Federated Learning Meets Multi-Objective Optimization

Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu

***Abstract*—Federated learning has emerged as a promising, massively distributed way to train a joint deep model over large amounts of edgedevices while keeping private user data strictly on device. In this work, motivated from ensuring fairness among users and robustness against malicious adversaries, we formulate federated learning as multi-objective optimization and propose a new algorithm `FedMGDA+` that is guaranteed to converge to Pareto stationary solutions. `FedMGDA+` is simple to implement, has fewer hyperparameters to tune, and refrains from sacrificing the performance of any *participating* user. We establish the convergence properties of `FedMGDA+` and point out its connections to existing approaches. Extensive experiments on a variety of datasets confirm that `FedMGDA+` compares favorably against state-of-the-art.**

***Index Terms*—Pareto optimization, Distributed algorithms, Federated learning, Edge computing, Machine learning, Neural networks.**

I. Introduction

DEEP learning has achieved impressive successes on a number of domain applications, thanks largely to innovations on algorithmic and architectural design, and equally importantly to the tremendous amount of computational power one can harness through GPUs, computer clusters and dedicated software and hardware. Edge devices, such as smart phones, tablets, routers, car devices, home sensors, *etc.*, due to their ubiquity and moderate computational power, impose new opportunities and challenges for deep learning. On the one hand, edge devices have direct access to privacy sensitive data that users may be reluctant to share (with say data centers), and they are much more powerful than their predecessors, capable of conducting a significant amount of on-device computations. On the other hand, edge devices are largely heterogeneous in terms of capacity, power, data, availability, communication, memory, *etc.*, posing new challenges beyond conventional in-house training of machine learning models. Thus, a new paradigm, known as federated learning (`FL`) [1] that aims at harvesting the prospects of edge devices, has recently emerged. Developing new `FL` algorithms and systems on edge devices has since become a hot research topic in machine learning.

From the beginning of its birth, `FL` has close ties to conventional distributed optimization. However, `FL` emerged from the pressing need to address news challenges in the mobile era that existing distributed optimization algorithms were not designed for *per se*. We mention the following characteristics of `FL` that are most relevant to our work, and refer to the excellent surveys [2]–[4] and the references therein for more challenges and applications in `FL`.

- *Non-IID:* Each user's data can be distinctively different from every other user's, violating the standard iid assumption in statistical learning and posing significant difficulty in formulating the goal in precise mathematical terms [5]. The distribution of user data is often severely unbalanced.
- *Limited communication:* Communication between each user and a central server is constrained by network bandwidth, device status, user participation incentive, *etc.*, demanding a thoughtful balance between computation (on each user device) and communication.
- *Privacy:* Protecting user (data) privacy is of uttermost importance in `FL`. It is thus not possible to share user data (even to a cloud arbitrator), which adds another layer of difficulty in addressing the previous two challenges.
- *Fairness:* As argued forcibly in recent works (e.g., [5], [6]), ensuring fairness among users has become another serious goal in `FL`, as it largely determines users' willingness to participate and ensures some degree of robustness against malicious user manipulations.
- *Robustness:* `FL` algorithms are eventually deployed in the wild hence subject to malicious attacks. Indeed, adversarial attacks (e.g., [7]–[9]) have been constructed recently to reveal vulnerabilities of `FL` systems against malicious manipulations at the user side.

In this work, motivated from the last two challenges above, i.e., fairness and robustness, we propose a new algorithm `FedMGDA+` that complements and improves existing `FL` systems. `FedMGDA+` is based on multi-objective optimization and is guaranteed to converge to Pareto stationary solutions. `FedMGDA+` is simple to implement, has fewer hyperparameters

Manuscript received November 1, 2020; revised March 18, 2022; accepted April 13, 2022. Date of publication April 22, 2022; date of current version June 27, 2022. This work was done at Huawei Noah Ark's Lab, Montreal, QC, H3N 1X9, Canada. This work was supported in part by NSERC, in part by CIFAR AI Chairs Program, and in part by Waterloo-Huawei Joint Innovation Lab. Recommended for acceptance by Prof. Xiaowen Chu. *(Corresponding author: Zeou Hu.)*

Zeou Hu and Yaoliang Yu are with the Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: z97hu@uwaterloo.ca; yaoliang.yu@uwaterloo.ca).

Kiarash Shaloudegi is with Amazon Advertising, Amazon Canada, Toronto, ON M5H 4A9, Canada (e-mail: kiarash.shaloudegi@huawei.com).

Guojun Zhang is with Huawei Noah Ark's Lab, Montreal, QC H3N 1X9, Canada (e-mail: g39zhang@uwaterloo.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNSE.2022.3169117, provided by the authors.

Digital Object Identifier 10.1109/TNSE.2022.3169117

to tune, and most importantly *refrains from sacrificing the performance of any participating user*. We demonstrate the superior performance of FedMGDA+ under a variety of metrics including accuracy, fairness, and robustness.

We summarize our contributions as follows:

- In Section III, based on the proximal average we provide a novel, unifying and revealing interpretation of existing FL practices.
- In Section IV, we summarize some background on multi-objective optimization and point out its connections to existing FL algorithms. We believe this new perspective will yield more fruitful exchanges between the two fields in the future.
- In Section V, we propose FedMGDA+ that complements existing FL systems while taking robustness and fairness explicitly into its algorithmic design. We prove that FedMGDA+ converges to a Pareto stationary solution under mild assumptions.
- In Section VI, we perform extensive experiments to validate the competitiveness of FedMGDA+ under a variety of desirable metrics, and to illustrate the respective pros and cons of our and alternative algorithms.

We discuss more related works in Section II and we conclude in Section VII with some future directions.

To facilitate reproducibility, we have released our code at: https://github.com/watml/Fed-MGDA.

II. Related Works

In this section we give a brief review of some recent works that are directly related to ours and put our contributions in context. To start with, [1] proposed the first FL algorithm known as "Federated Averaging" (a.k.a., FedAvg), which is a synchronous update scheme that proceeds in several rounds. At each round, the central server sends the current global model to a subset of users, each of which then uses its respective local data to update the received model. Upon receiving the updated local models from users, the server performs aggregation, such as simple averaging, to update the global model. For more discussion on different averaging schemes, see [10]. [11] extended FedAvg to better deal with non-i.i.d. distribution of data, by adding a "proximal regularizer" to the local loss functions and minimizing the Moreau envelope function for each user. The resulting algorithm FedProx, as pointed out in Section III, is a randomized version of the proximal average algorithm in [12] and reduces to FedAvg when regularization diminishes.

Analysing FedAvg has been a challenging task due to its flexible updating scheme, partial user participation, and non-iid distribution of client data [11]. The first theoretical analysis of FedAvg for strongly convex and smooth problems with iid and non-iid data appeared in [13] and [10], respectively, where the effect of different sampling and averaging schemes on the convergence rate of FedAvg was also investigated, leading to the conclusion that such effect becomes particularly important when the dataset is unbalanced and non-iid distributed. In [14], FedAvg was analyzed for non-convex problems, where FedAvg was formulated as a stochastic gradient-based algorithm with biased gradients, and the convergence of FedAvg with decaying step sizes to stationary points was proved. Moreover, [14] proposed FedMom, a server-side acceleration based on Nesterov's momentum, and proved again its convergence to stationary points. Lately, [15] proposed and analyzed federated versions of several popular adaptive optimizers (e.g. ADAM). They generalize the framework of FedAvg by decoupling the FL update scheme into *server optimizer* and *client optimizer*. Interestingly, same as us, [15] also observed the importance of learning rate decays on both clients and server.

Recently, an interesting work by [16] demonstrated theoretically that fixed points reached by FedAvg and FedProx (if exist) need not be stationary points of the original optimization problem, even in convex settings and with deterministic updates. To address this issue, they proposed FedSplit to restore the correct fixed points. It still remains open, though, if FedSplit can still converge to the correct fixed points under asynchronous and stochastic user updates, both of which are widely adopted in practice and studied here.

Ensuring fairness among users has become a serious goal in FL since it largely determines users' willingness to participate in the training process. [5] argued that existing FL algorithms can lead to federated models that are biased toward different users. To solve this issue, [5] proposed agnostic federated learning (AFL) to improve fairness among users. AFL considers the target distribution as a weighted combination of the user distributions and optimizes the centralized model for the worse-case realization, leading to a saddle-point optimization problem which was solved by a fast stochastic optimization algorithm. On the other hand, based on fair resource allocation in wireless networks, [6] proposed q-fair federated learning (q-FFL) to achieve more uniform test accuracy across users. [6] further proposed q-FedAvg as a communication efficient algorithm to solve q-FFL. However, both AFL and q-FedAvg do not explicitly encourage user participation and they suffer from adversarial attacks while our algorithm FedMGDA+ is designed to be fair among participants and robust against both additive and multiplicative attacks.

