Federated Communication-Efficient Multi-Objective Optimization

Baris Askin **Pranay Sharma** **Gauri Joshi** **Carlee Joe-Wong**
Carnegie Mellon University

Abstract

We study a federated version of multi-objective optimization (MOO), where a single model is trained to optimize multiple objective functions. MOO has been extensively studied in the centralized setting but is less explored in federated or distributed settings. We propose `FedCMOO`, a novel communication-efficient federated multi-objective optimization (FMOO) algorithm that improves the error convergence performance of the model compared to existing approaches. Unlike prior works, the communication cost of `FedCMOO` does not scale with the number of objectives, as each client sends a single aggregated gradient to the central server. We provide a convergence analysis of the proposed method for smooth and non-convex objective functions under milder assumptions than in prior work. In addition, we introduce a variant of `FedCMOO` that allows users to specify a preference over the objectives in terms of a desired ratio of the final objective values. Through extensive experiments, we demonstrate the superiority of our proposed method over baseline approaches.

1 INTRODUCTION

Multi-objective optimization (MOO) has gained significant attention lately due to its role in the success of multi-task learning (MTL) (Caruana, 1997; Evgeniou and Pontil, 2004; Zhang and Yang, 2021), especially in practical applications such as computer vision (Zhang et al., 2014), natural language processing (Sanh et al., 2019), reinforcement learning (Sodhani et al., 2021), experimental design (Christensen et al., 2021), and power systems (Yin et al., 2021). In MTL, the data from multiple related tasks is used to simultaneously learn model(s) for the tasks. MOO is an instance of MTL where the goal is to train a single model $\mathbf{x}$ that simultaneously optimizes multiple objectives:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathbf{F}(\mathbf{x}) := [F_1(\mathbf{x}), F_2(\mathbf{x}), \dots, F_M(\mathbf{x})]^\top, \qquad (1)$$

where M is the number of objectives, $\mathbf{F} \in \mathbb{R}^M$ is the vector of the M individual objective loss functions $\{F_i\}_{i=1}^M$, and $\mathbf{x} \in \mathbb{R}^d$ is the common model that seeks to minimize them. In general, there need not exist a model $\mathbf{x}$ that simultaneously minimizes all the M losses. On the contrary, minimizing one loss often adversely affects performance on other tasks (Kurin et al., 2022). This phenomenon is commonly referred to as *negative transfer* (Mueller et al., 2024; Zhang et al., 2022). Therefore, MOO algorithms instead aim to find a *Pareto optimal/stationary* solution, where any further improvement in one loss necessarily worsens another (see definitions in Section 3) (Sener and Koltun, 2018). This results in a trade-off across the objectives, for example, balancing treatment efficacy and side effects in personalized medicine (Miller, 2024). Preference-based MOO (Mahapatra and Rajan, 2020) extends traditional MOO by incorporating the user preferences of relative task priorities into the model training. This enables users to tailor the solution based on their preferences, rather than seeking an arbitrary Pareto solution.

With the increasing proliferation of mobile and Internet-of-Things devices, an ever-increasing proportion of data for many applications, including many MOO problems, comes from edge devices. Federated learning (FL) McMahan et al. (2017); Kairouz et al. (2021) is a distributed learning paradigm that facilitates efficient collaborative learning of machine learning models at the edge devices, or *clients*. In FL, clients train the models using their local data. With the ever-growing size of machine learning models (Villalobos et al., 2022), communication with the central server is the main bottleneck. Therefore, the local models are only periodically aggregated at the server. Existing work addresses the multiple challenges faced

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

by FL algorithms, including fairness, privacy, and robustness to heterogeneity (Cui et al., 2021; Hu et al., 2022; Mehrabi et al., 2022). However, the proposed solutions usually focus on solving a single-objective optimization problem, leaving federated multi-objective optimization relatively underexplored.

This work focuses on the federated multi-objective optimization (FMOO) problem. In addition to the healthcare applications mentioned earlier, FMOO applications include recommendation systems (Sun et al., 2022), where relevance, product ratings, and personal preferences must be optimized simultaneously using private user data. In both of these applications, federated approaches are essential to protect the privacy of user preferences and health records. Compared to centralized MOO, FMOO faces additional challenges, including data heterogeneity across clients and excessive communication requirements, making learning more difficult. For example, the communication cost in (Yang et al., 2023), one of the few existing works on FMOO, scales linearly with the number of objectives, M. In our work, we address these challenges.

Contributions. We consider a multi-objective optimization problem in the federated setting. After reviewing some related work in Section 2 and preliminaries in Section 3, we make the following contributions:

- We propose `FedCMOO`[1], a communication-efficient federated algorithm to solve the MOO problem (1). Unlike existing work, the communication complexity of `FedCMOO` does not scale with the number of objectives M (see Section 4).
- We theoretically prove the convergence of `FedCMOO` for smooth non-convex objectives under milder assumptions than prior work (Theorem 1). Also, the sample complexity of `FedCMOO` has a better dependence on the number of objectives M. (Section 5).
- We propose `FedCMOO-Pref`, to our knowledge, the first federated algorithm to train a model that satisfies user-specified preferences, in terms of balancing the different objective values (Section 6).
- Through extensive experiments, we demonstrate the superior performance and efficiency of our proposed methods (Section 7).

2 RELATED WORK

Multi-objective Optimization. Existing approaches to solve multi-objective optimization problems include gradient-free methods, such as evolutionary methods (Deb et al., 2002; Vargas et al., 2015) and Bayesian optimization (Belakaria et al., 2020). However, gradient-based methods (Désidéri, 2012; Fliege et al., 2019; Liu and Vicente, 2024; Peitz and Dellnitz, 2018) are more suited to solving high-dimensional problems (Sener and Koltun, 2018) and form the focus of our work. Following the seminal work of Désidéri (2012), several approaches to MOO have been proposed in the literature. The simplest of these is *linear scalarization* (Hu et al., 2024; Lin et al., 2024b; Liu et al., 2021; Royer et al., 2024), where the M-dimensional vector of losses in (1) is replaced by a convex combination of the losses, resulting in a simpler scalar minimization. However, this approach suffers in the presence of *conflicting* gradients across tasks (Yu et al., 2020). In addition, Hu et al. (2024) show that linear scalarization fails to explore the entire Pareto front. Several works propose algorithms to address the problem of conflicting gradients in MOO in the centralized setting (Kurin et al., 2022; Royer et al., 2024; Xin et al., 2022).

Federated MOO and MTL. In our work, we consider the MOO problem in a federated setting where the goal is to train a model to simultaneously minimize multiple objective functions whose data is distributed across clients. Recently, some works have started exploring this setting in the decentralized (Blondin and Hale, 2021) and federated setting (Yang et al., 2023). However, their proposed algorithms incur high communication overhead, and the accompanying analyses make stronger assumptions than our work, as we explain in more detail in later sections. Another related, but distinct line of work is on multi-task learning for the purpose of personalization in federated learning (Smith et al., 2017; Chen et al., 2023b; Lu et al., 2024; Marfoq et al., 2021; Mills et al., 2021). These works still consider one task at all clients (e.g., an image classification problem) and aim to train a personalized model for each client, or train models to optimize different metrics (e.g., accuracy, fairness, or robustness against adversaries) (Cui et al., 2021; Mehrabi et al., 2022; Hu et al., 2022).

Preference-based MOO. Pareto optimal solutions with specific trade-offs between losses are often desirable. Some applications include recommender systems (Milojkovic et al., 2019), drug design (Jain et al., 2023), image classification (Raychaudhuri et al., 2022), and reinforcement learning (Basaklar et al., 2022). (Mahapatra and Rajan, 2020) proposed the first gradient-based multi-objective method to reach a preference-based solution, followed by more recent work in (Ruchte and Grabocka, 2021; Zhang et al., 2024). A related line of work aims to discover the entire Pareto set, rather than a single solution with a

[1]The code is provided at https://github.com/askinb/FedCMOO.

preference (Chen and Kwok, 2022; Dimitriadis et al., 2023; Haishan et al., 2024; Lin et al., 2022). In our work, we extend the preference-based algorithm of Mahapatra and Rajan (2020) to a federated setting.

3 PRELIMINARIES

Notations. Given a positive integer k, we define the set $[k] \triangleq \{1, \ldots, k\}$. ∇ and $\widetilde{\nabla}$ denote the gradient and stochastic gradient operators respectively. When used with a vector function, these operators return the Jacobian matrix, e.g., $\widetilde{\nabla}\mathbf{F}(\mathbf{x}) = [\widetilde{\nabla}F_1(\mathbf{x}), \ldots, \widetilde{\nabla}F_M(\mathbf{x})] \in \mathbb{R}^{d\times M}$. We use bold letters (e.g., $\mathbf{x}$) to denote vectors. $|S|$ is the cardinality of the set S. $\|\cdot\|$ denotes the Euclidean norm.

Pareto Optimal and Stationary Solutions. Unlike single-objective optimization, with MOO, which aims to solve the problem in 1, all the objectives generally cannot be optimized simultaneously. The inherent trade-offs between objectives lead to non-dominated solutions, where any further improvement in one objective necessarily implies making another objective worse. A solution $\mathbf{x}^*$ is called *Pareto optimal* if it is non-dominated, i.e., there exists no other $\mathbf{x} \in \mathbb{R}^d$ such that $F_k(\mathbf{x}) \le F_k(\mathbf{x}^*)$ for all $k \in [M]$ and $F_{k'}(\mathbf{x}) < F_{k'}(\mathbf{x}^*)$ for some $k' \in [M]$. The set of (potentially infinite) Pareto optimal solutions is called Pareto front, with each point representing a different trade-off among the M objectives. However, for non-convex problems, finding a Pareto optimal solution is NP-hard in general. The solution $\mathbf{x}^*$ is called *Pareto stationary* if there exists $\mathbf{w} \in \mathcal{S}_M$ such that $\sum_k w_k \nabla F_k = \mathbf{0}$, where $\mathcal{S}_M$ denotes the probability simplex, i.e., zero vector is in the convex hull of the gradients of objectives. At Pareto stationary points, there is no common descent direction for all objective functions. Pareto optimal solutions are also Pareto stationary.

