Debiasing Model Updates for Improving Personalized Federated Training

Durmus Alp Emre Acar [1] **Yue Zhao** [2] **Ruizhao Zhu** [1]
Ramon Matas Navarro [2] **Matthew Mattina** [2] **Paul N. Whatmough** [2]
Venkatesh Saligrama [1]

Abstract

We propose a novel method for federated learning that is customized specifically to the objective of a given edge device. In our proposed method, a server trains a global meta-model by collaborating with devices without actually sharing data. The trained global meta-model is then personalized locally by each device to meet its specific objective. Different from the conventional federated learning setting, training customized models for each device is hindered by both the inherent data biases of the various devices, as well as the requirements imposed by the federated architecture. We propose gradient correction methods leveraging prior works, and explicitly de-bias the meta-model in the distributed heterogeneous data setting to learn personalized device models. We present convergence guarantees of our method for strongly convex, convex and nonconvex meta objectives. We empirically evaluate the performance of our method on benchmark datasets and demonstrate significant communication savings.

1. Introduction

Federated learning (FL) introduced by McMahan et al. (2017), proposes to leverage local data stored across massively distributed edge devices to train a prediction model that matches the performance of a learner with centralized data. An FL system is composed of a server that coordinates interactions with edge devices, and refines its model over several rounds of communication. FL is challenging because it imposes at least two fundamental constraints during training. Firstly, the local data itself cannot be shared or transmitted due to privacy concerns. Secondly, the number of rounds of server-device interaction, as well as the number of devices participating in any round, must be small on account of limited connectivity and available communication bandwidth.

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

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

In this paper, we propose a novel federated training approach based on meta-learning, which allows for sample-efficient customization of a centralized model to suit an end user's objectives. That meta-learning approaches are well-suited for our customization scenario is not surprising, and has been leveraged in prior works (Chen et al., 2018; Jiang et al., 2019; Fallah et al., 2020). Indeed, in our setup, each user is associated with a different task, and our goal, as in meta-learning, is to train a meta-model on a variety of tasks, such that this model can be rapidly re-purposed to solve new or existing tasks using only a few training examples.

Challenges. Personalized Federated Learning (PFL)

[1]Boston University, Boston, MA [2]Arm ML Research Lab, Boston, MA. Correspondence to: Durmus Alp Emre Acar <alpacar@bu.edu>, Venkatesh Saligrama <srv@bu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

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

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

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

Contributions.

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

1.1. Related Work

Federated learning. Federated learning (McMahan et al., 2017) is a distributed optimization problem where a server iteratively trains a global model by collaborating with many devices, without centralizing device data. Federated learning is a fast moving field; here we focus on closely related works and refer to Kairouz et al. (2019) for a more detailed discussion.

Federated learning aims to decrease the number of model transmissions between the server and the devices (McMahan et al., 2017; Zhang et al., 2013). The motivation comes from the resource constraint nature of the devices. In IoT devices, transmission costs due to communication dominates the energy consumption (Acar et al., 2020; Wang et al., 2019; Shamir et al., 2014; Zhu et al., 2019).

The FedAvg (McMahan et al., 2017) algorithm is an extension of local SGD (Zinkevich et al., 2010), where devices apply a predefined number of SGD updates on the global model, and the server averages these device models. FedAvg provides communication savings if the device datasets are close to each other. When device data are not identically distributed, the performance of FedAvg significantly degrades (Zhao et al., 2018). This degradation is due to the inconsistency between device level optimization and global optimization. A substantial amount of work has been proposed to address this inconsistency. One such line of research focuses on inexact minimization of device-level problems (Li et al., 2020a;b). For example, FedProx (Li et al., 2020a) forces device models to be close to the current server model using an explicit regularization. Another direction is to change server updates (Reddi et al., 2021; Hsu et al., 2019). For instance, FedAdam (Reddi et al., 2021) uses ADAM optimization on the server side, instead of averaging device models. Yet another line of research avoids this inconsistency by explicitly transferring corrections along with the server model (Shamir et al., 2014; Karimireddy et al., 2019). For example, SCAFFOLD (Karimireddy et al., 2019) defines a gradient state for all devices and transmits the gradient states as well as the models in each round. FedDyn (Acar et al., 2021a) proposes dynamic regularization to align local and global objectives. Our proposed approach builds on these works and specializes it to the personalization setting. In contrast to federated learning schemes where the server aims to find a global model that performs well on data from every device, personalized federated learning seeks to find a meta model on the server which each device can customize based on the locally available dataset.