FedAvg relies on a coordinate-wise averaging of local models to update the global model. According to [17], in neural network (NN) based models, such coordinate-wise averaging might lead to sub-optimal results due to the permutation invariance of NN parameters. To address this issue, [18] proposed probabilistic federated neural matching (PFNM), which is only applicable to fully connected feed-forward networks. The recent work [17] proposed federated matched averaging (FedMA) as a layer-wise extension of PFNM to accommodate CNNs and LSTMs. However, the Bayesian non-parametric mechanism in PFNM and FedMA may be vulnerable to model poisoning attack [7], [9], [19], while some simple defences, such as norm thresholding and differential privacy, were discussed in [8]. We note that these ideas are complementary to FedMGDA+ and we plan to investigate possible integrations of them in future work.

Lastly, we note that there is significant interest in standardizing the benchmarks, protocols and evaluations in FL, see for

instance [20], [21]. We have spent significant efforts in adhering to the suggested rules there, by reporting on common datasets, open sourcing our code and including all experimental details.

III. Problem Setup

We recall the federated learning (FL) framework of [1] and point out a simple interpretation that seemingly unifies different implementations. We consider FL with m users (edge devices), where the i-th user is interested in minimizing a function $f_i : \mathbb{R}^d \to \mathbb{R}, i = 1, \ldots, m$, defined on a shared model parameter $\mathbf{w} \in \mathbb{R}^d$. Typically, each user function f_i also depends on the respective user's local (private) data $\mathcal{D}_i$. The main goal in FL is to *collectively and efficiently* optimize *individual* objectives $\{f_i\}$ while meeting challenges such as those mentioned in the Introduction (Section I): non-iid distribution of user data, limited communication, user privacy, fairness, robustness, etc.

McMahan *et al.* [1] proposed FedAvg to optimize the arithmetic average of individual user functions:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}), \quad \text{where} \quad \mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) := \sum_{i=1}^m \lambda_i f_i(\mathbf{w}). \tag{1}$$

The weights λ_i need to be specified *beforehand*. Typical choices include the dataset size at each user, the "importance" of each user, or simply uniform, i.e., $\lambda_i \equiv 1/m$. FedAvg works as follows: At each round, a (random) subset of users is selected, each of which performs k epochs of local (full or minibatch) gradient descent:

$$\text{for all } i \text{ in parallel}, \quad \mathbf{w}^i \leftarrow \mathbf{w}^i - \eta \nabla f_i(\mathbf{w}^i), \tag{2}$$

and then the weights are averaged at the server side:

$$\mathbf{w} \leftarrow \sum_i \lambda_i \mathbf{w}^i, \tag{3}$$

which is finally broadcast to the users in the next round. The number of local epochs k turns out to be a key factor. Setting $k = 1$ amounts to solving (1) by the usual gradient descent algorithm, while setting $k = \infty$ (and assuming convergence for each local function f_i) amounts to (repeatedly) averaging the respective minimizers of f_i's. We now give a new interpretation of FedAvg that yields insights on what it *optimizes* with an intermediate k.

Our interpretation is based on the proximal average [22]. Recall that the Moreau envelope and proximal map of a convex[1] function f is defined respectively as:

$$\mathsf{M}^\eta_f(\mathbf{w}) = \min_{\mathbf{x}} \frac{1}{2\eta}\|\mathbf{x}-\mathbf{w}\|_2^2 + f(\mathbf{x}), \tag{4}$$

$$\mathsf{P}^\eta_f(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{x}} \frac{1}{2\eta}\|\mathbf{x}-\mathbf{w}\|_2^2 + f(\mathbf{x}). \tag{5}$$

[1] For nonconvex functions, similar results hold once we address multi-valuedness of the proximal map, see [23].

Given a set of convex functions $\mathbf{f} = (f_1, \ldots, f_m)$ and positive weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ that sum to 1, we define the proximal average as the *unique* function $\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}$ such that $\mathsf{P}^\eta_{\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}} = \sum_i \lambda_i \mathsf{P}^\eta_{f_i}$. In other words, the proximal map of the proximal average is the average of proximal maps. More concretely, [22] gave the following explicit, albeit complicated, formula for the proximal average:

$$\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) = \min_{\mathbf{w}_1,\ldots,\mathbf{w}_m} \sum_{i=1}^m \lambda_i \left[f_i(\mathbf{w}_i) + \frac{1}{2\eta}\|\mathbf{w}_i\|_2^2 \right] - \frac{1}{2\eta}\|\mathbf{w}\|_2^2 \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^m \lambda_i \mathbf{w}_i = \mathbf{w}. \tag{7}$$

From the above formula we can easily derive that

$$\mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) := \lim_{\eta\to 0+} \mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) = \textstyle\sum_i \lambda_i f_i(\mathbf{w}),$$

$$\mathsf{A}^\infty_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) := \lim_{\eta\to\infty} \mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) = \min_{\sum_i \lambda_i \mathbf{w}_i = \mathbf{w}} \textstyle\sum_i \lambda_i f_i(\mathbf{w}_i).$$

Interestingly, we can now interpret FedAvg in two extreme settings as minimizing the proximal average:

- FedAvg with $k = 1$ local step is exactly the same as minimizing the proximal average $\mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w})$ with $\eta = 0$. This is clear from the objective (1) of FedAvg (as our notation already suggests).
- FedAvg with $k = \infty$ local steps is exactly the same as minimizing the proximal average $\mathsf{A}^\infty_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w})$ with $\eta = \infty$. Indeed,

$$\left\{ \min_{\mathbf{w}} \ \mathsf{A}^\infty_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}) \right\} = \min_{\mathbf{w}_1,\ldots,\mathbf{w}_m} \sum_i \lambda_i f_i(\mathbf{w}_i), \tag{8}$$

  where the right-hand side decouples and hence $\mathbf{w}_i$ at optimality is a minimizer of f_i (recall that $\boldsymbol{\lambda} \geq 0$).

Therefore, we may interpret FedAvg with an intermediate k as minimizing $\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}$ with an intermediate η. More interestingly, if we apply the PA-PG algorithm in [12, Algo. 2] to minimize $\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}$, we obtain the simple update rule

$$\mathbf{w} \leftarrow \textstyle\sum_i \lambda_i \mathsf{P}^\eta_{f_i}(\mathbf{w}), \tag{9}$$

where the proximal maps are computed in parallel at the user's side. We note that the recent FedProx algorithm [11] is essentially a randomized version of (9). Crucially, we do not need to evaluate the complicated formula (6) as the update (9) only requires its proximal map, which by definition is the average of the individual proximal maps (computed by each user separately). Moreover, the difference between the proximal average $\mathsf{A}^\eta_{\mathbf{f},\boldsymbol{\lambda}}$ and the arithmetic average $\mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}$ can be *uniformly* bounded using the Lipschitz constant of each function f_i [12]. Thus, for small step size η, FedAvg (with any finite k) and FedProx all minimize some *approximate* form of the arithmetic average in (1).

How to set the weights $\boldsymbol{\lambda}$ in FedAvg has been a major challenge. In FL, data is distributed in a highly non-iid and unbalanced fashion, so it is not clear if some chosen arithmetic

average in (1) would really satisfy one's actual intention. A second issue with the arithmetic average in (1) is its well-known non-robustness against malicious manipulations, which has been exploited in recent adversarial attacks [9]. Instead, Agnostic FL (AFL [5]) aims to optimize the worst-case loss:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\lambda}\in\Lambda} \mathsf{A}^0_{\mathbf{f},\boldsymbol{\lambda}}(\mathbf{w}), \tag{10}$$

where the set Λ might cover reality better than any specific $\boldsymbol{\lambda}$ and provide some minimum guarantee for all users (hence achieving mild fairness). On the other hand, the worst-case loss in (10) is perhaps even more non-robust against adversarial attacks. For instance, adding a positive constant to some loss f_i can make it dominate the entire optimization process. The recent work q-FedAvg [6] proposes an ℓ_q norm interpolation between FedAvg (essentially ℓ_1 norm) and AFL (essentially ℓ_∞ norm). By tuning q, q-FedAvg can achieve better compromise than FedAvg or AFL.

IV. Multi-Objective Minimization (MoM)

Multi-objective minimization (MoM) refers to the setting where *multiple* scalar objective functions, possibly incompatible with each other, need to be minimized *simultaneously*. It is also called *vector optimization* [24] because the objective functions can be combined into a single vector-valued function. In mathematical terms, MoM can be written as

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbf{f}(\mathbf{w}) := (f_1(\mathbf{w}), f_2(\mathbf{w}), \ldots, f_m(\mathbf{w})), \tag{11}$$

where the minimum is defined wrt the *partial* ordering:

$$\mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{z}) \Leftrightarrow \forall i = 1, \ldots, m,\ f_i(\mathbf{w}) \leq f_i(\mathbf{z}). \tag{12}$$

(We remind that algebraic operations such as $\leq$ and $+$, when applied to a vector with another vector *or scalar*, are always performed component-wise.) Unlike single objective optimization, with multiple objectives it is possible that

$$\mathbf{f}(\mathbf{w}) \not\leq \mathbf{f}(\mathbf{z}) \text{ and } \mathbf{f}(\mathbf{z}) \not\leq \mathbf{f}(\mathbf{w}), \tag{13}$$

in which case we say $\mathbf{w}$ and $\mathbf{z}$ are not comparable.