Multi-gradient descent algorithm (MGDA). Among iterative gradient-based methods, the multi-gradient descent algorithm (`MGDA`) (Désidéri, 2012; Sener and Koltun, 2018; Xiao et al., 2023) guarantees convergence to *a Pareto stationary solution.* In `MGDA`, given some small learning rate η, the descent direction maximizes the minimum descent across tasks, i.e.,

$$\mathbf{d}^* = \max_{\mathbf{d}\in\mathbb{R}^d} \min_{k\in[M]} F_k(\mathbf{x}) - F_k(\mathbf{x} - \eta\mathbf{d}), \tag{2}$$

where $\mathbf{x}$ is the current model. Using first-order Taylor approximation, this optimization problem can equivalently be solved by finding the optimal weights for a convex combination of the objective gradients:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{S}_M} \|\textstyle\sum_k w_k \nabla F_k(\mathbf{x})\|, \tag{3}$$

where $\mathcal{S}_M$ is the probability simplex in $\mathbb{R}^M$. The descent vector is then $\mathbf{d}^* = \sum_k w_k^* \nabla F_k(\mathbf{x})$. Please refer to Appendix A for more details.

For large-scale applications, computing exact gradients is computationally expensive. Therefore, `Stochastic-MGDA` (Liu et al., 2021; Zhou et al., 2022a) uses stochastic gradients in (3). However, stochasticity also introduces additional theoretical challenges, and recent work focuses on proposing provably convergent stochastic MOO algorithms (Chen et al., 2023a; Fernando et al., 2022; Xiao et al., 2023).

Federated MOO. We consider the federated MOO problem (1), with N clients and a central server. For simplicity, we assume that each client has data for all tasks. However, our results easily extend to the more general case, where each client has data corresponding to only a subset of the M objectives. We denote the loss function corresponding to task k at client i as $f_{i,k}$, and the global loss function for task k is defined as:

$$F_k(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_{i,k}(\mathbf{x}), \quad \forall k \in [M]. \tag{4}$$

When solving (1) in FL, the server communicates the model $\mathbf{x}$ to the clients. On the other hand, clients retain their data and only communicate their gradients or accumulated local updates to the server.

Closest to our work, (Yang et al., 2023) introduces Federated Stochastic MGDA (`FSMGDA`). In one round of `FSMGDA`, each participating client performs local updates separately for all M objectives, and then sends the M individual updates to the server. The server averages these M updates across clients and uses them to compute the aggregation weights (3). As a result, `FSMGDA` faces two issues: 1) *High communication cost*, scaling linearly with the number of objectives M, since clients upload d-dimensional updates for each objective; and 2) *Significant local drift across objectives*, due to separate local trainings at clients, which deteriorates the performance of `FSMGDA` (see Section 7).

Next, we discuss our proposed algorithm `FedCMOO` (Federated Communication-efficient Multi-Objective Optimization) that addresses these shortcomings.

4 PROPOSED ALGORITHM

Our proposed `FedCMOO` (Algorithm 1) follows the same idea as `MGDA`. Once the server computes aggregation weights, the clients perform local training using the weighted sum of objectives and send updates to the server. The server then aggregates the client updates to compute the descent direction.

Algorithm 1 FedCMOO

```
1: Input: client and server learning rates $\eta_c, \eta_s$, number of local steps $\tau$
2: Initialize: global model $\mathbf{x}^{(0)} \in \mathbb{R}^d$, and task weights $\mathbf{w}^{(0)} \leftarrow [1/M, \dots, 1/M]^\top \in \mathcal{S}_M$
3: for $t = 0, 1, \dots, T-1$ do
4:   Select the client set $\mathcal{B}^{(t)}$ unif. randomly from $[N]$; send $\mathbf{x}^{(t)}$ to the clients in $\mathcal{B}^{(t)}$
5:   $G^{(t)} \leftarrow$ ApproxGramJacobian($\mathcal{B}^{(t)}, \mathbf{x}^{(t)}$) at the server, where $G^{(t)} \approx \nabla \mathbf{F}(\mathbf{x}^{(t)})^\top \nabla \mathbf{F}(\mathbf{x}^{(t)})$
6:   Compute $\mathbf{w}^{(t+1)} \leftarrow$ FindWeights($\mathbf{w}^{(t)}, G^{(t)}$) at server and send to clients in $\mathcal{B}^{(t)}$
7:   for each client $i \in \mathcal{B}^{(t)}$ in parallel do
8:     Initialize local model: $\mathbf{x}_i^{(t,0)} \leftarrow \mathbf{x}^{(t)}$
9:     for $r = 0, \dots, \tau - 1$ do
10:      $\mathbf{x}_i^{(t,r+1)} \leftarrow \mathbf{x}_i^{(t,r)} - \eta_c \sum_{k=1}^M w_k^{(t+1)} \widetilde{\nabla} f_{i,k}(\mathbf{x}_i^{(t,r)})$
11:    end for
12:    Send $\Delta_i^{(t)} \triangleq \frac{1}{\tau \eta_c}\left(\mathbf{x}^{(t)} - \mathbf{x}_i^{(t,\tau)}\right)$ to the server
13:  end for
14:  $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta_s \eta_c \tau \frac{1}{|\mathcal{B}^{(t)}|} \sum_{i \in \mathcal{B}^{(t)}} \Delta_i^{(t)}$
15: end for
16: Return $\mathbf{x}^{(T)}$
```

At the beginning of each round t, the server selects clients uniformly at random and sends the current model $\mathbf{x}^{(t)}$ to them (Algorithm 1, line 4). Next, the server approximates the Gram matrix of the Jacobian of loss vector $G^{(t)} \approx \nabla \mathbf{F}(\mathbf{x}^{(t)})^\top \nabla \mathbf{F}(\mathbf{x}^{(t)})$, with ApproxGramJacobian subroutine (line 5). For this approximation, each client in $\mathcal{B}^{(t)}$ needs to calculate a stochastic Jacobian matrix and send it to the server in a communication-efficient way, as described in Algorithm 3 in detail. The server uses $G^{(t)}$ to calculate the new aggregation weights $\mathbf{w}^{(t+1)}$ with the FindWeights subroutine (Algorithm 2) and sends these weights to the selected clients (line 6). Next, each participating client i performs τ local SGD steps on the weighted aggregate loss $\sum_{k=1}^M w_k^{(t+1)} f_{i,k}$ (lines 8-11). When finished, the clients send their updates $\{\Delta_i^{(t)}\}_{i \in \mathcal{B}^{(t)}}$ to the server, which aggregates these to arrive at the updated model $\mathbf{x}^{(t+1)}$, concluding one round of FedCMOO.

Next, we discuss the two subroutines used in FedCMOO, FindWeights and ApproxGramJacobian.

FindWeights. As the name suggests, FindWeights computes the aggregation weights $\{\mathbf{w}^{(t)}\}$ needed to compute the descent direction (3). The server solves the convex minimization in (3) using an iterative method, such as projected gradient descent (PGD). A single PGD update takes the form

$$\mathbf{w} \leftarrow \mathrm{proj}_{\mathcal{S}_M}\left(\mathbf{w} - \beta \nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x}) \mathbf{w}\right), \quad (5)$$

where β is the step size. The server can also utilize an approximation G of $\nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x})$ in (5), resulting in Algorithm 2, which runs K PGD steps. For $M \ll d$, the computational cost of FindWeights is negligible since $G \in \mathbb{R}^{M \times M}$ and $\mathbf{w} \in \mathbb{R}^M$.

Algorithm 2 FindWeights

```
1: Input: weights $\mathbf{w}$, Gram matrix $G$, step-size $\beta$, number of iterations $K$
2: $\mathbf{w} \leftarrow \mathrm{proj}_{\mathcal{S}_M}(\mathbf{w} - \beta G \mathbf{w})$ for $K$ iterations
3: Return $\mathbf{w}$
```

Next, we discuss the construction of $G \approx \nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x})$ at the server.

Randomized SVD-based Jacobian Approximation. Naively constructing $\nabla \mathbf{F}(\mathbf{x}) = [\nabla F_1(\mathbf{x}), \nabla F_2(\mathbf{x}), \dots, \nabla F_M(\mathbf{x})] \in \mathbb{R}^{d \times M}$ at the server would require each participating client i to calculate and communicate its gradients $\{\nabla f_{i,k}(\mathbf{x})\}_{k=1}^M$. Even constructing the stochastic version $\widetilde{\nabla} \mathbf{F}(\mathbf{x})$ with stochastic gradients would incur a communication cost of $\Theta(Md)$ per client, which scales linearly with the number of objectives M. In the ApproxGramJacobian subroutine, we propose a randomized SVD-based (Halko et al., 2011) approach to approximate the Gram matrix $\nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x})$, which reduces the per-client communication cost from $\Theta(Md)$ to $\Theta(d)$. We note that the proposed FedCMOO framework is compatible with any compressed communication method, which can be used as a substitute for the Randomized SVD-based ApproxGramJacobian. To effectively capture the correlation across task gradients, we specifically design ApproxGramJacobian and empirically validate its performance against several widely used compression techniques in Appendix B.

Algorithm 3 ApproxGramJacobian

```
1: Input: participating client set $\mathcal{B}$, model $\mathbf{x}$
2: for each client $i \in \mathcal{B}$ in parallel do
3:   $\widetilde{H}_i \triangleq \left[\widetilde{\nabla} f_{i,1}(\mathbf{x}), \dots, \widetilde{\nabla} f_{i,M}(\mathbf{x})\right] \in \mathbb{R}^{d \times M}$
4:   $\bar{H}_i \leftarrow$ Reshape($\widetilde{H}_i$) to $\mathbb{R}^{\sqrt{dM} \times \sqrt{dM}}$
5:   Send $\widehat{H}_i \leftarrow$ rand-SVD($\bar{H}_i, r$), the rank $r$ approximation of $\bar{H}_i$, to server
6:   $H_i \leftarrow$ Reshape $\widehat{H}_i$ back to $\mathbb{R}^{d \times M}$ at the server
7: end for
8: Return $G \leftarrow \left(\frac{1}{|\mathcal{B}|}\sum_i H_i\right)^\top \left(\frac{1}{|\mathcal{B}|}\sum_i H_i\right)$
```

We present the ApproxGramJacobian procedure in Algorithm 3. Each participating client i first reshapes its stochastic Jacobian $\widetilde{H}_i$ from a tall matrix $(d \times M)$ to a square matrix $(\sqrt{dM} \times \sqrt{dM})$, padding any missing entries with zeros (Algorithm 3, line 4). The reshaped matrices are then compressed using the leading

r components of randomized-SVD (Halko et al., 2011) and sent to the server. Without reshaping, the smallest component communicated to the server would be of size $d \times 1$. Reshaping enables us to achieve further compression. We adjust the number of singular components, r in randomized-SVD, so that the upload budget of each client is $d \times 1$, the same as the size of one model. While this compression technique requires additional computation, using randomized-SVD to compute a small number of components keeps the cost manageable. Also, stochastic gradients are reused in the first local iteration of clients, avoiding extra gradient computations due to Algorithm 3. At the server, the Gram matrix of the Jacobian is then approximated by the Gram matrix of the averaged compressed client Jacobians (line 8). Further details are provided in Appendix B.