Meta learning. Meta learning is defined as 'learning to learn' (Thrun & Pratt, 2012). In this concept, the aim of the meta learner is to learn how to learn new tasks. This paradigm is motivated from human learning where humans are able to transfer their experience from previous task instances into new tasks. Thus, meta learning is thought of as an important step towards future machine learning research (Lake et al., 2017). We refer to extensive surveys Vanschoren (2018); Hospedales et al. (2020) for a detailed discussion and we discuss closely related work here.

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

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

Personalized federated learning is related to online meta learning (Finn et al., 2019; Zhuang et al., 2020; Acar et al., 2021b). In online meta learning, the learner predicts on a new task, but has the benefit of past experiences, which allows for adaptation to the new task. The similarity stems from considering the large device limit, and when data/device is small. In this situation, the latency in terms of revisits to a specific device increases, and we are in the situation of online meta-learning. Extending personalized federated learning to online personalized federated learning is a promising future direction.

2. Method

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

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

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

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

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

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

We propose to use another meta transformation named as prototypical adaptation (Snell et al., 2017). Prototypical

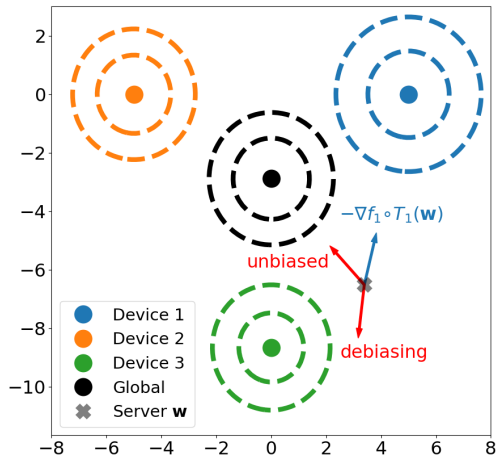

Figure 1. Toy example with three devices in a two dimensional parameter space. Only device 1 is active in the current round. In Fallah et al. (2020) algorithm, the model is pulled towards device 1's local minima due to the bias discrepancy. The debiasing and the correct unbiased directions are needed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

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

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

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

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

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

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

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

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

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

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

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

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \langle \nabla f_i \circ T_i(\boldsymbol{w}^*), \boldsymbol{w} \rangle + \frac{\alpha}{2} \|\boldsymbol{w} - \boldsymbol{w}^*\|^2 \quad (1)$$

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

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

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

We note that if the server model converges to a stationary point of OPT, then we recover the modified objective as in 1.

Algorithm 1 Personalized Federated Learning, PFL

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

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

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

where the gradient of the global loss, OPT, appears as a linear term. In Figure 1, the direction of these two terms are shown in red arrows where the first linear term debiases the current loss and the second linear term results in the correct gradient direction.

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

An Alternative Surrogate. Instead of using $\boldsymbol{w}^t$ model in the linear terms, we can as well use the recent device models, $\boldsymbol{w}_i^t$, to construct them. Then, the objective in Eq. 2 becomes,

$$\min_{\boldsymbol{w}} f_i \circ T_i(\boldsymbol{w}) - \left\langle \nabla f_i \circ T_i(\boldsymbol{w}_i^t), \boldsymbol{w} \right\rangle + \left\langle \frac{1}{m} \sum_{j\in[m]} \nabla f_j \circ T_j(\boldsymbol{w}_j^t), \boldsymbol{w} \right\rangle + \frac{\alpha}{2} \left\| \boldsymbol{w} - \boldsymbol{w}^t \right\|^2. \quad (3)$$

Algorithm 2 PFL Subroutines

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

We note that if the device models, $\boldsymbol{w}_i^t$s, converge to the stationary point of OPT, we still recover the proposed the modification as in Eq. 1. This seemingly subtle modification allows the server to transmit only one model instead of two models. This extension is further discussed in the subsequent section.

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

Learning Structure. The general structure of PFL is given in Algorithm 1. In the beginning of each communication round, P devices are selected uniformly at random as active device set $\mathcal{P}_t$. The current server meta model, $\boldsymbol{w}^t$, is sent to each of these active devices. Devices construct the regularizer depending on the objectives as in Eq. 2 or 3 where $\boldsymbol{g}_i^t$ corresponds to the gradient of device level meta loss.