We call $\mathbf{w}^*$ a Pareto optimal solution of (11) if its objective value $\mathbf{f}(\mathbf{w}^*)$ is a minimum element (wrt the partial ordering in (12)), or equivalently for any $\mathbf{w}$, $\mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{w}^*)$ implies $\mathbf{f}(\mathbf{w}) = \mathbf{f}(\mathbf{w}^*)$. In other words, it is not possible to improve *any* component objective in $\mathbf{f}(\mathbf{w}^*)$ without compromising some other objective. Similarly, we call $\mathbf{w}^*$ a *weakly* Pareto optimal solution if there does not exist any $\mathbf{w}$ such that $\mathbf{f}(\mathbf{w}) < \mathbf{f}(\mathbf{w}^*)$, i.e., it is not possible to improve *all* component objectives in $\mathbf{f}(\mathbf{w}^*)$. Clearly, any Pareto optimal solution is also weakly Pareto optimal but the converse may not hold.

We point out that the optimal solutions in MoM are usually a set (in general of infinite cardinality) [25], and without additional subjective preference information, all Pareto optimal solutions are considered equally good (as they are not comparable against each other). This is *fundamentally* different from the single objective case.

From now on, for simplicity we assume all objective functions are continuously differentiable but not necessarily convex (to accommodate deep models). Finding a (weakly) Pareto optimal solution in this setting is quite challenging (already so in the single objective case). Instead, we will contend with Pareto stationary solutions, namely those that satisfy an intuitive first order necessary condition:

Definition 1 (Pareto-stationarity [25]): We call $\mathbf{w}^*$ Pareto-stationary iff *some* convex combination of the gradients $\{\nabla f_i(\mathbf{w}^*)\}$ vanishes, i.e., there exists some $\boldsymbol{\lambda} \geq 0$ such that $\sum_i \lambda_i = 1$ and $\sum_i \lambda_i \nabla f_i(\mathbf{w}^*) = \mathbf{0}$.

Lemma 1 ([25]): Any Pareto optimal solution is Pareto stationary. Conversely, if all functions are convex, then any Pareto stationary solution is weakly Pareto optimal.

Needless to say, the above results reduce to the familiar ones for the single objective case ($m = 1$).

There exist many algorithms for finding Pareto stationary solutions. We briefly review three popular ones that are relevant for us, and refer the reader to the excellent monograph [26] for more details.

Weighted approach. Let $\boldsymbol{\lambda} \in \Delta$ (the simplex) and consider the following single, weighted objective:

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \lambda_i f_i(\mathbf{w}). \tag{14}$$

This is essentially the approach taken by FedAvg, with any (global) minimizer of (14) being weakly Pareto optimal (in fact, Pareto optimal if all weights λ_i are positive). From Definition 1 it is clear that any stationary solution of the weighted scalar problem (14) is a Pareto stationary solution of the original MoM (11). Note that the scalarization weights $\boldsymbol{\lambda}$, once chosen, are fixed throughout. Different $\boldsymbol{\lambda}$ leads to different Pareto stationary solutions.

ϵ-constraint. Let $\boldsymbol{\epsilon} \in \mathbb{R}^{m-1}$, $\iota \in \{1, \ldots, m\}$ and consider the following constrained scalar problem:

$$\min_{\mathbf{w}} f_\iota(\mathbf{w}) \tag{15}$$

$$\text{s.t. } f_i(\mathbf{w}) \leq \epsilon_i,\ \forall i \neq \iota. \tag{16}$$

Assuming the constraints are satisfiable, then any (global) minimizer of (15) is again weakly Pareto optimal. The ϵ-constraint approach is closely related to the weighted approach above, through the usual Lagrangian reformulation. Both require fixing an $m-1$ dimensional parameter in advance ($\boldsymbol{\lambda}$ *vs.* $\boldsymbol{\epsilon}$), though.

Chebyshev approach. Let $\mathbf{s} \in \mathbb{R}^m$ and consider the minimax problem (where recall that Δ is the simplex constraint):

$$\min_{\mathbf{w}} \max_{\boldsymbol{\lambda}\in\Delta} \boldsymbol{\lambda}^\top (\mathbf{f}(\mathbf{w}) - \mathbf{s}). \tag{17}$$

Again, any (global) minimizer is weakly Pareto optimal. Here $\mathbf{s}$ is a fixed vector that ideally lower bounds $\mathbf{f}$. This is essentially the approach taken by AFL [5] with $\mathbf{s} = \mathbf{0}$.

V. FL as Multi-Objective Minimization

Having introduced both FL and MoM, and observed some connections between the two, it is very natural to treat each

user function f_i in FL as a separate objective in MoM and aim to optimize them *simultaneously* as in (11). This will be the main approach we follow below, which, to the best of our knowledge, has not been formally explored before (despite of the apparent connections that we saw in the previous section, perhaps retrospectively). In particular, we will extend the multiple gradient descent algorithm [25] in MoM to FL, draw connections to existing FL algorithms, and prove convergence properties of our extended algorithm FedMGDA+. Very importantly, the notion of Pareto optimality and stationarity immediately enforces fairness among users, as we are discouraged from improving certain users by sacrificing others.

To further motivate our development, let us compare to the objective in AFL [5]:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\lambda}\in\Delta} \boldsymbol{\lambda}^\top \mathbf{f}(\mathbf{w}) \quad \equiv \quad \min_{\mathbf{w}} \max_{i=1,\ldots,m} f_i(\mathbf{w}), \tag{18}$$

where Δ denotes the simplex[2]. By optimizing the *worst* loss than the *average* loss in FedAvg, AFL provides some guarantee to all users hence achieving some form of fairness. However, note that AFL's objective (18) is not robust against adversarial attacks. In fact, if a malicious user artificially "inflates" its loss f_i (e.g., even by adding/multiplying a constant), it can completely dominate and mislead AFL to solely focus on optimizing its performance. The same issue applies to q-FedAvg [6], albeit with a less dramatic effect if q is small.

AFL's objective (18) is very similar to the Chebyshev approach in MoM (see Section IV), which inspires us to propose the following iterative algorithm for solving (11):

$$\tilde{\mathbf{w}}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \max_{\boldsymbol{\lambda}\in\Delta} \boldsymbol{\lambda}^\top (\mathbf{f}(\mathbf{w}) - \mathbf{f}(\tilde{\mathbf{w}}_t)), \tag{19}$$

where we *adaptively* "center" the user functions using function values from the previous iteration. When the functions f_i are smooth, we apply the quadratic bound to obtain:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \max_{\boldsymbol{\lambda}\in\Delta} \boldsymbol{\lambda}^\top J_{\mathbf{f}}^\top(\mathbf{w}_t)(\mathbf{w}-\mathbf{w}_t) + \frac{1}{2\eta}\|\mathbf{w}-\mathbf{w}_t\|^2, \tag{20}$$

where $J_{\mathbf{f}} = [\nabla f_1, \ldots, \nabla f_m] \in \mathbb{R}^{d\times m}$ is the Jacobian and $\eta > 0$ is the step size. Crucially, note that $\mathbf{f}(\mathbf{w}_t)$ does not appear in the above bound (20) since we subtracted it off in (19). Since (20) is convex in $\mathbf{w}$ and concave in $\boldsymbol{\lambda}$ we can swap min with max and obtain the dual:

$$\max_{\boldsymbol{\lambda}\in\Delta} \min_{\mathbf{w}} \boldsymbol{\lambda}^\top J_{\mathbf{f}}^\top(\mathbf{w}_t)(\mathbf{w}-\mathbf{w}_t) + \frac{1}{2\eta}\|\mathbf{w}-\mathbf{w}_t\|^2. \tag{21}$$

Solving $\mathbf{w}$ by setting its derivative to $\mathbf{0}$ we arrive at:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{d}_t, \quad \mathbf{d}_t = J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}_t^*, \tag{22}$$

$$\text{where} \quad \boldsymbol{\lambda}_t^* = \operatorname*{argmin}_{\boldsymbol{\lambda}\in\Delta} \|J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}\|^2. \tag{23}$$

[2] To be precise, AFL restricted $\boldsymbol{\lambda}$ to a subset $\Lambda \subseteq \Delta$. We simply set $\Lambda = \Delta$ to ease the discussion.

Note that $\mathbf{d}_t$ is precisely the minimum-norm element in the convex hull of the columns (i.e., gradients) in the Jacobian $J_{\mathbf{f}}$, and finding $\boldsymbol{\lambda}_t^*$ amounts to solving a simple quadratic program. The resulting iterative algorithm in (22) is known as multiple gradient descent algorithm (MGDA), which has been (re)discovered in [25], [27], [28] and recently applied to multitask learning in [29], [30] and to training GANs in [31]. Our concise derivation here reveals some new insights about MGDA, in particular its connection to AFL.

To adapt MGDA to the federated learning setting, we propose the following extensions.