`FedCMOO` addresses the two key issues with `FSMGDA` (Yang et al., 2023), 1) local training drift, and 2) high communication cost, by using the aggregated stochastic gradients $\sum_k w_k^{(t+1)} \widetilde{\nabla} f_{i,k}$ for local iterations at the clients (line 10 in Algorithm 1). This reduces the model drift compared to `FSMGDA`, which carries M independent local updates for the M objectives. Experimental results in Figure 2 (Section 7.2) and Figure 11 (Appendix E.4) also illustrate the effect of this phenomenon.

Proposition 1 (Communication cost comparison). *For the server to calculate the aggregation weights, the clients in `FSMGDA` (Yang et al., 2023) send M separate updates for the M objective functions, which results in an upload cost of $\Theta(Md)$ for each participating client per round. In contrast, the clients in `FedCMOO` upload a single averaged model update $\Delta_i^{(t)}$ (line 12 in Algorithm 1), reducing the communication cost to $\Theta(d)$. Computing the averaging weights in `FindWeights` incurs an additional communication cost of $\Theta(d)$ (Algorithm 3). Therefore, the per-client communication cost in every round of `FedCMOO` is $\Theta(d)$, significantly reducing the communication cost relative to `FSMGDA`.*

5 THEORETICAL ANALYSIS

For non-convex and smooth objective functions, we demonstrate the convergence of `FedCMOO` to a Pareto stationary solution using stochastic gradients.

Algorithmic Simplifications. For theoretical simplicity, we analyze `FedCMOO` with a slightly modified subroutine `ApproxGramJacobian` (see Appendix G.2).

- Clients communicate their stochastic Jacobians using an unbiased compression operator $\mathcal{Q}$ (see Assumption 5 below), rather than the more practical but biased randomized-SVD.
- To ensure that the output G of `ApproxGramJacobian` is an unbiased estimator of $\nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x})$, the server selects two additional sets $\mathcal{B}_1, \mathcal{B}_2$ of n' clients, each sampled uniformly at random. The resulting Gram matrix estimate is given by $\left(\frac{1}{n'} \sum_{i \in \mathcal{B}_1} \mathcal{Q}\left(\widetilde{H}_i\right)\right)^\top \left(\frac{1}{n'} \sum_{i \in \mathcal{B}_2} \mathcal{Q}\left(\widetilde{H}_i\right)\right)$.

In addition, in `FindWeights` subroutine, we use $K = 1$. The full procedure with all simplifications for theoretical analyses can be found in Algorithm 10 in Appendix G.2.

Even in the centralized setting, the theoretical convergence of MOO algorithms has only recently been explored Xiao et al. (2023); Fernando et al. (2022); Chen et al. (2023a). In our federated setting, additional challenges arise from local training at the clients, where client drift occurs due to heterogeneous data distribution, partial client participation, and communication cost considerations.

Assumptions. We provide the convergence analysis of `FedCMOO` under the following assumptions, which are standard in the FL and MOO literature (Jhunjhunwala et al., 2022; Xiao et al., 2023; Yang et al., 2023; Reisizadeh et al., 2020), and milder than those used in the existing literature.

Assumption 1 (Smoothness). *The loss functions are L-smooth, i.e. for all clients $i \in [N]$, tasks $k \in [M]$, and $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$, $\|\nabla f_{i,k}(\boldsymbol{x}) - \nabla f_{i,k}(\boldsymbol{x}')\| \leq L \|\boldsymbol{x} - \boldsymbol{x}'\|$.*

Assumption 2 (Bounded Variance). *Local stochastic gradients are unbiased and bounded-variance estimators of true local gradient, i.e., for all $\boldsymbol{x} \in \mathbb{R}^d$, $i \in [N]$, and $k \in [M]$, $\mathbb{E}[\widetilde{\nabla} f_{i,k}(\boldsymbol{x})] = \nabla f_{i,k}(\boldsymbol{x})$ and $\mathbb{E}||\widetilde{\nabla} f_{i,k}(\boldsymbol{x}) - \nabla f_{i,k}(\boldsymbol{x})||^2 \leq \sigma_l^2$.*

Assumption 3 (Bounded Heterogeneity). *Average distances of local gradients to the global gradient is bounded, i.e., for all $\boldsymbol{x} \in \mathbb{R}^d$, and $k \in [M]$, $\frac{1}{N} \sum_i ||\nabla f_{i,k}(\boldsymbol{x}) - F_k(\boldsymbol{x})||^2 \leq \sigma_g^2$.*

Assumption 4 (Bounded Gradients). *The local objective gradients have bounded norms, i.e, for all $\boldsymbol{x} \in \mathbb{R}^d$, $i \in [N]$, $k \in [M]$, $||\nabla f_{i,k}(\boldsymbol{x})|| \leq b$ for some $b > 0$.*

Assumption 5 (Unbiased Compression). *The random compression operator $\mathcal{Q}(\cdot)$ is unbiased and has a variance bound growing with the norm of the input, i.e, for all $\boldsymbol{x} \in \mathbb{R}^d$, $\mathbb{E}[\mathcal{Q}(\boldsymbol{x})] = \boldsymbol{x}$ and $\mathbb{E}||\mathcal{Q}(\boldsymbol{x}) - \boldsymbol{x}||^2 \leq q||\boldsymbol{x}||^2$, for some $q > 0$. This assumption is satisfied by several common quantizers (Reisizadeh et al., 2020).*

Theorem 1 (Convergence of `FedCMOO`). *Suppose Assumptions 1-5 hold, the client learning rate satisfies $\eta_c \leq \frac{1}{2L\tau}$, and the server selects $|\mathcal{B}^{(t)}| = n$ clients in* *every round. Then the iterates of `FedCMOO` satisfy,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\sum_k w_k^{(t)}\nabla F_k\left(\boldsymbol{x}^{(t)}\right)\right\|^2 \leq \underbrace{\mathcal{O}\left(\beta M C_1^2\right)}_{\text{MOO Weight Error}}$$
$$+ \underbrace{\mathcal{O}\left(\frac{1}{T\eta_s\eta_c\tau}+\frac{L\eta_s\eta_c\sigma_l^2}{n}\right)}_{\substack{\text{Centralized Optimization Error}\\ \text{for Scalarized Loss}}} + \underbrace{\mathcal{O}\left(L\eta_s\eta_c\tau b^2\right)}_{\substack{\text{Partial Participation}\\ \text{Error}}}$$
$$+ \underbrace{\mathcal{O}\left(L\eta_c(\tau b^2+\sqrt{\tau}b\sigma_l)\right)}_{\text{Local Drift Error}}, \tag{6}$$

where $C_1 \triangleq \mathcal{O}\left(\frac{q+1}{n'}\sigma_l^2+\frac{qb^2}{n'}+\frac{N-n'}{n'(N-1)}\sigma_g^2+b^2\right)$ *is a constant independent of T and M.*

Proof. See Appendix G.

The convergence bound of `FedCMOO` is decomposed according to the different sources of error. The step-size β in the `FindWeights` subroutine affects the error due to the changing MOO weights. If β is set to 0, the algorithm reduces to `FedAvg` (McMahan et al., 2017), with the scalar loss $\sum_k w_k^{(0)}F_k$. Setting $\beta=0$ in (6) also recovers the same dominant terms found in the FedAvg bound (Jhunjhunwala et al., 2022). However, in practice, dynamically adjusting the objective weights helps avoid gradient conflicts (see Table 2 in Section 7).

Corollary 1.1 (Convergence Rate)**.** *With the client and server learning rates $\eta_c=\frac{1}{L\tau\sqrt{\tau T}}$, $\eta_s=\sqrt{\tau}$, and step size of `FindWeights` $\beta=\frac{1}{M\sqrt{T}}$, the bound on $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\sum_k w_k^{(t)}\nabla F_k\left(\boldsymbol{x}^{(t)}\right)\right\|^2$ in (6) reduces to,*

$$\mathcal{O}\left(\frac{b^2+C_1^2}{\sqrt{T}}\right)+\mathcal{O}\left(\frac{\sigma_l^2}{\tau\sqrt{T}}\left(\frac{1}{n}+\frac{1}{\tau}\right)\right).$$

`FedCMOO` converges to a Pareto stationary solution at the rate $\mathcal{O}(1/\sqrt{T})$, which matches the best-known result for single-objective FL (Jhunjhunwala et al., 2022) under similar conditions.

The number of communication rounds needed to achieve $\mathbb{E}\|\sum_k w_k\nabla F_k\|^2 \leq \epsilon$ is $T=\mathcal{O}(1/\epsilon^2)$. The corresponding *sample complexity* of `FedCMOO`, which measures the total number of stochastic gradient computations required by the algorithm, is $T\tau\cdot Mn=\mathcal{O}(Mn/\epsilon^2)$, assuming $\tau=\mathcal{O}(1)$. The M factor follows since the clients' local updates in `FedCMOO` require computing stochastic gradients for all M objectives. The sample complexity of `FedCMOO` has a better dependence on the number of objectives, $\mathcal{O}(M)$, compared to the prior work. In contrast, the complexity of centralized MOO (Xiao et al., 2023, Theorem 3) grows as $\mathcal{O}(M^2)$, while that of `FSMGDA` (Yang et al., 2023, Theorem 5) grows as $\mathcal{O}(M^4)$.

Finally, we remove the impractical assumption in `FSMGDA` (Yang et al., 2023), which requires the variance of stochastic gradients to be bounded by $\mathcal{O}\left(\eta\sigma^2\right)$, where η is chosen as $\mathcal{O}(1/\sqrt{T})$ in (Yang et al., 2023, Assumption 4 and Corollary 6). We provide a detailed discussion of this assumption in Appendix F.