We continue device level optimization with a subroutine consisting of SGD steps and the constructed regularizer as described in Update method in Algorithm 2. First, we

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

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

where $\tilde{f}_i(\overline{\boldsymbol{w}}_{i,k}^{t+1}, D_i^{k'})$ is the empirical loss with the customized model $\overline{\boldsymbol{w}}^{t+1}$ on minibatch $D_i^{k'}$ and β is the learning rate. After performing K SGD updates, the subroutine ends and we set the device meta model as $\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_{i,K+1}^{t+1}$.

By definition, the regularizer, $\mathcal{R}_i^t(\boldsymbol{w})$, depends on the current server model and the current gradient information, $\boldsymbol{g}_i^t$. We update the gradient information for the next round depending on the objective as in Eq. 2 or 3.

After solving the device level optimization, active devices transmit the trained model $\boldsymbol{w}_i^{t+1}$ back to the server for aggregation. On the other hand, inactive devices do not receive the current model and freeze their local gradients and local models to be their past values. We note that we communicate the local gradient information along with the trained model if we use the objective in Eq. 2.

PFLDyn Algorithm. PFLDyn algorithm is based on the local objective as in Eq. 3 which is an extension of FedDyn to personalized federated learning.

The server integrates the global gradient information to the server model as explained in Update$_1$. Since the server meta model already has the global gradient information, devices do not need extra tranmsmission of the global gradient and they construct the regularizer as shown in Regularizer method in Algorithm 2. In PFLDyn, devices communicate only device models.

PFLScaf Algorithm. Using local objective as in Eq. 2, we can get to PFLScaf algorithm which is an extension of SCAFFOLD to personalized federated learning.

The server obtains the device models and device gradients from the active devices. It then calculates the global gradient and the server meta model as shown in Update$_2$. The server meta model and the global gradient are transmitted to the devices. Devices correct the biased gradient of the local loss with global gradient information as described in Regularizer method in Algorithm 2. Different from PFLDyn, PFLScaf communicates the models as well as gradient information.

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

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

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

It is important to highlight the fact that the gradient of the individual device level meta objectives are not necessarily $\mathbf{0}$ .i.e ($\nabla f_i(\overline{\boldsymbol{w}}_i^*) \neq \mathbf{0}$). Indeed, were this to be the case, it would imply that fully optimizing device meta models would lead to device specific biases in the meta-model. PFL proposes to sequentially modify device empirical risk functions to eliminate such data-specific bias. The fact that this is possible is justified in the following Proposition 1.

Proposition 1. *For sufficiently large K, if the device meta models in Algorithm PFLDyn converge, they converge to the optimal meta model as,*

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

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

2.1. Analysis of PFL

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

Centralized Competitor. The centralized meta model $\boldsymbol{w}^*$ minimizes OPT with access to all device datasets. Since Algorithm 1 does not share data among devices, we characterize its performance with the number of communication

rounds required to achieve ϵ difference relative to performance of centralized learner minimizing OPT, namely,

$$E\left[F(\boldsymbol{w}_{\text{Alg}}^T)\right] - F(\boldsymbol{w}^*) \leq \epsilon, \tag{4}$$

where $\boldsymbol{w}_{\text{Alg}}^T$ is the meta model Algorithm 1 finds after T communication rounds, $\boldsymbol{w}^*$ is the optimal meta model and expectation is with respect to randomness due to active device set at each round ($\mathcal{P}_t$) . While the statement 4 is tractable for convex $\{f_i \circ T_i\}$ functions, for nonconvex functions, following convention, we state the convergence as the number of communication rounds required to achieve a stationary point of OPT with ϵ error as,

$$E\left\|\nabla F\left(\boldsymbol{w}_{\text{Alg}}^T\right)\right\|^2 \leq \epsilon. \tag{5}$$

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

Theorem 1. *For a suitably chosen $\alpha \in \mathbb{R}$, PFLDyn Algorithm, for sufficiently large K, returns an expected error less than ϵ in T communication rounds as,*

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

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

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

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

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

The expected error is defined as in Eq. 4 and in Eq. 5 for convex and nonvonex functions respectively. The expectation is over the randomness of device participation.

Theorem 1 shows that with sufficient number of iterations, Algorithm 1 reaches to an expected ϵ error for convex and nonconvex device level meta function as in relations Eq. 4 and 5 respectively. We see the number of communication rounds to achieve expected ϵ error scales with $\frac{1}{\epsilon}$ for both convex and nonconvex settings. We present results for strongly convex functions in the supplementary section.

