.

The dataset is generated by printing the same 15-layer cube on the six printers. Every printer prints the cube 12 times with 12 different printing speeds. For each experiment, accelerations



(a) Colored MNIST. (b) Rotated MNIST.

Figure 4. Change of performance with respect to heterogeneity.

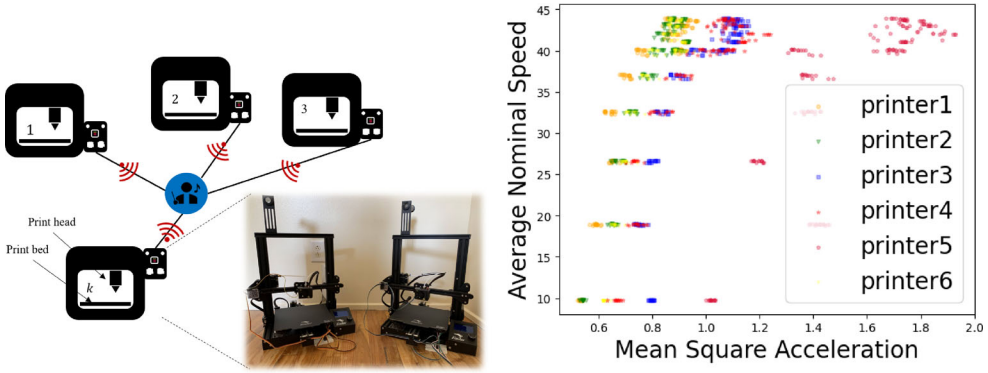


Figure 5. Setup and raw data of desktop 3D printers.

on x and y axis are collected at sampling rate 227Hz. To evaluate the overall vibrations, a program estimates the mean square of acceleration signals on both axis for each layer: σ_{a_x} and σ_{a_y}, and calculates the square root of the sum $\sigma_a = \sqrt{\sigma_{a_x}^2 + \sigma_{a_y}^2}$. Then the program calculates speed s by dividing the print head trajectory length by the total time elapsed. The raw data is plotted in Figure 5(b).

Our task is to learn customized models to predict the printing speed at a given acceleration. The predictions are useful when one user tries to set the printing speed at a given acceleration (thus, model precision) constraint. Intuitively, since all printers have the same type, the relations among different clients should follow a similar trend. However, due to differences in shipping, assembling, tuning, different clients have idiosyncratic characteristics as shown in Figure 5(b). This individuality renders personalized FL approaches suitable for model learning.

Now for model learning, a direct insight from the Figure 5(b) is that the (σ_a, s) curves on different printers have clear shifts on the x-axis. For this reason, we set the encoder functions $\Phi_{\boldsymbol{\beta}_i}$ to be linear transformations. Further, the (σ_a, s) curves are clearly nonlinear. Therefore, we use nonlinear functions for the decoder $g_{\boldsymbol{\gamma}_i}$ including sigmoid functions, Gaussian basis functions, and nonlinear neural networks. As such, the different $\boldsymbol{\gamma}_i$'s will mainly decide the different curve shapes and $\Phi_{\boldsymbol{\beta}_i}$ will map the phased input into a domain invariant space.

We study three algorithms, `Ditto`, `indiv`, and `PFL-DA`. For each algorithm, we use three models: a sigmoid decoder model, a Gaussian decoder model, and a neural network model. The encoder and decoder functions are specified in Table 4.

Table 4. Model specifications for generalized linear model with sigmoid decoder, generalized linear model with Gaussian decoder, and neural networks.

Model	$\Phi_i(x)$	$g_i(x)$
Sigmoid decoder model	$\phi_{1,i}x + \phi_{2,i}$	$g_{1,i}\frac{1}{1+e^{-x}} + g_{2,i}$
Gaussian decoder model	$\phi_{1,i}x + \phi_{2,i}$	$\sum_{k=1}^{K} w_k e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$
Neural network	Neural network	Neural network

To evaluate the performance of all methods, we split each printer's dataset into a training and testing set. The testing set consists of all data collected in the experiment where the printing speed is set to 30. The training set consists of the remaining data. We fit the model with training data, then use testing set to estimate testing loss between model prediction and actual measurements. The metrics for both training and testing loss are the mean squared error. Since the 3D printers usually work smoothly at printing speed 30, the testing loss roughly represents the prediction error in the moderate vibration region. We set $\lambda_1 = 0.9$, to allow greater flexibility across the models. We repeat this experiment three times, starting from different random initialization.

The testing loss is presented in Table 5. The results show that `PFL-DA` performs consistently well across all decoder functions and achieves the lowest testing loss. Also, we find that `Indiv`

Table 5. Test loss with standard deviations.