6 PREFERENCE-BASED FEDERATED MOO

`FedCMOO`, like most MOO algorithms in the literature, finds an *arbitrary* Pareto solution. However, users often prefer solutions with specific trade-offs among the different objectives. For example, in healthcare diagnostics, it is desirable to achieve similar detection accuracy across various ethnic groups (Yang and Ying, 2022). These preferences can be specified in terms of the ratios of the objective function values, as illustrated in the experiment in Figure 4. Li et al. (2024); Lin et al. (2019a, 2024b); Mahapatra and Rajan (2020) study preference-based MOO in the centralized setting. To our knowledge, ours is the first work to address this problem in a federated or distributed setting. Given the preference vector $\mathbf{r}\in\mathbb{R}_+^M$, we solve,

$$\begin{aligned}&\min_{\mathbf{x}\in\mathbb{R}^d}\mathbf{F}\left(\mathbf{x}\right):=\left[F_1(\mathbf{x}),F_2(\mathbf{x}),\ldots,F_M(\mathbf{x})\right]^\top\\&\text{subject to}\quad r_1F_1\left(\mathbf{x}\right)=\cdots=r_MF_M\left(\mathbf{x}\right).\end{aligned}\tag{7}$$

In other words, we want the normalized scaled losses $\hat{\mathbf{u}}(\mathbf{r})\triangleq\frac{\mathbf{r}\odot\mathbf{F}(\mathbf{x})}{\sum_k r_kF_k(\mathbf{x})}$ ($\odot$ denotes element-wise product) to satisfy $\hat{u}_i\approx 1/M$, for all $i\in[M]$. To solve (7), inspired by Mahapatra and Rajan (2020), we minimize the KL divergence of $\hat{\mathbf{u}}(\mathbf{r})$ with respect to the uniform distribution, $\mu_\mathbf{r}=\mathrm{KL}\left(\hat{\mathbf{u}}(\mathbf{r})\mid\mid\mathbf{1}/M\right)$. It is shown in Mahapatra and Rajan (2020) that using the descent direction $\nabla\mathbf{F}\left(\mathbf{x}\right)\mathbf{a}$, where $a_k=r_k\left(\log\left(\hat{u}_kM\right)-\mu_\mathbf{r}\right)$, guarantees a decrease in $\mu_\mathbf{r}$ at each iteration. The aggregation weight vector $\mathbf{w}$ solves the problem

$$\max_{\mathbf{w}\in\mathcal{S}_M\cap\mathcal{W}}\mathbf{w}^\top\nabla\mathbf{F}\left(\mathbf{x}\right)^\top\nabla\mathbf{F}\left(\mathbf{x}\right)\left(\mathbf{a}\mathbb{1}_\mu+\mathbf{1}(1-\mathbb{1}_\mu)\right),\tag{8}$$

where $\mathbb{1}_\mu$ is an indicator function, which takes the value 1 if the KL divergence μ is greater than a certain threshold. $\mathcal{S}_M$ is the probability simplex, while $\mathcal{W}$ is the set of additional practical constraints (see Appendix C for more details). The solution to (8) either reduces the KL divergence $\mu_\mathbf{r}$ or maximizes the total decrease across all objective values.

Our proposed preference-based method `FedCMOO-Pref` (see Appendix C) is similar to `FedCMOO`, with the difference that the aggregation weights are computed using `FindWeights-Pref` (Algorithm 4) instead of `FindWeights`. The per-client communication cost for one round of `FedCMOO-Pref`, like `FedCMOO`, is $\Theta(d)$.

Algorithm 4 `FindWeights-Pref`

```
1: Input: preferences $\mathbf{r}$, losses $\mathbf{F}$, Gram matrix $G$
2: Compute normalized losses $\hat{\mathbf{u}}(\mathbf{r}) \triangleq \frac{\mathbf{r} \odot \mathbf{F}(\mathbf{x})}{\sum_k r_k F_k(\mathbf{x})}$
3: Compute non-uniformity: $\mu_{\mathbf{r}} = \mathrm{KL}\left(\hat{\mathbf{u}}(\mathbf{r}) \;||\; \frac{\mathbf{1}}{M}\right)$
4: Compute $a_k = r_k \left(\log\left(\hat{u}_k M\right) - \mu_{\mathbf{r}}\right)$, $\forall\, k \in [M]$
5: Return $\arg\max_{\mathbf{w} \in \mathcal{S}_M \cap \mathcal{W}} \mathbf{w}^\top G \left(\mathbf{a}\mathbb{1}_\mu + \mathbf{1}(1 - \mathbb{1}_\mu)\right)$
```

7 EXPERIMENTAL RESULTS

We first describe our experimental setup (Section 7.1), followed by our results and insights (Section 7.2).

7.1 Datasets and Implementation

We consider the following experimental settings:

1. **MultiMNIST**: Using the MNIST dataset (Deng, 2012), we create a two-objective dataset by randomly merging images. The image size is preserved, with two randomly sampled digits (one in the top left and one in the bottom right) in each sample.
2. **MNIST+FMNIST**: Similar to the construction of MultiMNIST, we construct another dataset by combining two randomly sampled images, one from the MNIST dataset and the other from the FashionMNIST dataset Xiao et al. (2017).
3. **CIFAR10+FMNIST**: We use three-channel images from the CIFAR-10 dataset (Krizhevsky et al., 2009) and place the same randomly selected MNIST digit at a fixed position in all the channels.
4. **CelebA** and **CelebA-5**: CelebA is a large-scale face dataset, with each sample containing 40 binary attributes, resulting in a problem with 40 objectives. In addition, we construct the CelebA-5 dataset with 5 objectives, by partitioning the attributes of CelebA into groups of 8.
5. **QM9**: The QM9 dataset (Ramakrishnan et al., 2014) consists of graph representations of molecules, each associated with 11 continuous-valued molecular properties for regression.

We use CNN-type models with varying sizes for all experiments, except for QM9, where a graph neural network is used. The smallest model has 34.6k parameters, while the largest model, ResNet-18, has 11.2M parameters. All architectures consist of a shared encoder, used by all the objective functions, and objective-specific decoder components, which form the final one or two linear layers and are applied separately for each objective. See Appendix D for more details.

Each experiment, except QM9, has $N = 100$ clients, with 10 selected in each round. Since the MNIST, FashionMNIST, and CIFAR-10 datasets have 10 classes each, each sample in the constructed datasets MultiMNIST, MNIST+FMNIST, and CIFAR10+FMNIST belongs to one of the 100 *composite* classes. For the Celeb-A datasets, we select the three most balanced attributes in terms of the 0/1 ratio and generate 8-class labels using these three binary labels. We follow the Dirichlet distribution with $\alpha = 0.3$, as in Acar et al. (2021), to partition the data samples across clients. For QM9 experiments, we use a total of 20 clients, with 4 active in each round. We split the dataset uniformly at random across the clients. We run all experiments with multiple random seeds on NVIDIA H100 GPUs and present the average results.

We compare the performance of the discussed methods in extensive experiments. We compare `FedCMOO` (Algorithm 1), the preference-based `FedCMOO-Pref` (Section 6), and prior work `FSMGDA` (Yang et al., 2023). In some experiments, we also examine `Scalarized`, which minimizes the average of individual objective losses (Xiao et al., 2023; Lin et al., 2024c).

7.2 Results and Insights

Test Accuracy Curves. We plot the mean test accuracy of the objectives trained together over global rounds in Figure 1. For `FedCMOO-Pref`, we assign equal importance to all objectives. We observe that our proposed `FedCMOO` achieves faster training. Since `FSMGDA` performs local training separately for each objective and fuses the gradient information of all objectives only at the end of each round, its performance lags behind due to local model drift. This drift is more pronounced as the number of objectives increases; for example, `FSMGDA` performs much worse when training on the CelebA dataset with 40 objectives. We also provide the final test accuracy for each objective in the two-objective experiments in Table 1. As the gap between the difficulty levels of the two objectives increases, `FedCMOO-Pref` performs better on the more challenging objective. For instance, `FedCMOO-Pref` performs best on the CIFAR-10 task of CIFAR10+FMNIST. However, `FedCMOO-Pref` requires a higher number of global rounds to train compared to `FedCMOO`, as it strives to keep the objective values balanced. Please refer to Appendix E.1 for additional training curves from all experiments, Appendix E.2 for radial plots of individual tasks' final test losses in CelebA and CelebA-5, and Appendix E.3 for Δ_M results, a commonly used metric in MOO literature (Fernando et al., 2022) that measures the average percentage performance loss due to MOO.

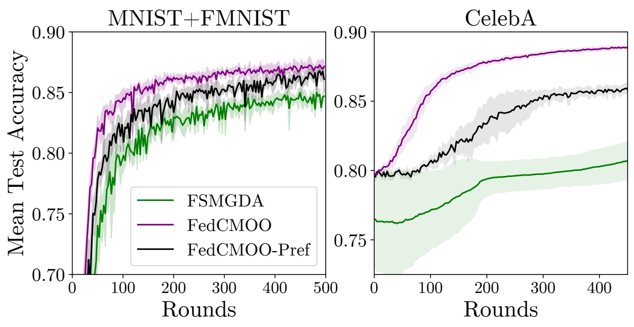

Figure 1: Mean test accuracy with MNIST+FMNIST and CelebA datasets. FedCMOO outperforms FSMGDA both in training speed and final accuracy.

Table 1: The final test accuracy (%) of the first/second objectives in 2-objective settings. The bold values indicate the best accuracy for each objective.

Experiment	MNIST+ FMNIST	Multi MNIST	CIFAR10+ MNIST
FSMGDA	93.0/75.4	92.3/88.2	58.2/96.2
FedCMOO	**95.5**/78.8	**94.4**/**92.6**	57.4/**96.8**
FedCMOO-Pref	94.0/**79.2**	93.8/91.1	**63.6**/89.1

Local drift of FSMGDA. We design an experiment to further validate our claim that FSMGDA suffers from local objective drift. In Figure 2, we compare the mean test accuracy levels achieved by FedCMOO and FSMGDA on the MultiMNIST and MNIST+FMNIST datasets with varying numbers of local training iterations. We observe that the drift caused by the separate training of each local objective in FSMGDA results in worse accuracy. We also compare the per-round local training progress made by FedCMOO and FSMGDA (see the result in Appendix E.4). The drift across objectives in FSMGDA hinders the performance of local training.