3. Experiments

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

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

We first start with a summary datasets and models used in this section and we refer to Appendix A.1 for details of the empirical setup. We then continue how we construct diverse device datasets that reflects heterogeneity among devices. Then, we explain the metrics and measures we use to compare the methods. Finally, we present our findings.

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

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

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

Table 1. The number of model transmissions relative to one round of Fallah et al. (2020) required to reach the target test accuracy for the average level personalization performance in the Active Class Induced Diversity (ACID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of Fallah et al. (2020). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and the gain with respect to Fallah et al. (2020) method is shown.

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

experiment with increasing and decreasing the number of classes per device, and these experiments point to how PFL handles different levels of task diversity.

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

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

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

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

Oracle Accuracy. In Section 2.1 we derived convergence guarantees with respect to an oracle that has centralized data access and optimizes Eq. OPT accordingly. In this measure, we report the number of communication rounds required to get close to the target accuracy of an oracle.

Partial Participation. Following federated learning settings, we tested the methods with 100 devices where 10% of them are active at each round. For CIFAR-10 and CIFAR-100, we considered both ACID and ALID with $3, 5$ and 7 classes per device schemes. We refer to Appendix A.1 for additional experimental details.

Analysis and Discussions. Table 1 shows the gain compared to Fallah et al. (2020) method for various settings of $3, 5$ and 7 classes per devices for both datasets in the ACID setting. Similarly, Table 2 demonstrates the device performances for the ALID scenario. The convergence curves as well as the lowest and the highest level personalization results are given Appendix A.1.

Personalization is needed in both ACID and ALID scenarios. As seen in Figure 4 and 5, no personalization baselines converge to substantially lower average test accuracy in the ACID scenario. Furthermore, these methods predict the classes randomly in the ALID scenario due to the label anonymity, which in turn indicates the need of personalization. Thus, the results of PFL and FedAvg without personalization are not tabulated in tables due to their non-comparable performance.

PFLDyn (Proto) leads to significant savings in both ACID and ALID scenarios. We observe that PFLDyn using Proto adaptation reaches the target accuracy faster than all the other methods on the average device performance metric in Table 1 and 2. For instance, in CIFAR-10, ACID 7 classes per device setting, PFLDyn (Proto) leads to more than $5\times$ communication savings to achieve the same level of performance compared to Fallah et al. (2020) method. This effect is also seen in Figure 2 where PFLDyn (Proto) converges faster and to a higher point than Fallah et al. (2020).

PFL based optimization outperforms FedAvg based optimization regardless of adaption function (MAML or Proto).

Table 2. The number of model transmissions relative to one round of Fallah et al. (2020) required to reach the target test accuracy for the average level personalization performance in the Anonymous Label Induced Diversity (ALID) scenario. Target accuracies are selected among the highest accuracy of our methods and the highest accuracy of Fallah et al. (2020). The methods without personalization are omitted due to their poor performance levels. The best method is highlighted and gain with respect to Fallah et al. (2020) method is shown.

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

As shown in Table 1 and 2, the PFL based methods converge to the target accuracy with fewer number of model transmissions in nearly all the experiments. As seen in Figure 2 and 3, PFL methods achieve a higher test accuracy compared to Fallah et al. (2020). We can infer that PFL is capable of debiasing meta-model updates at the server allowing for superior device personalization. Thus, PFL leads to a faster and robust convergence than FedAvg regardless of the adaption function.

PFL improves the performance of the lowest level personalization. In addition to the average test accuracy among all devices, PFL based methods converges faster than Fallah et al. (2020) even for the lowest level performing devices in both ACID and ALID scenarios as shown in Table 3 and 4. PFL improves the performance by finding a better meta model for personalization among all devices. In particularly, the savings of PFL on the average device performance doesn't sacrifice the performance of a subset of devices.

PFL achieves centralized performance. The centralized performance for CIFAR-10, 5 classes per device ACID and ALID using Proto adaptation is 89.8% and 90% respectively. Based on Table 1 and 2 we see that PFLDyn (Proto) achieves near the centralized performance around 300 communication rounds. Similarly, the centralized performance for CIFAR-100, 5 classes per device ACID setting using Proto adaptation is 89.7% . PFLDyn (Proto) achieves near centralized performance within 400 communication rounds without actually sharing device data.