Balancing user average performance and fairness. We observe that the MGDA update in (22) resembles FedAvg, with the crucial difference that MGDA *automatically* tunes the dual weighting variable $\boldsymbol{\lambda}$ in each step while FedAvg presets $\boldsymbol{\lambda}$ based on *a priori* information about the user functions (or simply uniform in lack of such information). Importantly, the direction $\mathbf{d}_t$ found in MGDA is a common descent direction for *all participating objectives*:

$$\begin{aligned}\mathbf{f}(\mathbf{w}_{t+1}) &\le \mathbf{f}(\mathbf{w}_t) + J_{\mathbf{f}}^\top(\mathbf{w}_t)(\mathbf{w}_{t+1}-\mathbf{w}_t) + \frac{1}{2\eta}\|\mathbf{w}_{t+1}-\mathbf{w}_t\|^2 \\ &\le \mathbf{f}(\mathbf{w}_t),\end{aligned} \tag{24}$$

where the first inequality follows from familiar smoothness assumption on $\mathbf{f}$ while the second inequality follows simply from plugging $\mathbf{w} = \mathbf{w}_t$ in (20) and noting that $\mathbf{w}_{t+1}$ by definition can only decrease (20) even more. It is clear that equality is attained iff $\mathbf{d}_t = J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}_t^* = \mathbf{0}$, i.e., $\mathbf{w}_t$ is Pareto-stationary (see Section IV). In other words, MGDA never sacrifices any participating objective to trade for more sizable improvements over some other objective, something FedAvg with a fixed weighting $\boldsymbol{\lambda}$ might attempt to do. On the other hand, FedAvg with a fixed weighting $\boldsymbol{\lambda}$ may achieve higher *average performance* under the weighting $\boldsymbol{\lambda}$. It is natural to introduce the following trade-off between average performance and fairness:

$$\text{update (22) with } \boldsymbol{\lambda}_t^* = \operatorname*{argmin}_{\boldsymbol{\lambda}\in\Delta, \|\boldsymbol{\lambda}-\boldsymbol{\lambda}_0\|_\infty \le \epsilon} \|J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}\|^2. \tag{25}$$

Clearly, setting $\epsilon = 0$ recovers FedAvg with *a priori* weighting $\boldsymbol{\lambda}_0$ while setting $\epsilon = 1$ recovers MGDA where the weighting variable $\boldsymbol{\lambda}$ is tuned without any restriction to achieve maximal fairness. In practice, with an intermediate $\epsilon \in (0,1)$ we may strike a desirable balance between the two (sometimes) conflicting goals. Moreover, even with the uninformative weighting $\boldsymbol{\lambda}_0 = \mathbf{1}/m$, using an intermediate ϵ allows us to upper bound the contribution of each user function to the common direction $\mathbf{d}_t$ hence achieve some form of robustness against malicious manipulations.

Robustness against malicious users through normalization. Existing works (e.g., [9], [32]) have demonstrated that the *average* gradient in FedAvg can be easily manipulated by even a single malicious user. While more robust aggregation strategies are studied recently (see e.g., [33]–[35]), they do not necessarily maintain the convergence properties of FedMGDA+ (e.g. finding a common descent direction and converging to a Pareto stationary solution). Instead, we propose to simply normalize

Algorithm 1: FedMGDA+.

for $t = 1, 2, \ldots$ **do**
  choose a subset I_t of $\lceil pm \rceil$ clients/users
  for $i \in I_t$ **do**
    $\mathbf{g}_i \leftarrow$ Client Update $(i, \mathbf{w}_t)$
    $\bar{\mathbf{g}}_i := \mathbf{g}_i / \|\mathbf{g}_i\|$ // normalize
  $\boldsymbol{\lambda}^* \leftarrow \operatorname{argmin}_{\boldsymbol{\lambda} \in \Delta, \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|_\infty \leq \epsilon} \| \sum_i \lambda_i \bar{\mathbf{g}}_i \|^2$
  $\mathbf{d}_t \leftarrow \sum_i \lambda_i^* \bar{\mathbf{g}}_i$ // common direction
  choose (global) step size η_t
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{d}_t$
Function Client Update $(i, \mathbf{w})$:
  $\mathbf{w}^0 \leftarrow \mathbf{w}$
  Repeat (k epochs)
    // split local data into r batches
    $\mathcal{D}_i \rightarrow \mathcal{D}_{i,1} \cup \cdots \cup \mathcal{D}_{i,r}$
    for $j \in \{1, \ldots, r\}$ **do**
      $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f_i(\mathbf{w}; \mathcal{D}_{i,j})$
  return $\mathbf{g} := \mathbf{w}^0 - \mathbf{w}$ to server

the gradients from each user to unit length, based on the following considerations: (a) Normalizing the (sub)gradient is common for specialists in nonsmooth and stochastic optimization [36] and sometimes eases step size tuning. (b) Solving the weights $\boldsymbol{\lambda}_t^*$ in (22) with normalized gradients still guarantees fairness, i.e., the resulting direction $\mathbf{d}_t$ is descending for all participating objectives (by a completely similar reasoning as the remark after (24)). (c) Normalization restores robustness against multiplicative "inflation" from any malicious user, which, combined with MGDA's built-in robustness against additive "inflation" (see (19)), offers reasonable robustness guarantees against adversarial attacks.

Balancing communication and on-device computation. Communication between user devices and the central server is heavily constrained in FL, due to a variety of reasons mentioned in Section III. On the other hand, modern edge devices are capable of performing reasonable amount of on-device computations. Thus, we allow each user device to perform multiple local updates before communicating its update $\mathbf{g} = \mathbf{w}^0 - \mathbf{w}^{\blacksquare}$, namely the difference between the initial $\mathbf{w}^0$ and the final $\mathbf{w}^{\blacksquare}$, to the central server. The server then calls the (extended) MGDA to perform a global update, which will be broadcast to the next round of user devices. We note that similar strategy was already adopted in many existing FL systems (e.g., [1], [6], [11]).

Subsampling to alleviate non-iid and enhance throughput. Due to the massive number of edge devices in FL, it is not realistic to expect most devices to participate at each or even most rounds. Consequently, the current practice in FL is to select a (different) subset of user devices to participate in each round [1]. Moreover, randomly subsampling user devices can also help combat the non-iid distribution of user-specific data (e.g., [1], [10]). Here we point out an important advantage of our MGDA-based algorithm: its update is along a common descending direction (see (24)), meaning that the objective of any *participating* user can only decrease. We believe this unique property of MGDA provides strong incentive for users to participate in FL. To our best knowledge, existing FL algorithms do not provide similar algorithmic incentives. Last but not the least, subsampling also solves a degeneracy issue in MGDA: when the number of participating users exceeds the dimension d, the Jacobian $J_{\mathbf{f}}$ has full row-rank hence (22) achieves Pareto-stationarity in a single iteration and stops making progress. Subsampling removes this undesirable effect and allows different subsets of users to be continuously optimized.

With the above extensions, we summarize our extended algorithm FedMGDA+ in Algorithm 1, and we prove the following convergence guarantees (precise statements and proofs can be found in the supplementary material, Section A):

Theorem 1a: Let each user function f_i be L-Lipschitz smooth and M-Lipschitz continuous, and choose step size η_t so that $\sum_t \eta_t = \infty$ and $\sum_t \sigma_t \eta_t < \infty$, where $\sigma_t^2 := \mathbf{E}\|\mathbf{d}_t - \hat{\mathbf{d}}_t\|^2$ with

$$\mathbf{d}_t := J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}_t, \ \boldsymbol{\lambda}_t = \operatorname*{argmin}_{\boldsymbol{\lambda} \in \Delta} \|J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}\|, \tag{26}$$

$$\hat{\mathbf{d}}_t := \hat{J}_{\mathbf{f}}(\mathbf{w}_t)\hat{\boldsymbol{\lambda}}_t, \ \hat{\boldsymbol{\lambda}}_t = \operatorname*{argmin}_{\boldsymbol{\lambda} \in \Delta} \|\hat{J}_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}\|. \tag{27}$$

Then, with $k = r = 1$ we have:

$$\min_{t=0,\ldots,T} \mathbf{E}\|J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}_t\|^2 \rightarrow 0. \tag{28}$$

Here k is the number of local updates and r is the number of minibatches in each local update. The convergence rate depends on how quickly the "variance" term σ_t of the stochastic common descent direction $\hat{d}_t$ diminishes (if at all), which in turn depends on how aggressively we subsample users or how heterogeneous the users are.