Model	Ditto	indiv	PFL-DA
Sigmoid GLM	0.28 ± 0.02	0.268 ± 0.01	**0.23** ± 0.01
Gaussian GLM	**0.26** ± 0.04	**0.25** ± 0.01	**0.25** ± 0.01
Neural Network	0.48 ± 0.04	**0.25** ± 0.02	**0.23** ± 0.02

NOTE: Bold values indicates the highest accuracy results that are statistically significantly better than the rest.

outperforms Ditto. This is not surprising as Ditto can easily fail in the presence of large domain shifts.

In the supplementary materials we provide an additional illustrative figure on the predictions of the three benchmarks across all decoder functions.

5. Conclusion

This article proposes PFL-DA: a personalized FL approach to tackle heterogeneity in both a concept shift in the output–input relationship and a covariate shift in the input space across clients. Our approach learns a personalized featurizer and decoder over all clients yet facilitates information sharing by restricting the personalized decoder to the vicinity of a global decoder. Through theoretical analysis, simple mathematical examples and case studies, we show that PFL-DA excels in the presence of both a covariate and concept shift while yielding comparable, often better, performance when only a concept shift exists. Further, our approach is shown to circumvent fundamental shortcoming with state of the art approaches in FL and is able promote fairness across clients.

Our approach assumes independent samples from independent clients. However, modeling correlations is an interesting yet challenging topic to investigate. Correlations are helpful in network learning or clustering over clients, yet it poses new challenges in optimization, communication, and privacy. This is an exciting direction that we hope to investigate.

Supplementary Materials

The code for numerical experiments in this article are available in the GitHub repository. In the supplementary material, we will show the proof of Theorem 3.1. We will also present additional experiments and illustrations, mathematical derivations of the proof of concept experiment, and details of our case study implementations.

References

Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., and Saligrama, V. (2019), "Federated Learning Based on Dynamic Regularization," in *International Conference on Learning Representation*. [1,2]

Albuquerque, I., Monteiro, J., Falk, T. H., and Mitliagkas, I. (2019), "Adversarial Target-Invariant Representation Learning for Domain Generalization." in ArXiv. [8]

Arivazhagan, M. G., Aggarwal, V., Singh, A. K., and Choudhary, S. (2019), "Federated Learning with Personalization Layers," arXiv preprint arXiv:1912.00818. [2,3]

Asadi, N., Hosseinzadeh, M., and Eftekhari, M. (2019), "Towards Shape Biased Unsupervised Representation Learning for Domain Generalization," in ArXiv. [8]

Atzori, L., Iera, A., and Morabito, G. (2010), "The Internet of Things: A Survey," *Computer Networks*, 54, 2787–2805. [1]

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konecny, J., McMahan, H. B., Smith, V., and Talwalkar, A. (2019), "LEAF: A Benchmark for Federated Settings," in *NeurIPS*. [8]

Chou, C.-H., and Okwudire, C. (2021), "Federated Learning Dataset: A Case Study of Vibration Analysis for Desktop 3D Printers," available at *https://doi.org/10.5281/zenodo.5747732*. [9]

Dinh, C. T., Tran, N. H., and Nguyen, T. D. (2020), "Personalized Federated Learning with Moreau Envelopes," in *34th Conference on Neural Information Processing Systems*. [2,3,7]

Du, W., Xu, D., Wu, X., and Tong, H. (2021), "Fairness-Aware Agnostic Federated Learning," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, SIAM, pp. 181–189. [1,3]

Duan, M., Yoon, D., and Okwudire, C. E. (2018), "A Limited-Preview Filtered B-Spline Approach to Tracking Control–With Application to Vibration-Induced Error Compensation of a 3D Printer," *Mechatronics*, 56, 287–296. [9]

Fang, X., Paynabar, K., and Gebraeel, N. (2017), "Multistream Sensor Fusion-based Prognostics Model for Systems with Single Failure Modes," *Reliability Engineering & System Safety*, 159, 322–331. [2]

Ganin, Y., and Lempitsky, V. (2015), "Unsupervised Domain Adaptation by Backpropagation," *Journal of Machine Learning Research*, 37, 117–139. [3]

Ghadimi, S., and Lan, G. (2013), "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming," arXiv preprint arXiv:1309.5549. [5]

Gulrajani, I., and Lopez-Paz, D. (2020), "In Search of Lost Domain Generalization," arXiv preprint arXiv:2007.01434. [3]

Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. (2018), "Federated Learning for Mobile Keyboard Prediction," arXiv preprint arXiv:1811.03604. [2]