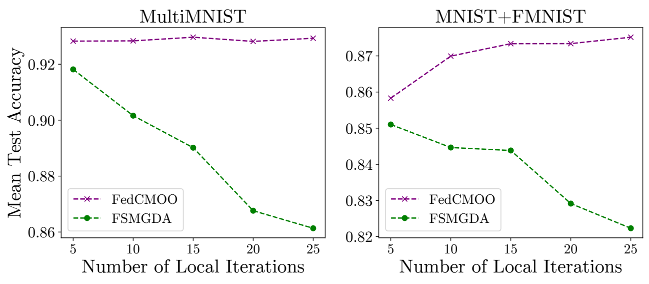

Figure 2: Mean test accuracy after 500 global rounds. As discussed earlier, with an increasing number of local iterations, the local objective drift of FSMGDA leads to worse performance compared to FedCMOO.

The communication efficiency of proposed methods. We present the average test loss curves across 11 regression tasks of the QM9 dataset in Figure 3. The results confirm the superiority of FedCMOO over the baseline FSMGDA in terms of training speed relative to the number of server rounds. To further assess the communication efficiency of the proposed methods, we also plot the same results with respect to the amount of uploaded data. The right plot in Figure 3 highlights the significant advantage of our methods, as their communication cost does not scale with the number of objectives, unlike the baseline. Moreover, we present a radial plot of the final test accuracies for each method in Figure 12 in Appendix E.5, further demonstrating the superiority of FedCMOO across individual tasks.

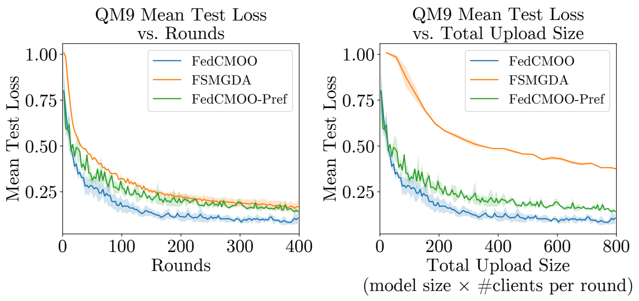

Figure 3: Mean test loss curves of FedCMOO, FedCMOO-Pref (uniform preference), and FSMGDA across 11 QM9 tasks. Left: Test loss vs. rounds. Right: Test loss vs. uploaded data (*model size* × *number of clients per round*). FedCMOO outperforms FSMGDA due to reduced local drift, while both FedCMOO and FedCMOO-Pref achieve superior communication efficiency.

Finding Pareto solutions with user preferences. We test the ability of our proposed FedCMOO-Pref to find Pareto solutions with user-specified preferences. In Figure 4, we present the final objective values achieved by FedCMOO-Pref with varying preferences across objectives, as indicated in plot legends, showing FedCMOO-Pref's success in identifying solutions that align with user-defined preferences. The vertical and horizontal dashed lines indicate the minimum loss values from single-objective training, meaning the gray regions are infeasible. When the difficulty gap between the objectives is smaller, e.g., MultiMNIST, the loss values of the trained models follow the specified preferences. However, preferences with significantly different weights for the two objectives are not fully satisfied, and the performance is closer to the gray region. On the other hand, in MNIST+FMNIST, where the FashionMNIST objective is inherently harder than the MNIST objective, the resulting model reflects the preference trend with a smaller value of the easy MNIST objective. The federated setting further exacerbates the difficulty of alignment with the preference vector.

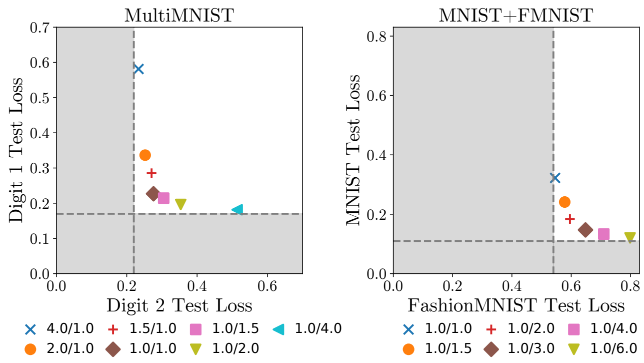

Figure 4: `FedCMOO-Pref` effectively finds solutions that align well with user-preferences in most cases. Imbalanced preferences or differences in the difficulty levels of objectives may cause misalignment.

The drawback of using scalarized loss. We further compare the performance of MOO-specific algorithms, i.e., `FedCMOO` and `FSMGDA`, with the single-objective vanilla FL algorithm, FedAvg (McMahan et al., 2017). We scalarize all objectives into a single objective by averaging them, and refer to this approach as `Scalarized`. Although linear scalarization methods can find a Pareto stationary point, they do not have the practical benefits of descent direction maximizing the minimum descent Zhou et al. (2022b). We observe that its performance is highly influenced by loss types and scales across objectives. In Table 2, we change the Digit 1 objective of MultiMNIST and the MNIST objective of MNIST+FMNIST to ℓ_2-norm loss while using negative log-likelihood (NLL) loss for the other objectives. We find that `Scalarized` focuses primarily on training the first objective, resulting in poor performance on the other objective. Although MOO-specific `FSMGDA` performs better than `Scalarized`, as discussed above, it suffers from local drift. `FedCMOO` outperforms both `Scalarized` and `FSMGDA`.

Table 2: Final test accuracy (%) comparison showing poor performance of `Scalarized` when different loss types (ℓ_2 and NLL) are used across objectives. MOO-specific algorithms maintain more balanced results.

Methods	MultiMNIST		CIFAR10+FMNIST	
	Digit 1	Digit 2	CIFAR-10	MNIST
`Scalarized`	10.5	93.4	20.7	97.6
`FSMGDA`	94.6	17.7	64.2	61.0
`FedCMOO`	95.0	70.3	66.3	80.0

8 CONCLUDING REMARKS

In this work, we study the federated multi-objective optimization (FMOO) problem. We propose `FedCMOO`, a novel algorithm that achieves significant communication savings compared to existing methods. Our convergence analysis holds under milder assumptions than those made in prior work and shows that the complexity of `FedCMOO` achieves better dependence on the number of objectives than existing methods. We also propose a variant of `FedCMOO` that allows users to enforce preferences towards specific solutions on the Pareto front, to achieve different trade-offs among the objective values. Our experiments show the efficacy of our proposed methods. Potential future directions include exploring the effect of task characteristics on training in FMOO and investigating memory- and communication-efficient variance reduction techniques for FMOO settings.

Acknowledgements

This work was partially supported by the US National Science Foundation under grants CNS-2106891 and CNS-2409138 to CJW and CCF 2045694, CNS-2112471, CPS-2111751, and ONR N00014-23-1-2149 to GJ.

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. (2021). Federated learning based on dynamic regularization. In *International Conference on Learning Representations.*

Askin, B., Sharma, P., Joe-Wong, C., and Joshi, G. (2024). FedAST: Federated asynchronous simultaneous training. In *The 40th Conference on Uncertainty in Artificial Intelligence.*

Basaklar, T., Gumussoy, S., and Ogras, U. (2022). Pd-morl: Preference-driven multi-objective reinforcement learning algorithm. In *The Eleventh International Conference on Learning Representations.*

Belakaria, S., Deshwal, A., Jayakodi, N. K., and Doppa, J. R. (2020). Uncertainty-aware search framework for multi-objective bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10044–10052.

Blondin, M. J. and Hale, M. (2021). A decentralized multi-objective optimization algorithm. *Journal of Optimization Theory and Applications*, 189(2):458–485.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28:41–75.

Chen, L., Fernando, H., Ying, Y., and Chen, T. (2023a). Three-way trade-off in multi-objective learning: Optimization, generalization and conflict-avoidance. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 70045–70093. Curran Associates, Inc.

Chen, W. and Kwok, J. (2022). Multi-objective deep learning with adaptive reference vectors. *Advances in Neural Information Processing Systems*, 35:32723–32735.

Chen, Y., Zhang, T., Jiang, X., Chen, Q., Gao, C., and Huang, W. (2023b). Fedbone: Towards large-scale federated multi-task learning. *arXiv preprint arXiv:2306.17465*.

Christensen, M., Yunker, L. P., Adedeji, F., Häse, F., Roch, L. M., Gensch, T., dos Passos Gomes, G., Zepel, T., Sigman, M. S., Aspuru-Guzik, A., et al. (2021). Data-science driven autonomous process optimization. *Communications Chemistry*, 4(1):112.

Crawshaw, M. (2020). Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.

Cui, S., Pan, W., Liang, J., Zhang, C., and Wang, F. (2021). Addressing algorithmic disparity and performance inconsistency in federated learning. *Advances in Neural Information Processing Systems*, 34:26091–26102.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dimitriadis, N., Frossard, P., and Fleuret, F. (2023). Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models. In *International Conference on Machine Learning*, pages 8015–8052. PMLR.

Désidéri, J.-A. (2012). Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5):313–318.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117.

Fernando, H., Shen, H., Liu, M., Chaudhury, S., Murugesan, K., and Chen, T. (2022). Mitigating gradient bias in multi-objective learning: A provably convergent stochastic approach. *arXiv preprint arXiv:2210.12624*.

Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. (2021). Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516.

Fliege, J., Vaz, A. I. F., and Vicente, L. N. (2019). Complexity of gradient descent for multiobjective optimization. *Optimization Methods and Software*, 34(5):949–959.

Haishan, Z., Das, D., and Tsuda, K. (2024). Preference-optimized pareto set learning for blackbox optimization. *arXiv preprint arXiv:2408.09976*.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hu, Y., Xian, R., Wu, Q., Fan, Q., Yin, L., and Zhao, H. (2024). Revisiting scalarization in multi-task learning: A theoretical perspective. *Advances in Neural Information Processing Systems*, 36.

Hu, Z., Shaloudegi, K., Zhang, G., and Yu, Y. (2022). Federated learning meets multi-objective optimization. *IEEE Transactions on Network Science and Engineering*, 9(4):2039–2051.

Jain, M., Raparthy, S. C., Hernández-Garcıa, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. (2023). Multi-objective gflownets. In *International conference on machine learning*, pages 14631–14653. PMLR.

Jhunjhunwala, D., SHARMA, P., Nagarkatti, A., and Joshi, G. (2022). FedVARP: Tackling the variance due to partial client participation in federated learning. In *The 38th Conference on Uncertainty in Artificial Intelligence*.

Jhunjhunwala, D., Wang, S., and Joshi, G. (2023). Fedexp: Speeding up federated averaging via extrapolation. In *The Eleventh International Conference on Learning Representations*.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210.

Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kurin, V., De Palma, A., Kostrikov, I., Whiteson, S., and Mudigonda, P. K. (2022). In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183.