Proto adaptation is robust to label anonymity. The Proto-based adaptation demonstrates similar convergence performance in the ALID scenario with anonymous labels compared to the ACID scenario. In contrast, the performance of the MAML-based adaption (PFLDyn, PFLScaf and (Fallah et al., 2020)) degrades significantly in the ALID scenario. According to Figure 2 and 3, the convergence curves of methods using the Proto adaption are similar in the ACID and ALID scenarios. However, the MAML-based methods converge to significantly lower average test accuracy. In addition, the same observation can be found in Table 1 and 2. For instance, in the CIFAR-10 5 classes setting, the Proto-based models (PFLDyn, PFLScaf and P-Avg) require 161, 452 and 341 communication rounds respectively to achieve the average test accuracy 87.5% in the ALID scenario, while similarly require 163, 436 and 286 rounds to achieve 87.6% in the ACID scenario. But Fallah et al. (2020) can no longer achieve such an accuracy level within 1000 rounds in the ALID scenario as in the ACID scenario. Thus, the Proto-based adaption is more robust than the MAML-based adaption in more strictly privacy-preserving scenario when the labels are anonymous among devices.

4. Conclusion

We propose a novel method PFL in federated learning with personalized user objectives. In this setting, a server trains a meta model in which a small adaptation using available data leads to a high performance for the corresponding device. Prior works propose solutions using FedAvg method with the personalized objective. We point out that non identical distributions among devices cause different optimal meta models for devices and this leads to suboptimal results if FedAvg is used. Different from the recent work, PFL debiases the meta-model, so that each device can be effectively customized towards its objective. We analyse PFL both theoretically and empirically show that it leads to significant communication savings compared to the competitors.

Acknowledgements

This research was supported by a gift from the ARM corporation, and by Army Research Office Grant W911NF2110246, the National Science Foundation grants CCF-2007350 (VS), CCF-2022446(VS), CCF-1955981 (VS), the Office of Naval Research Grant N0014-18-1-2257.

References

Acar, D. A. E., Gangrade, A., and Saligrama, V. Budget learning via bracketing. In *International Conference on Artificial Intelligence and Statistics*, pp. 4109–4119. PMLR, 2020.

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=B7v4QMR6Z9w.

Acar, D. A. E., Zhu, R., and Saligrama, V. Memory efficient online meta learning. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2021b.

Antoniou, A., Edwards, H., and Storkey, A. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Chen, F., Luo, M., Dong, Z., Li, Z., and He, X. Federated meta-learning with fast convergence and efficient communication. *arXiv preprint arXiv:1802.07876*, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1920–1930, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/finn19a.html.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.

Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K., and Chintala, S. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Hong, M., Luo, Z.-Q., and Razaviyayn, M. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey, 2020.

Hsu, T. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019. URL http://arxiv.org/abs/1909.06335.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

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

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019. URL http://arxiv.org/abs/1910.06378.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020*, pp. 429–450, 2020a.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HJxNAnVtDS.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Makhdoumi, A. and Ozdaglar, A. Convergence rate of distributed admm over networks. *IEEE Transactions on Automatic Control*, 62(10):5082–5095, 2017.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850, 2016.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pp. 1000–1008, 2014.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Vanschoren, J. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Wang, H., Saligrama, V., Sclaroff, S., and Ablavsky, V. Cost-aware fine-grained recognition for iots based on sequential fixations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Zhang, M., Sapra, K., Fidler, S., Yeung, S., and Alvarez, J. M. Personalized federated learning with first order model optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ehJqJQk9cw.

Zhang, Y., Duchi, J., Jordan, M. I., and Wainwright, M. J. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/d6ef5f7fa914c19931a55bb262ec879c-Paper.pdf.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhu, P., Acar, D. A. E., Feng, N., Jain, P., and Saligrama, V. Cost aware inference for iot devices. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2770–2779. PMLR, 2019.

Zhuang, Z., Wang, Y., Yu, K., and Lu, S. No-regret non-convex online meta-learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3942–3946. IEEE, 2020.

Zinkevich, M., Weimer, M., Smola, A. J., and Li, L. Parallelized stochastic gradient descent. In Lafferty, J. D., Williams, C. K. I., Shawe-Taylor, J., Zemel, R. S., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pp. 2595–2603. Curran Associates, Inc., 2010.