For deterministic gradient updates, we can prove convergence even with more local updates (i.e., $k > 1$):

Theorem 1b: Let each user function f_i be L-Lipschitz smooth and M-Lipschitz continuous. For any number of local updates k, if the global step size $\eta_t \rightarrow 0$ with $\sum_t \eta_t = \infty$, local learning rate $\eta_t^l \rightarrow 0$ and $\varepsilon_t := \|\boldsymbol{\lambda}_t - \hat{\boldsymbol{\lambda}}_t\| \rightarrow 0$, then we have:

$$\min_{t=0,\ldots,T} \|J_{\mathbf{f}}(\mathbf{w}_t)\boldsymbol{\lambda}_t\|^2 \rightarrow 0. \tag{29}$$

Please refer to the supplementary material, Section A, for the precise statement of the theorem and its proof. We note that one natural approach to bound the deviation ε_t is by applying the ϵ-constrained version of FedMGDA. For example, if $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}_0\|_\infty \leq \epsilon_t$, and ϵ_t is bounded, then $\varepsilon_t \leq 2\sqrt{m}\epsilon_t$ is also bounded. Thus, $\varepsilon_t \rightarrow 0$ when $\epsilon_t \rightarrow 0$. Moreover, when $k = 1$, we do not need the local learning rate η_t^l to decay for convergence; in addition, if $\varepsilon_t \equiv 0$ (e.g. in FedAvg), then our convergence guarantee reduces to the usual one for gradient descent, which is expected since we know FedAvg with $k = 1, r = 1$ is the same as centralized gradient descent. Lastly, we note that when $k > 1$, local learning rate η_t^l must vanish in order to obtain convergence. This importance of local learning rate decay is also pointed out in [15].

When the functions f_i are convex, we can derive a finer result:

TABLE I
DATASET SUMMARY

Dataset	Train Clients	Train samples	Test clients	Test samples	Batch size
CIFAR-10 [38]	100	50000	100	10000	$\{10, \infty\}$
F-MNIST [39]	100	60000	100	10000	$\{10, \infty\}$
FEMNIST [20]	3406	709385	3406	80011	$\{20, \infty\}$
Shakespeare [11]	31	92959	31	23255	$\{10\}$
Adult [40]	2	32561	2	16281	$\{10\}$

Theorem 2: Suppose each user function f_i is convex and M-Lipschitz continuous. Suppose at each round `FedMGDA+` includes a strongly convex user function whose weight is bounded away from 0. Then, with the choice $\eta_t = \frac{2}{c(t+2)}$ and $k = r = 1$, we have

$$\mathbf{E}\|\mathbf{w}_t - \mathbf{w}_t^*\|^2 \leq \frac{4M^2}{c^2(t+3)}, \tag{30}$$

and $\mathbf{w}_t - \mathbf{w}_t^* \to 0$ almost surely, where $\mathbf{w}_t^*$ is the nearest Pareto stationary solution to $\mathbf{w}_t$ and c is some constant.

A slightly stronger result where we also allow some user functions to be nonconvex can be found in the supplementary material, Section A. The same results hold if the gradient normalization is bounded away from 0 (otherwise we are already close to Pareto stationarity). For $r, k > 1$, using a similar argument as in Section III, we expect `FedMGDA+` to optimize some *proxy problem* (such as the proximal average), and we leave the thorough theoretical analysis for future work.

We remark that convergence rate for MGDA, even when restricted to the deterministic case, was only derived recently in [37]. The stochastic case (that we consider here) is much more challenging and our theorems provide one of the first convergence guarantees for `FedMGDA+`. We wish to emphasize that `FedMGDA+` is not just an alternative algorithm for `FL` practitioners; it can be used as a post-processing step to enhance existing `FL` systems or combined with existing `FL` algorithms (such as `FedProx` or `q-FedAvg`). This is particularly appealing with *nonconvex* user functions as MGDA is capable of converging to all Pareto stationary points while approaches such as `FedAvg` do not necessarily enjoy this property even when we enumerate the weighting $\boldsymbol{\lambda}_0$ [26]. Furthermore, it is possible to find multiple or even enumerate all Pareto optimal solutions (i.e. the Pareto front). For instance, we may run `FedMGDA+` multiple times with different random seeds or initializations. As shown by [30], we could also incorporate additional linear constraints in (22) to encode one's preference and encourage more diverse solutions. However, these techniques become less effective in higher dimensions (i.e., when the number of users is large) and in communication limited settings. Practically, the server may dynamically adjust the linear constraints in (22) to steer the algorithm to a more desirable Pareto stationary solution.

Lastly, we mention that finding the common descent direction (i.e., Line 6 of Algorithm 1) is a standard quadratic programming (QP) problem that is solved only at the server side. For moderate number of (sampled) users, it suffices to employ a generic QP solver while for large number of users we could also solve λ efficiently using for instance the conditional gradient algorithm [29], with per-step complexity proportional to the model dimension and the number of participating users. For our experiments below, we used a generic QP sovler and we observed that this overhead is negligible, resulting almost the same overall running time for `FedAvg` and `FedMGDA`.

VI. EXPERIMENTS

A. Experimental Setups

In this subsection we provide experimental details including dataset descriptions, sampling schemes, model configurations and hyper-parameter settings. A quick summary of the datasets that we use can be found in Table I. We have two parameters in `FedMGDA+` to control the total number of local updates in each communication round: k, the number of local epochs, and $r = n/b$, the number of updates in each local epoch. Here n is the number of samples at each user (assumed the same for simplicity) while b is the minibatch size for each local update. As observed by [1, e.g. Table 2], having a larger k is similar as having a smaller b (or equivalently a larger r), in terms of total number of local updates. Moreover, $k = 1$ with a suitable b usually leads to satisfying performance while very large k can result in plateau or divergence. Thus, in our experiments we fix $k = 1$ while vary b to reduce the total number of hyper-parameters. This corresponds to a single pass of the local data at each user in every communication round.

1) CIFAR-10 [38] and Fashion MNIST [39] Datasets: In order to create a non-i.i.d. dataset, we follow a similar sampling procedure as in [1]: first we sort all data points according to their classes. Then, they are split into 500 shards, and each user is randomly assigned 5 shards of data. By considering 100 users, this procedure guarantees that no user receives data from more than 5 classes and the data distribution of each user is different from each other. The local datasets are balanced–all users have the same amount of training samples. The local data is split into train, validation, and test sets with percentage of 80%, 10%, and 10%, respectively. In this way, each user has 400 data points for training, 50 for test, and 50 for validation. We use a CNN model which resembles the one in [1], with two convolutional layers followed by three fully connected layers. The details are included in Table II for CIFAR-10 and in Table III for Fashin MNIST. To update the local models at each user using its local data, we apply stochastic gradient descent (SGD) with local batch size $b = 10$, local epoch $k = 1$, and local learning rate $\eta = 0.01$, or $b = 400$, $k = 1$, and $\eta = 0.1$. To model the fact that not all users may participate in each communication round, we employ a parameter p to control the fraction of participating users: $p = 0.1$ is the

TABLE II
CIFAR-10 MODEL

Layer	Output Shape	# of Trainable Parameters	Activation	Hyper-parameters
Input	$(3, 32, 32)$	0		
Conv2d	$(64, 28, 28)$	4864	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(64, 14, 14)$	0		pool size=$(2, 2)$
LocalResponseNorm	$(64, 14, 14)$	0		size=2
Conv2d	$(64, 10, 10)$	102464	ReLU	kernel size =5; strides=$(1, 1)$
LocalResponseNorm	$(64, 10, 10)$	0		size=2
MaxPool2d	$(64, 5, 5)$	0		pool size=$(2, 2)$
Flatten	1600	0		
Dense	384	614784	ReLU	
Dense	192	73920	ReLU	
Dense	10	1930	softmax	
Total		797962		

TABLE III
FASHION MNIST MODEL

Layer	Output Shape	# of Trainable Parameters	Activation	Hyper-parameters
Input	$(1, 28, 28)$	0		
Conv2d	$(10, 24, 24)$	260	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(10, 12, 12)$	0		pool size=$(2, 2)$
Conv2d	$(20, 8, 8)$	5020	ReLU	kernel size =5; strides=$(1, 1)$
MaxPool2d	$(20, 4, 4)$	0		pool size=$(2, 2)$
Dropout2d	$(20, 4, 4)$	0		$p = 0.5$
Flatten	320	0		
Dense	50	16050	ReLU	
Dropout	50	0		$p = 0.5$
Dense	10	510	softmax	
Total		21840		

TABLE IV
FEDERATED EMNIST MODEL [15]

Layer	Output Shape	# of Trainable Parameters	Activation	Hyper-parameters
Input	$(1, 28, 28)$	0		
Conv2d	$(32, 26, 26)$	320		kernel size =3; strides=$(1, 1)$
Conv2d	$(64, 24, 24)$	18496	ReLU	kernel size =3; strides=$(1, 1)$
MaxPool2d	$(64, 12, 12)$	0		pool size=$(2, 2)$
Dropout	$(64, 12, 12)$	0		$p = 0.25$
Flatten	9216	0		
Dense	128	1179776		
Dropout	128	0		$p = 0.5$
Dense	62	7998	softmax	
Total		1206590		

default setting which means that only 10% of users participate in each communication round.

2) Federated EMNIST Dataset [20]: For this experimental setup, we use the same dataset, model, and hyper-parameters as [15]. We use the federated EMNIST dataset of [20]. The dataset consists of images of digits, and English characters—both lower and upper cases, with 62 classes in total. The images are partitioned by their authors in a way that naturally makes the dataset heterogeneous and unbalanced. We use the model described in Table IV and the following hyper-parameters: local learning rate $\eta = 0.1$ and selecting 10 clients per communication round as recommended. The only difference between our setup and the one in [15] is that we use local epoch $k = 1$ for all algorithms.