Langner, S., Häse, F., Perea, J. D., Stubhan, T., Hauch, J., Roch, L. M., Heumueller, T., Aspuru-Guzik, A., and Brabec, C. J. (2020). Beyond ternary opv: high-throughput experimentation and self-driving laboratories optimize multicomponent systems. *Advanced Materials*, 32(14):1907801.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, Z., Li, T., Smith, V., Bilmes, J., and Zhou, T. (2024). Many-Objective Multi-Solution transport. *arXiv preprint arXiv:2403.04099*.

Lin, B. and Zhang, Y. (2023). LibMTL: A Python library for multi-task learning. *Journal of Machine Learning Research*, 24(209):1–7.

Lin, X., Liu, Y., Zhang, X., Liu, F., Wang, Z., and Zhang, Q. (2024a). Few for many: Tchebycheff set scalarization for many-objective optimization. *arXiv preprint arXiv:2405.19650*.

Lin, X., Yang, Z., Zhang, X., and Zhang, Q. (2022). Pareto set learning for expensive multi-objective optimization. *Advances in neural information processing systems*, 35:19231–19247.

Lin, X., Zhang, X., Yang, Z., Liu, F., Wang, Z., and Zhang, Q. (2024b). Smooth Tchebycheff Scalarization for Multi-Objective Optimization. arXiv:2402.19078 [cs, math].

Lin, X., Zhang, X., Yang, Z., Liu, F., Wang, Z., and Zhang, Q. (2024c). Smooth tchebycheff scalarization for multi-objective optimization. *arXiv preprint arXiv:2402.19078*.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q., and Kwong, S. (2019a). Pareto Multi-Task Learning. arXiv:1912.12854 [cs, stat].

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. (2019b). Pareto multi-task learning. *Advances in neural information processing systems*, 32.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. (2021). Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Liu, S. and Vicente, L. N. (2024). The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *Annals of Operations Research*, 339(3):1119–1148.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Lu, Y., Huang, S., Yang, Y., Sirejiding, S., Ding, Y., and Lu, H. (2024). Fedhca2: Towards hetero-client federated multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5599–5609.

MacLeod, B. P., Parlane, F. G., Morrissey, T. D., Häse, F., Roch, L. M., Dettelbach, K. E., Moreira, R., Yunker, L. P., Rooney, M. B., Deeth, J. R., et al. (2020). Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867.

Mahapatra, D. and Rajan, V. (2020). Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto ptimization. In *International Conference on Machine Learning PMLR*.

Marfoq, O., Neglia, G., Bellet, A., Kameni, L., and Vidal, R. (2021). Federated multi-task learning under a mixture of distributions. *Advances in Neural Information Processing Systems*, 34:15434–15447.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

McMahan, B. and Ramage, D. (2017). Federated learning: Collaborative machine learning without centralized training data. https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/. Accessed: 2024-10-12.

Mehrabi, N., de Lichy, C., McKay, J., He, C., and Campbell, W. (2022). Towards multi-objective statistically fair federated learning. *arXiv preprint arXiv:2201.09917*.

Mercier, Q., Poirion, F., and Désidéri, J.-A. (2018). A stochastic multiple gradient descent algorithm. *European Journal of Operational Research*, 271(3):808–817.

Miller, D. S. (2024). Machine learning models for predicting drug efficacy and side effects: Developing machine learning models to predict drug efficacy and potential side effects based on patient characteristics and pharmacogenomic data, aiding in treatment selection and pers. 4:25–32.

Mills, J., Hu, J., and Min, G. (2021). Multi-task federated learning for personalised deep neural networks in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 33(3):630–641.

Milojkovic, N., Antognini, D., Bergamin, G., Faltings, B., and Musat, C. (2019). Multi-gradient descent for multi-objective recommender systems. *arXiv preprint arXiv:2001.00846*.

Momma, M., Dong, C., and Liu, J. (2022). A multi-objective/multi-task learning framework induced by pareto stationarity. In *International Conference on Machine Learning*, pages 15895–15907. PMLR.

Mueller, D., Dredze, M., and Andrews, N. (2024). Can optimization trajectories explain multi-task transfer? *arXiv preprint arXiv:2408.14677*.

Peitz, S. and Dellnitz, M. (2018). Gradient-based multiobjective optimization with uncertainties. In *NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico*, pages 159–182. Springer.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7.

Raychaudhuri, D. S., Suh, Y., Schulter, S., Yu, X., Faraki, M., Roy-Chowdhury, A. K., and Chandraker, M. (2022). Controllable dynamic multi-task architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10955–10964.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. (2020). Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International conference on artificial intelligence and statistics*, pages 2021–2031. PMLR.

Royer, A., Blankevoort, T., and Ehteshami Bejnordi, B. (2024). Scalarization for multi-task and multi-domain learning at scale. *Advances in Neural Information Processing Systems*, 36.

Ruchte, M. and Grabocka, J. (2021). Scalable pareto front approximation for deep multi-objective learning. In *2021 IEEE international conference on data mining (ICDM)*, pages 1306–1311. IEEE.

Sanh, V., Wolf, T., and Ruder, S. (2019). A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6949–6956.

Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. *Advances in neural information processing systems*, 30.

Sodhani, S., Zhang, A., and Pineau, J. (2021). Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pages 9767–9779. PMLR.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. (2020). Which tasks should be learned together in multi-task learning? In *International conference on machine learning*, pages 9120–9132. PMLR.

Sun, Z., Xu, Y., Liu, Y., He, W., Kong, L., Wu, F., Jiang, Y., and Cui, L. (2022). A survey on federated recommendation systems. *arXiv preprint arXiv:2301.00767*.

Vargas, D. V., Murata, J., Takano, H., and Delbem, A. C. B. (2015). General subpopulation framework and taming the conflict inside populations. *Evolutionary computation*, 23(1):1–36.

Villalobos, P., Sevilla, J., Besiroglu, T., Heim, L., Ho, A., and Hobbhahn, M. (2022). Machine learning model sizes and the parameter gap. *arXiv preprint arXiv:2207.02852*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xiao, P., Ban, H., and Ji, K. (2023). Direction-oriented multi-objective learning: Simple and provable stochastic algorithms. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 4509–4533. Curran Associates, Inc.

Xin, D., Ghorbani, B., Gilmer, J., Garg, A., and Firat, O. (2022). Do current multi-task optimization methods in deep learning even help? *Advances in neural information processing systems*, 35:13597–13609.

Yang, H., Liu, Z., Liu, J., Dong, C., and Momma, M. (2023). Federated multi-objective learning. *Advances in Neural Information Processing Systems*, 36.

Yang, T. and Ying, Y. (2022). Auc maximization in the era of big data and ai: A survey. *ACM Computing Surveys*, 55(8):1–37.

Yin, L., Wang, T., and Zheng, B. (2021). Analytical adaptive distributed multi-objective optimization algorithm for optimal power flow problems. *Energy*, 216:119245.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020). Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Zhang, W., Deng, L., Zhang, L., and Wu, D. (2022). A survey on negative transfer. *IEEE/CAA Journal of Automatica Sinica*, 10(2):305–329.

Zhang, X., Lin, X., and Zhang, Q. (2024). Pmgda: A preference-based multiple gradient descent algorithm. *arXiv preprint arXiv:2402.09492*.

Zhang, Y. and Yang, Q. (2021). A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). Facial landmark detection by deep multi-task learning. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 94–108. Springer.

Zhou, S., Zhang, W., Jiang, J., Zhong, W., Gu, J., and Zhu, W. (2022a). On the convergence of stochastic multi-objective gradient manipulation and beyond. *Advances in Neural Information Processing Systems*, 35:38103–38115.

Zhou, S., Zhang, W., Jiang, J., Zhong, W., GU, J., and Zhu, W. (2022b). On the convergence of stochastic multi-objective gradient manipulation and beyond. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Zhu, W., Luo, J., and White, A. D. (2022). Federated learning of molecular properties with graph neural networks in a heterogeneous setting. *Patterns*, 3(6).

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes, please see Section 4 and Section 5.]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes, please see Section 5.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes, please see Supplementary Materials.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes, please see Section 5.]
   (b) Complete proofs of all theoretical results. [Yes, please see Appendix G.]
   (c) Clear explanations of any assumptions. [Yes, please see Section 5.]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, please see https://github.com/askinb/FedCMOO.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes, please see Appendix D.]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes, please see Section 7 and Appendix D.]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, the experiments are run on our internal clusters with NVIDIA H100 Tensor Core GPUs as indicated in Section 7.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes, please see Section 7 and Appendix D.]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix for "Federated Communication-Efficient Multi-Objective Optimization"

A MULTI-OBJECTIVE OPTIMIZATION AND MGDA

The multi-gradient descent algorithm (`MGDA`), a seminal work in multi-objective optimization (MOO) by Désidéri (2012), proposes an iterative gradient-based method to find a Pareto stationary solution with a convergence guarantee. In every iteration, `MGDA` aims to find a descent direction of model $\mathbf{x}$ that maximizes the minimum descent across the tasks by forming the problem as

$$\max_{\mathbf{d}\in\mathbb{R}^d} \min_{k\in[M]} \left\{ \frac{1}{\eta} \left(F_k\left(\mathbf{x}\right) - F_k\left(\mathbf{x}-\eta\mathbf{d}\right) \right) \right\}. \tag{9}$$

Using the first-order Taylor expansion, we can approximate $\frac{1}{\eta}\left(F_k\left(\mathbf{x}\right) - F_k\left(\mathbf{x}-\eta\mathbf{d}\right)\right) \approx \mathbf{d}^\top \nabla F_k\left(\mathbf{x}\right)$. Also, by regularizing the norm square of $\mathbf{d}$, the problem is to compute

$$\max_{\mathbf{d}\in\mathbb{R}^d} \min_{k\in[M]} \left\{ \mathbf{d}^\top \nabla F_k\left(\mathbf{x}\right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2 \right\}. \tag{10}$$

Noticing that $\min_{k\in[M]} \mathbf{d}^\top \nabla F_k\left(\mathbf{x}\right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2 = \min_{\mathbf{w}\in\mathcal{S}_M} \mathbf{d}^\top \left(\sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right)\right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2$ where $\mathcal{S}_M$ is M-probability simplex, Equation (10) is equivalent to

$$\max_{\mathbf{d}\in\mathbb{R}^d} \min_{\mathbf{w}\in\mathcal{S}_M} \left\{ \mathbf{d}^\top \left(\sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right) \right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2 \right\}. \tag{11}$$

Since the problem is concave in $\mathbf{d}$ and affine in $\mathbf{w}$, we can switch the order of min and max in Equation (11) to obtain

$$\min_{\mathbf{w}\in\mathcal{S}_M} \max_{\mathbf{d}\in\mathbb{R}^d} \left\{ \mathbf{d}^\top \left(\sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right) \right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2 \right\}.$$

Now, the solution to $\max_{\mathbf{d}\in\mathbb{R}^d} \left\{ \mathbf{d}^\top \left(\sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right)\right) - \frac{1}{2}\left\|\mathbf{d}\right\|^2 \right\}$ can be easily found as $\mathbf{d}^{*\max} = \sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right)$. Inserting this into the problem, the problem reduces to find $\mathbf{w}$. We obtain `MGDA` problem in Equation (3) in the main text:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{S}_M} \left\| \sum_{k\in[M]} w_k \nabla F_k\left(\mathbf{x}\right) \right\|, \tag{12}$$

where $\mathcal{S}_M$ is M-probability simplex. In every iteration of `MGDA` (Désidéri, 2012), the gradients of all objective functions are calculated and then used to find aggregation weights. Then the descent direction is determined as the weighted average of gradients. `MGDA` is depicted in Algorithm 5.