IoFT datasets (2021), "The Internet of Federated Things (IoFT) Data Directory," available at *https://ioft-data.engin.umich.edu/*, accessed: 2021-07-18. [7,9]

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2019), "Advances and Open Problems in Federated Learning," arXiv preprint arXiv:1912.04977. [9]

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. (2020), "SCAFFOLD: Stochastic Controlled Averaging for Federated Learning," in *International Conference on Machine Learning*, PMLR, pp. 5132–5143. [1,2]

Kingma, D. P., and Ba, J. (2015), "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*. [8]

Kontar, R., Raskutti, G., and Zhou, S. (2020), "Minimizing Negative Transfer of Knowledge in Multivariate Gaussian Processes: A Scalable and Regularized Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43, 3508–3522. [3]

Kontar, R., Shi, N., Yue, X., Chung, S., Byon, E., Chowdhury, M., Jin, J., Kontar, W., Masoud, N., Noueihed, M., Okwudire, C. E., Raskutti, G., Saigal, R., Singh, K., and Ye, Z.-S. (2021), "The Internet of Federated Things (IoFT)," *IEEE Access*, 9, 156071–156113. [1,2,3,9]

Kontar, R., Zhou, S., Sankavaram, C., Du, X., and Zhang, Y. (2017), "Nonparametric-Condition-based Remaining Useful Life Prediction Incorporating External Factors," *IEEE Transactions on Reliability*, 67, 41–52. [2]

Li, T., Beirami, A., Sanjabi, M., and Smith, V. (2020a), "Tilted Empirical Risk Minimization," arXiv preprint arXiv:2007.01162. [3]

Li, T., Hu, S., Beirami, A., and Smith, V. (2021), "Ditto: Fair and Robust Federated Learning Through Personalization," arXiv preprint arXiv:2012.04221. [2,3,5,7]

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020b), "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, 37, 50–60. [1]

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., and Ameet Talwalkar, V. S. (2018), "Federated Optimization in Heterogeneous Networks," in *Proceedings of the 3rd MLSys Conference*. [2]

Li, T., Sanjabi, M., Beirami, A., and Smith, V. (2019), "Fair Resource Allocation in Federated Learning," in *International Conference on Learning Representations*. [3,9]

Liang, P. P., Liu, T., Ziyin, L., Salakhutdinov, R., and Morency, L.-P. (2020), "Think Locally, Act Globally: Federated Learning with Local and Global Representations," arXiv preprint arXiv:2001.01523. [3,7]

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. (2020), "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, 22, 2031–2063. [2]

Mangasarian, O. L., and Solodov, M. V. (1994), "Backpropagation Convergence via Deterministic Nonmonotone Perturbed Minimization," *Advances in Neural Information Processing Systems*, pp. 383–383. [1]

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. (2017), "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*. [1,2,5,7]

Ngo, T. D., Kashani, A., Imbalzano, G., Nguyen, K. T., and Hui, D. (2018), "Additive Manufacturing (3D Printing): A Review of Materials, Methods, Applications and Challenges," *Composites Part B: Engineering*, 143, 172–196. [9]

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. (2021), "Adaptive Federated Optimization," in *International Conference on Learning Representations*. [1,2]

Richardson, M., and Wallace, S. (2012), *Getting Started with Raspberry Pi*, Sebastopol, CA: O'Reilly Media, Inc. [1]

Shahrubudin, N., Lee, T., and Ramlan, R. (2019), "An Overview on 3D Printing Technology: Technological, Materials, and Applications," in *Procedia Manufacturing* (Vol. 35), pp. 1286–1296, the 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8–10 March 2019, Sun City, South Africa. [9]

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. (2017), "Federated Multi-Task Learning," in *NeurIPS*. [2,3]

Sun, B., and Saenko, K. (2016), "Deep CORAL: Correlation Alignment for Deep Domain Adaptation," in *ECCV*. [3]

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2017), "Simultaneous Deep Transfer Across Domains and Tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2107–2116. [3]

Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., and Ramage, D. (2019a), "Federated Evaluation of On-Device Personalization," arXiv preprint arXiv:1910.10252. [2,3]

Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., and Chen, M. (2019b), "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, 33, 156–165. [2]

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. (2018), "Applied Federated Learning: Improving Google Keyboard Query Suggestions," arXiv preprint arXiv:1812.02903. [2]

Yu, T., Bagdasaryan, E., and Shmatikov, V. (2020), "Salvaging Federated Learning by Local Adaptation," arXiv preprint arXiv:2002.04758. [3,6,7]

Yue, X., Nouiehed, M., and Kontar, R. A. (2021), "GIFAIR-FL: An Approach for Group and Individual Fairness in Federated Learning," arXiv preprint arXiv:2108.02741. [1,3,5]