3) Shakespeare Dataset [11]: For experiments on the Shakespeare dataset, we use the same model, data pre-processing and sampling procedure as in `q-FedAvg` paper [6]. The dataset is built from *The Complete Works of William Shakespeare*, where each role in the play represents one user. Following [11], we subsample 31 users to train a neural language model for next character prediction. Each character is embedded in an 8-dimensional space and the sequence length is 80 characters. The model we use is a two-layer LSTM (with hidden size 256) followed by one dense layer [1], [11]. Joint hyper-parameters that are shared by all algorithms include: total communication rounds $T = 200$, local batch size $b = 10$, local epoch $k = 1$, and local optimizer being SGD, unless otherwise stated.

TABLE V
HYPERPARAMETERS USED IN OUR EXPERIMENTS.

Name	Parameters
AFL	$\gamma_\lambda \in \{0.01, 0.1, 0.2, 0.5\}, \gamma_w \in \{0.01, 0.1\}$
q-FedAvg	$q \in \{0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10\}$, $L \in \{0.1, 1, 10\}$
FedMGDA+	$\eta \in \{0.5, 1, 1.5, 2\}$, and Decay $\in \{0, \frac{1}{40}, \frac{1}{30}, \frac{1}{20}, \frac{1}{10}, \frac{1}{3}, \frac{1}{2}\}$
FedAvg-n	$\eta \in \{0.5, 1, 1.5, 2\}$, and Decay $\in \{0, \frac{1}{40}, \frac{1}{30}, \frac{1}{20}, \frac{1}{10}, \frac{1}{3}, \frac{1}{2}\}$
FedProx	$\mu \in \{0.001, 0.01, 0.1, 0.5, 1, 10\}$
MGDA-Prox	$\mu = 0.1, \eta \in \{0.5, 1, 1.5, 2\}$, and Decay $\in \{0, \frac{1}{40}, \frac{1}{30}, \frac{1}{20}, \frac{1}{10}, \frac{1}{5}, \frac{1}{3}, \frac{1}{2}\}$

4) Adult Dataset [40]: Following the setting in AFL [5], we split the Adult dataset into two non-overlapping domains based on the *education* attribute—phd domain and non-phd domain. The resulting FL setting consists of two users each of which has data from one of the two domains. Further, data is pre-processed as in [6] to have 99 binary features. We use a logistic regression model for all FL algorithms mentioned in the main paper. Local data is split into train, validation, and test sets with percentage of 80%, 10%, and 10%, respectively. In each round, both users participate and the server aggregates their losses and gradients (or weights). Joint hyper-parameters that are shared by all algorithms include: total communication rounds $T = 500$, local batch size $b = 10$, local epoch $k = 1$, local learning rate $\eta = 0.01$, and local optimizer being SGD without momentum, unless otherwise stated. Algorithm-specific hyper-parameters will be mentioned in the appropriate places below. One important note is that the phd domain has only 413 samples while the non-phd domain has 32,148 samples, so the split is very unbalanced while training *only* on the phd domain yields inferior performance on all domains due to the insufficient sample size.

5) Hyper-Parameters: We evaluate the performance of different algorithms with a wide range of hyper-parameters, summarized in Table V. In particular, following [36] we tried sublinear $O(1/t)$ and exponential decay $O(\beta^t)$ learning rates η on the server, and a fixed local learning rate η for client updates. Eventually we settled on decaying η_t by a factor of β every 100 steps: $\eta_t = \beta^{[\frac{t}{100}]}$, where $\beta = \texttt{decay}^{100/T}$ and T is the total number of communication rounds (with e.g. decay = $1/10$). We note that [15] also found exponential decay to be most effective in their experiments. We use grid search to choose suitable local learning rates for all algorithms.

We evaluate our algorithm FedMGDA+ on several public datasets: CIFAR-10 [38], F-MNIST [39], Federated EMNIST [20], Shakespeare [11] and Adult [40], and compare to existing FL systems including FedAvg [1], FedProx [11], q-FedAvg [6], and AFL[3] [5]. In addition, from the discussions in Section V, one can envision several potential extensions of existing algorithms to improve their performance. So, we also compare to the following extensions: FedAvg-n which is FedAvg with gradient normalization, and MGDA-Prox which is FedMGDA+ with a proximal regularizer added to each user's loss function.[4] We distinguish between FedMGDA+ and FedMGDA which is a vanilla extension of MGDA to FL.

We point out that FL algorithms are to be deployed on smart devices with moderate computational capabilities. Thus, the models we chose to experiment on are medium-sized (see Tables II to IV for details), with similar complexity to the ones in FedAvg, q-FedAvg and AFL. We only report some representative results in the main paper, and defer the full set of experiments to the supplementary material, Section B.

B. Experimental Results

In this subsection we report experimental results about our proposed algorithm FedMGDA+ and compare it with state-of-the-art (SOTA) alternatives under a variety of performance metrics, including accuracy, robustness and fairness. We remind that the accuracy metric is exactly what FedAvg aims to optimize during training, and hence it has some advantage in this metric over other alternative algorithms such as FedMGDA+, AFL, and q-FedAvg, which all aim to bring some fairness among users, perhaps at some occasional, and hopefully small, loss of accuracy.

1) Recovering FedAvg: As mentioned in Section V, we can control the balance between the user average performance and fairness by tuning the ϵ-constraint in (25). Setting $\epsilon = 0$ recovers FedAvg while setting $\epsilon = 1$ recovers FedMGDA. To verify this empirically, we run (25) with different ϵ, and report results on CIFAR-10 in Fig. 1 for both iid and non-iid distributions of data (for results on F-MNIST, see supplementary material, Section B1). These results confirm that changing ϵ from 0 to 1 yields an interpolation between FedAvg and FedMGDA, as expected. Since FedAvg essentially optimizes the (uniformly) averaged training loss, it naturally performs the best under this metric (Fig. 1(c) and (d)). Nevertheless, it is interesting to note that some intermediate ϵ values actually lead to better user accuracy than FedAvg in the *non-iid* setting (Fig. 1(a)).

2) Robustness: We discussed earlier in Section V that the gradient normalization and MGDA's built-in robustness allow

[3] Experiments of AFL in the original work [5] and later works that compare with it (e.g. [6]) were reported on datasets with very few clients (2 or 3), possibly due to applicability reasons. We followed this convention in our work.

[4] One can also apply the gradient normalization idea to q-FedAvg; however, we observed from our experiments that the resulting algorithm is unstable particularly for large q values.

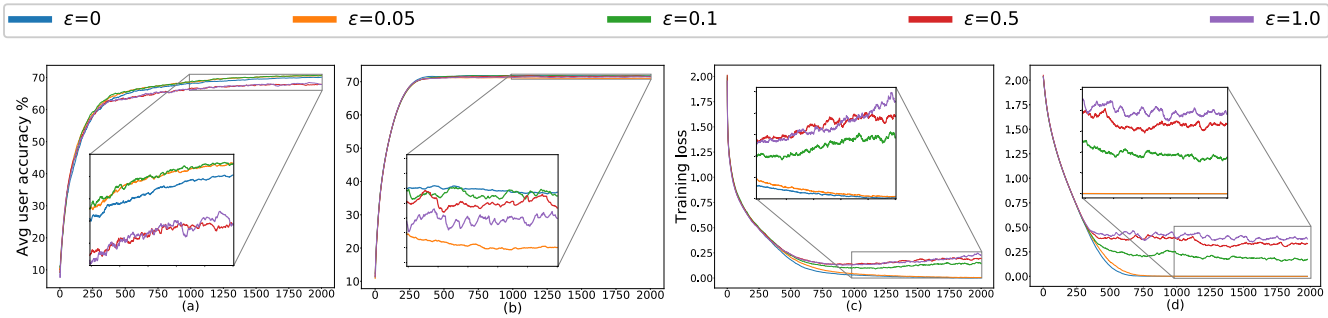

Fig. 1. Interpolation between FedAvg and FedMGDA on CIFAR-10. x-axis is the number of communication rounds. From left to right: (**a**) and (**b**) Average user accuracy in non-iid/iid setting resp. (**c**) and (**d**) Uniformly averaged training loss in non-iid/iid setting resp. Results are averaged over 5 runs with different random seeds.