Algorithm 5 The multi-gradient descent algorithm (`MGDA`) by Désidéri (2012)

```
1: Input: Learning rate $\eta$
2: Initialization: $\mathbf{x}^{(0)} \in \mathbb{R}^d$
3: for $t = 0, 1, \ldots, T-1$ do
4:     Calculate gradients for all objectives, $\nabla F_1\left(\mathbf{x}^{(t)}\right), \nabla F_2\left(\mathbf{x}^{(t)}\right), \ldots, \nabla F_M\left(\mathbf{x}^{(t)}\right)$
5:     $\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w} \in \mathcal{S}_M} \left\| \sum_{k \in [M]} w_k \nabla F_k\left(\mathbf{x}^{(t)}\right) \right\|$
6:     $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta \sum_{k \in [M]} w_k^{(t+1)} \nabla F_k\left(\mathbf{x}^{(t)}\right)$
7: end for
8: Return $\mathbf{x}^{(T)}$
```

Calculating exact gradients is often not feasible for large-scale applications. Therefore, in practice, stochastic gradients are used in place of exact gradients in Algorithm 5 (Mercier et al., 2018). However, introducing stochasticity presents additional theoretical challenges, as using the same random gradients for both weight calculation (Equation (12)) and descent introduces bias. Fernando et al. (2022) demonstrate the non-convergence of `MGDA` when using stochastic gradients. Recent works have focused on developing provably convergent stochastic MOO algorithms (Chen et al., 2023a; Fernando et al., 2022; Xiao et al., 2023).

B ApproxGramJacobian SUBROUTINE

The `ApproxGramJacobian` algorithm is used to approximate the Gram matrix of the Jacobian of the objective loss vector, $\nabla \mathbf{F}(\mathbf{x})^\top \nabla \mathbf{F}(\mathbf{x})$, in a communication-efficient manner in our proposed `FedCMOO` method. Since only stochastic gradients are computed by the participating clients in each round in the FL setting (line 3 in Algorithm 3), the best approximation would involve transmitting those stochastic gradients and calculating the stochastic Gram matrix on the server. However, this would require $M \times d$ communication, where M is the number of objectives and d is the size of the model. With `ApproxGramJacobian`, we reduce communication cost while approximating the Gram matrix. We define the stochastic Jacobian of Client i as $\widetilde{H}_i \triangleq \left[\widetilde{\nabla} f_{i,1}(\mathbf{x}), \ldots, \widetilde{\nabla} f_{i,M}(\mathbf{x})\right] \in \mathbb{R}^{d \times M}$. The goal is to approximate $\frac{1}{(|\mathcal{B}|)^2} \left(\sum_{i \in \mathcal{B}} \widetilde{H}_i\right)^\top \left(\sum_{i \in \mathcal{B}} \widetilde{H}_i\right)$ on the server.

We introduce a compression operator $\mathcal{Q}$, which compresses $\widetilde{H}_i$ into H_i for transmission at a lower communication cost. In our proposed algorithm, we use a randomized-SVD-based compressor, applied after reshaping $\widetilde{H}_i$ into a square matrix by padding it with zeros to handle non-square cases. After the leading components are communicated, the matrix is reconstructed and returned to its original shape. The number of leading components to communicate is selected such that the upload cost is fixed at d. We provide two options for `ApproxGramJacobian`:

1. *One-way communication*: The first option requires each client i to upload $Q(\widetilde{H}_i)$, which has an upload size of d. After receiving these compressed matrices, the server reconstructs the approximate Jacobian matrix, $\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{Q}\left(\widetilde{H}_i\right)$, and estimates the Gram matrix as:

$$G = \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{Q}\left(\widetilde{H}_i\right)\right)^\top \left(\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{Q}\left(\widetilde{H}_i\right)\right).$$

2. *Two-way communication*: In most wireless networks, downloads are typically less costly than uploads (McMahan et al., 2017; McMahan and Ramage, 2017). This option requires each client i to perform an additional download of size d to further reduce the approximation error. To achieve this, we first conduct a deeper analysis of the error components in the approximation. Define the compression error of $\widetilde{H}_i$ as R_i, such that $\widetilde{H}_i = \mathcal{Q}\left(\widetilde{H}_i\right) + R_i = H_i + R_i$. We expand the target Gram matrix below, annotating each term to indicate the communication cost required to obtain the term on the server side. For the communication analysis, we ignore terms of size M or M^2 relative to d, since $M^2 \ll d$. For instance, in the CelebA-5 experiment, $M = 5$ and $d = 11.2$M.

$$\begin{aligned}
\left(\sum_{i\in\mathcal{B}}\widetilde{H}_i\right)^\top\left(\sum_{i\in\mathcal{B}}\widetilde{H}_i\right) &= \underbrace{\sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i}_{\substack{M\times M\\ \text{upload}}} + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}\widetilde{H}_i^\top\widetilde{H}_j \\
&= \sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}\left(H_i+R_i\right)^\top\left(H_j+R_j\right) \\
&= \underbrace{\sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i}_{\substack{M\times M\\ \text{upload}}} + \underbrace{\sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top H_j}_{\substack{\approx d\\ \text{upload}}} + 2\sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top R_j + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}R_i^\top R_j. \qquad (13)
\end{aligned}$$

With an upload size of $M \times M + d$ per client, the server can exactly recover the first two terms of Equation (13). Additionally, we observe that with an extra upload of size d, the third term can also be estimated. To achieve this, the server compresses the sum of the received H_j matrices, $\sum_{j\in\mathcal{B}} H_j$, using $\mathcal{Q}$, and sends $\mathcal{Q}\left(\sum_{j\in\mathcal{B}} H_j\right)$ back to the clients. Each client i then uses $\mathcal{Q}\left(\sum_{j\in\mathcal{B}} H_j\right)$, along with its own H_i and R_i, to approximately recover the third term in Equation (13), requiring an additional upload of size $M \times M$ and a download of size d.

$$\begin{aligned}
&\textit{Equation}\ (13) = \sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top H_j + 2\sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top R_j + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}R_i^\top R_j \\
&= \sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top H_j + 2\sum_{i\in\mathcal{B}}R_i^\top\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_j + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}R_i^\top R_j \\
&= \sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top H_j + 2\sum_{i\in\mathcal{B}}R_i^\top\left(\sum_{j\in\mathcal{B}}H_j - H_i\right) + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}R_i^\top R_j \\
&\approx \underbrace{\sum_{i\in\mathcal{B}}\widetilde{H}_i^\top\widetilde{H}_i}_{\substack{M\times M\\ \text{upload}}} + \underbrace{\sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}H_i^\top H_j}_{\substack{\approx d\\ \text{upload}}} + \underbrace{2\sum_{i\in\mathcal{B}}\underbrace{R_i^\top}_{\substack{\text{already}\\ \text{on client } i}}\left(\underbrace{\mathcal{Q}\left(\sum_{j\in\mathcal{B}}H_j\right)}_{\substack{\approx d\\ \text{download}}} - \underbrace{H_i}_{\substack{\text{already}\\ \text{on client } i}}\right)}_{\substack{\approx d \text{ download}\\ +M\times M \text{ upload}}} + \sum_{i\in\mathcal{B}}\sum_{\substack{j\in\mathcal{B}\\ j\neq i}}R_i^\top R_j \qquad (14)
\end{aligned}$$

We treat the last term as an error. This two-way communication option requires an upload of size $(2M \times M + d)$ and a download of size d. Since M, the number of objectives, is much smaller than the model dimension, the communication cost is approximately an upload of size d and a download of size d for each participating client per round.

We use randomized-SVD-based compression as Q in our proposed methods. In the main experiments, we utilize the second option with two-way communication. However, we note that the difference in performance, in terms of accuracy and loss, between the two options is minimal. The detailed steps of `ApproxGramJacobian` are outlined in Algorithm 6.