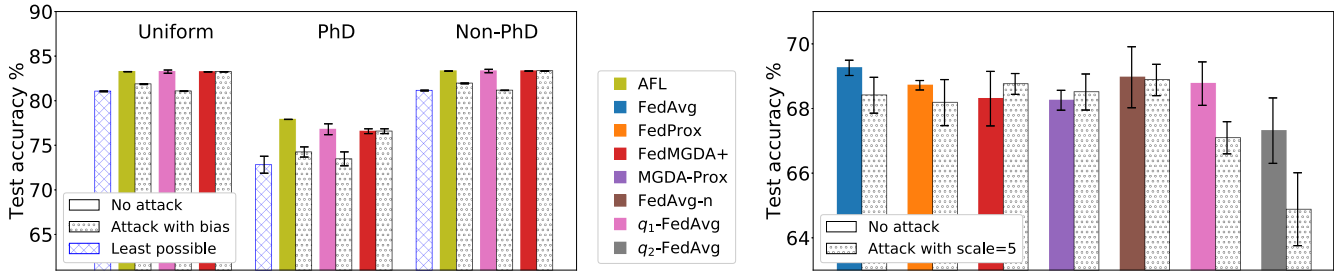

Fig. 2. (*Left*) Test accuracy of SOTA algorithms on Adult dataset with adversarial biases added to the loss of PhD domain; and compared to the baseline of training only on PhD domain. The scales of biases for AFL and q-FedAvg are different because AFL uses averaged loss while q-FedAvg uses (non-averaged) total loss. (*Right*) Test accuracy of different algorithms on CIFAR-10 in the presence of a malicious user who scales its loss function with a constant factor. All algorithms are run for 500 rounds on Adult and 1500 rounds on CIFAR-10. The reported results are averaged across 5 runs with different random seeds. For detailed hyperparameter setting, see supplementary material, Section B2.

FedMGDA+ to combat against certain adversarial attacks in practical FL deployment. We now empirically evaluate the robustness of FedMGDA+ against these attacks. We run various FL algorithms in the presence of a single malicious user who aims to manipulate the system by inflating its loss. We consider an adversarial setting where the attacker participates in each communication round and inflates its loss function by (i) adding a bias to it, or (ii) multiplying it by a scaling factor, termed the *bias* and *scaling* attack, respectively. In the first experiment, we simulate a bias attack on the Adult dataset by adding a constant bias to the underrepresented user, i.e. the PhD domain, since it's more natural to expect an attacker to be consisted of a small number of users. In this setup, the worst performance we can get is bounded by training the model using PhD data *only*. Results under the bias attack are presented in Fig. 2 (Left); also see the supplementary material, Section B2, for more results. We observe that AFL and q-FedAvg perform slightly better than FedMGDA+ without the attack; however, their performances deteriorate to a level close to the worst case scenario under the attack. In contrast, FedMGDA+ is not affected by the attack with any bias, which empirically supports our claim in Section V. Note that we did not include FedAvg in this comparison since from its definition it is clear that FedAvg, like FedMGDA+, is not affected by the bias attack. Fig. 2 (Right) shows the results of different algorithms on CIFAR-10 with and without an adversarial scaling. As mentioned earlier, q-FedAvg with gradient normalization is highly unstable particularly under the scaling attack, so we did not include its result here. From Fig. 2 (Right) it is immediate to see that (i) the scaling attack affects all algorithms that do not employ gradient normalization; (ii) q-FedAvg is the most affected under this attack; (iii) surprisingly, FedMGDA+ and, to a lesser extent, MGDA-Prox actually converge to slightly better Pareto solutions, compared to their own results under no scaling attack. The above results empirically verify the robustness of FedMGDA+ under perhaps the most common *bias* and *scaling* attacks.

3) Fairness: Lastly, we compare FedMGDA+ with existing FL algorithms using different notions of fairness on CIFAR-10. For the first experiment, we adopt the same fairness metric as [6], and measure fairness by calculating the variance of users' test error. We run each algorithm with different hyperparameters, and among the results, we pick the best ones in terms of average accuracy to be shown in Fig. 3; full table of results can be found in the supplementary material, Section B3. From this figure, we observe that (i) FedMGDA+ achieves the best average accuracy while its standard deviation is comparable with that of q-FedAvg; (ii) FedMGDA+ significantly outperforms FedMGDA, which clearly justifies our proposed modifications in Algorithm 1 to the vanilla MGDA; and (iii) FedMGDA+ outperforms FedAvg-n, which uses the same normalization step as FedMGDA+, in terms of average accuracy and standard deviation. These observations confirm the effectiveness of FedMGDA+ in inducing fairness. We perform the same experiment on the Federated EMNIST dataset, and observed similar results, which can be found in Table VI and in the supplementary material, Section B4.

In the next experiment, we show that FedMGDA+ not only yields a fair final solution but also maintains fairness during

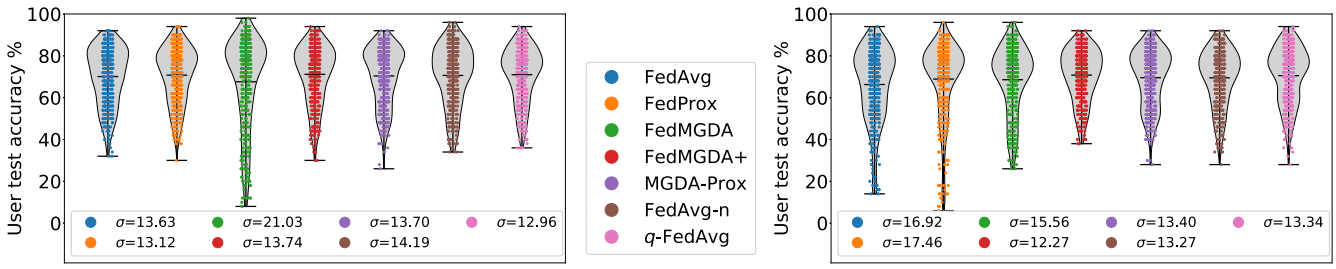

Fig. 3. Distribution of the user test accuracy on CIFAR-10: (**Left**) the algorithms are run for 2000 communication rounds and $b = 10$. The hyperparameters are: $\mu = 0.01$ for `FedProx`; $\eta = 1.5$ and decay $= 1/10$ for `FedMGDA+` and `FedAvg`; $\eta = 1.0$ and decay $= 1/10$ for `MGDA-Prox`; $q = 0.5$ and $L = 1.0$ for `q-FedAvg` . (*Right*) the algorithms are run for 3000 communication rounds and $b = 400$. The hyperparameters are: $\mu = 0.5$ for `FedProx`; $\eta = 1.0$ and decay $= 1/40$ for `FedMGDA+`, `MGDA-Prox`, and `FedAvg`; $q = 0.1$ and $L = 0.1$ for `q-FedAvg` . The reported statistics are averaged across 4 runs with different random seeds.

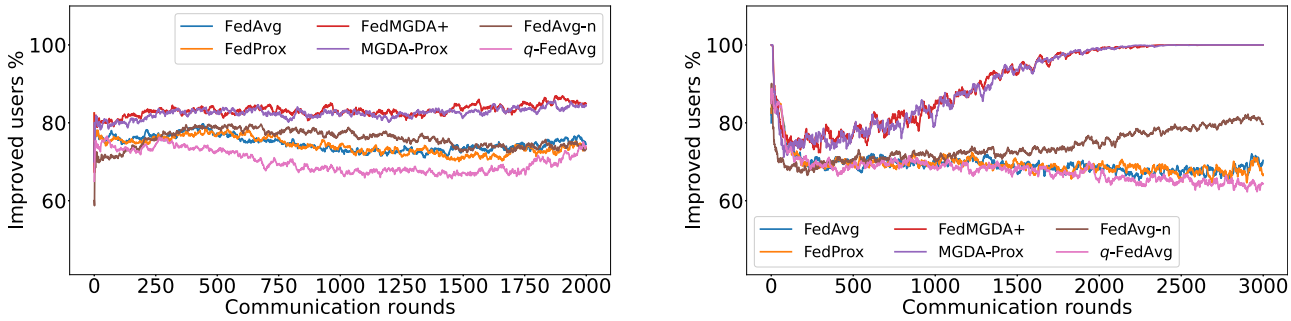

Fig. 4. The percentage of improved users in terms of training loss vs communication rounds on the CIFAR-10 dataset. Two representative cases are shown: (**Left**) the local batch size $b = 10$, and (**Right**) the local batch size $b = 400$. The results are averaged across 4 runs with different random seeds.

TABLE VI
TEST ACCURACY OF USERS ON FEDERATED EMNIST WITH FULL BATCH, 10 USERS PER ROUNDS, LOCAL LEARNING RATE $\eta = 0.1$, TOTAL COMMUNICATION ROUNDS 1500. THE REPORTED STATISTICS ARE AVERAGED ACROSS 4 RUNS WITH DIFFERENT RANDOM SEEDS

Algorithm	Average (%)	Std. (%)
`FedMGDA`	85.73 ± 0.05	14.79 ± 0.12
`FedMGDA+`	87.60 ± 0.20	13.68 ± 0.19
`MGDA-Prox`	87.59 ± 0.19	13.75 ± 0.18
`FedAvg`	84.97 ± 0.44	15.25 ± 0.36
`FedAvg-n`	87.57 ± 0.09	13.74 ± 0.11
`FedProx`	84.97 ± 0.45	15.26 ± 0.35
`q-FedAvg`	84.97 ± 0.44	15.25 ± 0.37

the entire training process in the sense that, in each round, it refrains from sacrificing the performance of any *participating* user for the sake of improving the overall performance. To the best of our knowledge, "fairness during training" has not been investigated before, in spite of having great practical implications—it encourages user participation. To examine this fairness, we run several experiments on CIFAR-10 and measure the percentage of improved *participants* in each communication round. Specifically, we measure the training loss before and after each round for all participating users, and report the percentage of those improved or stay unchanged.[5] Fig. 4 shows the percentage of improved participating users in each communication round in terms of training loss for two representative cases; see supplementary material, Section B5, for full results with different hyperparameters.

We can see that `FedMGDA+` consistently outperforms other algorithms in terms of percentage of improved users, which means that by using `FedMGDA+`, fewer users' performances get worse after each participation. Furthermore, we notice from Fig. 4 (Left) that, with local batch size $b = 10$, the percentage of improved users is less than 100%, which can be explained as follows: for small batch sizes (i.e., $b < |\mathcal{D}|$ where $\mathcal{D}$ represents a local dataset), the received updates from users are not the true gradients of users' losses given the global model (i.e., $\mathbf{g}_i \neq \nabla f_i(\mathbf{w})$); they are noisy estimates of the true gradients. Consequently, the common descent direction calculated by MGDA is noisy and may not always work for all participating users. To remove the effect of this noise, we set $b = |\mathcal{D}|$ which allows us to recover the true gradients from the users. The results are presented in Fig. 4 (Right), which confirms that, when step size decays (less overshooting), the percentage of improved users for `FedMGDA+` reaches towards 100% during training, as is expected.

VII. CONCLUSION

We have proposed a novel algorithm `FedMGDA+` for federated learning. `FedMGDA+` is based on multi-objective optimization and aims to converge to Pareto stationary solutions. `FedMGDA+` is simple to implement, has fewer hyperparameters to tune, and complements existing `FL` systems nicely. Most importantly, `FedMGDA+` is robust against additive and

[5] The percentage of improved users at time t is defined as $\sum_{i \in I_t} \mathbb{I}\{f_i(\mathbf{w}_{t+1}) \leq f_i(\mathbf{w}_t)\}/|I_t|$, where I_t is the selected users at time t, and $\mathbb{I}\{A\}$ is the indicator function of an event A.

multiplicative adversarial manipulations and ensures fairness among all participating users. We established preliminary convergence guarantees for FedMGDA+, pointed out its connections to recent FL algorithms, and conducted extensive experiments to verify its effectiveness. In the future we plan to formally quantify the tradeoff induced by multiple local updates and to establish some privacy guarantee for FedMGDA+.

Acknowledgment

We thank the anonymous reviewers and the managing editor for their critical comments. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute. We thank NVIDIA Corporation (the data science grant) for donating two Titan V GPUs that enabled in part the computation in this work. We gratefully acknowledge funding support from NSERC, the Canada CIFAR AI Chairs Program, and Waterloo-Huawei Joint Innovation Lab.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," in *Proc. IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019, doi: 10.1145/3298981.

[4] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[5] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, vol. 97, pp. 4615–4625. [Online]. Available: http://proceedings.mlr.press/v97/mohri19a.html

[6] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=ByexElSYDr

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[8] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?," 2019, *arXiv:1911.07963*.

[9] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643. [Online]. Available: http://proceedings.mlr.press/v97/bhagoji19a.html

[10] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJxNAnVtDS

[11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450. [Online]. Available: https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf

[12] Y. Yu, "Better approximation and faster algorithm using the proximal average," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 458–466. [Online]. Available: https://proceedings.neurips.cc/paper/2013/hash/49182f81e6a13cf5eaa496d51fea6406-Abstract.html

[13] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1g2JnRcFX

[14] Z. Huo, Q. Yang, B. Gu, L. Carin, and H. Huang, "Faster on-device training using new federated momentum algorithm," 2020, *arXiv:2002.02090*.

[15] S. J. Reddi *et al.*, "Adaptive federated optimization,"*Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=LkFG3lB13U5

[16] R. Pathak and M. J. Wainwright, "FedSplit: An algorithmic framework for fast federated optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7057–7066. [Online]. Available: https://proceedings.neurips.cc//paper/2020/hash/4ebd440d99504722d80de606ea8507da-Abstract.html

[17] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BkluqlSFDS

[18] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7252–7261. [Online]. Available: http://proceedings.mlr.press/v97/yurochkin19a.html

[19] H. Wang *et al.*, "Attack of the tails: Yes, you really can backdoor federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 16070–16084. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/b8ffa41d4e492f0fad2f13e29e1762eb-Paper.pdf

[20] S. Caldas *et al.*, "LEAF: A benchmark for federated settings," 2019, *arXiv:1812.01097*.

[21] C. He *et al.*, "FedML: A research library and benchmark for federated machine learning," 2020, *arXiv:2007.13518*.

[22] H. H. Bauschke, R. Goebel, Y. Lucet, and X. Wang, "The proximal average: Basic theory," *SIAM J. Optim.*, vol. 19, no. 2, pp. 766–785, 2008. [Online]. Available: https://epubs.siam.org/doi/pdf/10.1137/070687542

[23] Y. Yu, X. Zheng, M. Marchetti-Bowick, and E. P. Xing, "Minimizing nonconvex non-separable functions," in *Proc. Artif. Intell. Statist.*, 2015, pp. 1107–1115. [Online]. Available: http://proceedings.mlr.press/v38/yu15.html

[24] J. Jahn, *Vector Optimization*. Berlin, Germany: Springer, 2009. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-17005-8

[25] H. Mukai, "Algorithms for multicriterion optimization," *IEEE Trans. Autom. Control*, vol. 25, no. 2, pp. 177–186, Apr. 1980. [Online]. Available: https://ieeexplore.ieee.org/document/1102298

[26] K. M. Miettinen, *Nonlinear Multiobjective Optimization*. Berlin, Germany: Springer, 1998. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4615-5563-6

[27] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Math. Methods Operations Res.*, vol. 51, no. 3, pp. 479–494, 2000. [Online]. Available: https://doi.org/10.1007/s001860000043

[28] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5, pp. 313–318, 2012. [Online]. Available: https://doi.org/10.1016/j.crma.2012.03.014

[29] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 525–536. [Online]. Available: https://papers.nips.cc/paper/7334-multi-task-learning-as-multi-objective-optimization.pdf

[30] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong, "Pareto multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12060–12070. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/685bfde03eb646c27ed565881917c71c-Paper.pdf

[31] I. Albuquerque, J. Monteiro, T. Doan, B. Considine, T. Falk, and I. Mitliagkas, "Multi-objective training of generative adversarial networks with multiple discriminators," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 202–211. [Online]. Available: http://proceedings.mlr.press/v97/albuquerque19a/albuquerque19a.pdf

[32] C. Xie, S. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation," in *Proc. Uncertainty Artif. Intell.*, 2019, pp. 261–270.

[33] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[34] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[35] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1596–1606.

[36] K. M. Anstreicher and L. A. Wolsey, "Two 'well-known' properties of subgradient optimization," *Math. Program.*, vol. 120, no. 1, pp. 213–220, 2009, doi: 10.1007/s10107-007-0148-y.

[37] J. Fliege, A. I. F. Vaz, and L. N. Vicente, "Complexity of gradient descent for multiobjective optimization," *Optim. Methods Softw.*, vol. 34, no. 5, pp. 949–959, 2019, doi: 10.1080/10556788.2018.1510928.
[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/kriz/cifar.html
[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1708.07747
[40] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: https://archive.ics.uci.edu/ml/index.php
[41] Q. Mercier, F. Poirion, and J.-A. Désidéri, "A stochastic multiple gradient descent algorithm," *Eur. J. Oper. Res.*, vol. 271, no. 3, pp. 808–817, 2018, doi: 10.1016/j.ejor.2018.05.064.

Zeou Hu is currently working toward the Ph.D. degree with David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada. His main research interests include multi-objective optimization and federated learning.

Kiarash Shaloudegi received the Ph.D. degree from Electrical and Electronic Engineering Department, Imperial College London, London, U.K. He is currently a Senior Applied Scientist with Amazon Advertising. Previously, he was a Senior Machine Learning Researcher with Huawei Noah's Ark Lab. His main research interests include machine learning, convex optimization, statistical learning, and distributed systems.

Guojun Zhang received the bachelor's degree in physics from the University of Science and Technology of China, Hefei, China, where he was awarded the Guomoruo Scholarship, master's degree in physics from Perimeter Institute, University of Waterloo, Waterloo, ON, Canada, and the Ph.D. degree from David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada. He is currently a Senior Researcher and the Tech Lead of federated learning with Huawei Noah Ark's Lab, Montréal, QC, Canada. He was a Student Affiliate of Vector Institute. He has authored or coauthored in several top ML journals or conferences, including ICML, NeurIPS, ICLR, and JMLR, and regularly is a reviewer for these conferences. His current research interests include federated learning and transfer learning. During his Ph.D., he was the recipient of the David R. Cheriton Scholarship.

Yaoliang Yu received the Ph.D. degree from Computing Science Department, University of Alberta, Edmonton, AB, Canada. He is currently an Associate Professor with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada, and a Faculty Member with Vector Institute. His main research interests include robust regression and classification, representation learning, kernel methods, convex and nonconvex optimization, distributed system, and applications in computer vision and natural language.