Algorithm 6 `ApproxGramJacobian` full procedure

```
1: Input: participating client set $\mathcal{B}$, model $\mathbf{x}$, option
2: for each client $i \in \mathcal{B}$ in parallel do
3:     $\widetilde{H}_i \triangleq \left[\widetilde{\nabla} f_{i,1}(\mathbf{x}), \dots, \widetilde{\nabla} f_{i,M}(\mathbf{x})\right] \in \mathbb{R}^{d \times M}$
4:     $\bar{H}_i \leftarrow$ Reshape($\widetilde{H}_i$) to $\mathbb{R}^{\sqrt{dM} \times \sqrt{dM}}$
5:     Send $\widehat{H}_i \leftarrow$ randomized-SVD($\bar{H}_i, r$), the rank $r$ approximation of $\bar{H}_i$, to server    # $d$ upload
6:     $H_i \leftarrow$ Reshape $\widehat{H}_i$ back to $\mathbb{R}^{d \times M}$ at the server
7: end for
8: if the first option, one-way communication, is used then
9:     $G \leftarrow \left(\frac{1}{|\mathcal{B}|} \sum_i H_i\right)^\top \left(\frac{1}{|\mathcal{B}|} \sum_i H_i\right)$ at the server
10: else if the second option, two-way communication, is used then
11:     At the server: $\bar{h} \leftarrow$ Reshape($\sum_{i \in \mathcal{B}} H_i$) to $\mathbb{R}^{\sqrt{dM} \times \sqrt{dM}}$
12:     Server sends $\widehat{h} \leftarrow$ randomized-SVD $(\bar{h}, r)$, the rank $r$ approximation of $\bar{h}$, to clients in $\mathcal{B}$    # $d$ download
13:     for each client $i \in \mathcal{B}$ in parallel do
14:         $h \leftarrow$ Reshape $\widehat{h}$ back to $\mathbb{R}^{d \times M}$
15:         Send $\widetilde{H}_i^\top \widetilde{H}_i$ to the server    # $M \times M$ upload
16:         Send $R_i^\top (h - H_i)$ to server where $R_i = \widetilde{H}_i - H_i$    # $M \times M$ upload
17:     end for
18:     $G \leftarrow \frac{1}{|\mathcal{B}|^2} \left(\sum_{i \in \mathcal{B}} \widetilde{H}_i^\top \widetilde{H}_i + \sum_{i \in \mathcal{B}} \sum_{\substack{j \in \mathcal{B} \\ j \neq i}} H_i^\top H_j + 2 \sum_{i \in \mathcal{B}} R_i^\top (h - H_i)\right)$ at the server
19: end if
20: Return $G$
```

The rank r used in randomized-SVD-based compression is selected so that each client's upload budget is $d \times 1$, equivalent to the size of one model. Both the *one-way* and *two-way communication* options require an upload cost of d, while the *two-way communication* option also incurs an additional d download for each participating client per round. The total communication cost for both options is $\Theta(d)$.

Next, we compare the empirical performance of the two communication options for the proposed `ApproxGramJacobian`. Additionally, we evaluate the performance of `ApproxGramJacobian` against two commonly used compression methods as alternatives to randomized-SVD-based compression.

Comparison with other compression methods. We compare the proposed communication-efficient approximation method, `ApproxGramJacobian`, which uses randomized-SVD-based compression (Halko et al., 2011), with top-k sparsification and random masking. In top-k sparsification, only the k elements with the largest absolute values are communicated, with k determined by the compression ratio, considering the size of the index map of the communicated entries as well. In random masking, a subset of entries at random positions is sent, with the number of entries determined by the compression ratio. For a fair comparison, we ensure that the communication costs for randomized-SVD-based compression, top-k sparsification, and random masking are the same. The compared methods are `ApproxGramJacobian` with *one-* and *two-way communication* options using randomized-SVD-based compression, top-k sparsification, and random masking. Additionally, we consider using the sum of the clients' Gram matrices as an estimate of the global Gram matrix $\left(\sum_i \widetilde{H}_i^\top \widetilde{H}_i\right)$ with a proper scale.

We run `FedCMOO-Pref` (with uniform preference across objectives) on different datasets, sending all stochastic gradients from clients to the server. At each round, we evaluate the reconstruction error of the Gram matrix (G) by comparing the ground truth, computed using all stochastic gradients, with the approximations obtained from the different compression methods. We use normalized root mean squared error (nRMSE) as the error metric, defined as nRMSE(ground truth, estimate) $= \frac{\|\text{ground truth} - \text{estimate}\|_2}{\|\text{ground truth}\|_2}$. The results are presented in Table 3. We observe that the proposed `ApproxGramJacobian` with randomized-SVD provides better estimation performance for the Gram matrix (G) of the stochastic Jacobian compared to other compression methods as the number of objectives and model dimension grow. Although methods have slight performance differences in the experiments with small number of objectives, randomized-SVD outperforms others in CelebA experiment with 40 objectives.

Table 3: Average nRMSE (%) of the estimation of Gram matrix of the stochastic Jacobian (G) in experiments with `FedCMOO-Pref`.

Compression Method	CelebA	CIFAR10+ MNIST	MNIST+ F.MNIST	MultiMNIST	Mean
Sum of client Gram matrices	50.95	11.51	20.61	14.56	24.41
`ApproxGramJacobian` *two-way comm.* with randomized-SVD	10.15	1.38	2.04	1.76	3.83
`ApproxGramJacobian` *two-way comm.* with random masking	51.25	10.26	17.40	12.98	22.97
`ApproxGramJacobian` *two-way comm.* with top-k sparsification	17.15	0.15	0.04	0.15	4.38
`ApproxGramJacobian` *one-way comm.* with randomized-SVD	10.87	1.68	1.79	1.96	4.08
`ApproxGramJacobian` *one-way comm.* with random masking	76.54	4.23	6.29	5.70	23.19
`ApproxGramJacobian` *one-way comm.* with top-k sparsification	24.13	0.04	0.01	0.10	6.07

C FedCMOO-Pref ALGORITHM

We provide the details of the proposed preference-based federated communication-efficient algorithm, `FedCMOO-Pref`.

C.1 The Preference Definition and Evaluation Metric

We consider an experimental setting where the user specifies the preference vector $\mathbf{r} \in \mathbb{R}_+^M$, whose entries are preference weights, $r_1, r_2, \ldots, r_M$. The aim is to find a Pareto stationary solution $\mathbf{x}$ that satisfies $r_1 F_1(x) = r_2 F_2(x) = \cdots = r_M F_M(x)$. Given the preference vector $\mathbf{r} \in \mathbb{R}_+^M$, the problem of interest is formulated as:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^d} \mathbf{F}(\mathbf{x}) := [F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_M(\mathbf{x})]^\top \\ &\text{subject to} \quad r_1 F_1(\mathbf{x}) = \cdots = r_M F_M(\mathbf{x}). \end{aligned} \tag{15}$$

Inspired by Mahapatra and Rajan (2020), we follow a KL divergence-based method to solve Equation (15). First, we define the normalized scaled losses of model $\mathbf{x}$ for a given preference vector $\mathbf{r}$:

$$\hat{\mathbf{u}}(\mathbf{r}) \triangleq \frac{\mathbf{r} \odot \mathbf{F}(\mathbf{x})}{\sum_k r_k F_k(\mathbf{x})}$$

where $\odot$ denotes element-wise product. If the constraint of Equation (15) is satisfied, each entry $\hat{u}_k$ ($\forall\ k \in [M]$) of $\hat{\mathbf{u}}(\mathbf{r})$ should be $1/M$. Then, we define the KL divergence, which measures the distance of $\hat{\mathbf{u}}$ from the uniform vector with entries of $1/M$, i.e. non-uniformity,

$$\mu_{\mathbf{r}} = \mathrm{KL}\left(\hat{\mathbf{u}}(\mathbf{r}) \mid\mid \mathbf{1}/M\right) = \sum_{k\in[M]} \hat{u}_k \log\left(\frac{\hat{u}_k}{1/M}\right).$$

C.2 The Descent Direction and Task Weights

By Theorem 1 in Mahapatra and Rajan (2020), it is guaranteed that when we have exact gradients $\{\nabla F_k(\mathbf{x})\}_{k\in[M]}$, an update step $\mathbf{x} \leftarrow \mathbf{x} - \eta \underbrace{\nabla \mathbf{F}(\mathbf{x})\,\mathbf{a}}_{\sum_{k\in[M]} a_k \nabla F_k(\mathbf{x})}$ does not increase $\mu_{\mathbf{r}}$ with a sufficiently small step size η where $a_k = r_k \left(\log\left(\frac{\hat{u}_k}{1/M}\right) - \mu_{\mathbf{r}}\right)$.

We use an indicator function, $\mathbb{1}_\mu$, which is 0 if $\mu_{\mathbf{r}} \leq \epsilon$ and 1 otherwise, where the threshold $\epsilon = 0.01$ is used in the experiments. To find the update direction using exact gradients, the task weights can be determined by either maximizing the total descent across objectives or decreasing the KL divergence, such that:

$$\max_{\mathbf{w}\in\mathcal{S}_M} \mathbf{w}^\top \nabla\mathbf{F}(\mathbf{x})^\top \nabla\mathbf{F}(\mathbf{x})\left(\mathbf{a}\mathbb{1}_\mu + \mathbf{1}(1-\mathbb{1}_\mu)\right). \tag{16}$$

However, solving Equation (16) may not yield task weights that perform well, as decreasing KL divergence $\mu_\mathbf{r}$ without any constraints may lead to increases in all objectives, which is undesirable. To address this, first, the following sets are defined:

1. $J \triangleq \left\{k \mid \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}\right)^\top \nabla F_k(\mathbf{x}) > 0\right\}$: The indices of objectives whose gradients are positively aligned with the direction of decreasing KL divergence, $\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}$. The loss of these objectives is expected to decrease.
2. $\bar{J} \triangleq \left\{k \mid \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}\right)^\top \nabla F_k(\mathbf{x}) \leq 0\right\}$: The indices of objectives whose gradients are negatively aligned with or perpendicular to the direction of decreasing KL divergence, $\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}$. The loss of these objectives is expected to increase or remain the same.
3. $J^* \triangleq \{k \mid k \in \arg\max_j r_j F_j(\mathbf{x})\}$: The indices of objectives for which the preference multiplied by the current loss value is the maximum across all objectives. These are the objectives for which we should not increase the loss.

To ensure not increasing the loss of objectives in J^*, we require the solution weights $\mathbf{w}^*$ to satisfy:

$$\left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{w}^*\right)^\top \nabla F_k(\mathbf{x}) \geq 0, \qquad \forall k \in J^*.$$

Additionally, to prevent a potential increase in the loss functions of all objectives when J is an empty set, we introduce the following constraint. We define $\mathbb{1}_{J\neq\varnothing}$ as an indicator function that returns 0 if J is empty and 1 otherwise:

$$\left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{w}^*\right)^\top \nabla F_k(\mathbf{x}) \geq \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}\right)^\top \nabla F_k(\mathbf{x})\,\mathbb{1}_{J\neq\varnothing}, \qquad \forall k \in \bar{J} \setminus J^*.$$

These two constraints form $\mathcal{W}$, the practical constraints mentioned in Problem (8) in the main text. Incorporating these constraints into Problem (16), we obtain:

$$\begin{aligned} \text{Problem (8)} \equiv \max_{\mathbf{w}\in\mathcal{S}_M}\ & \mathbf{w}^\top \nabla\mathbf{F}(\mathbf{x})^\top \nabla\mathbf{F}(\mathbf{x})\left(\mathbf{a}\mathbb{1}_\mu + \mathbf{1}(1-\mathbb{1}_\mu)\right) \\ \text{s.t.}\quad & \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{w}\right)^\top \nabla F_k(\mathbf{x}) \geq 0, \quad \forall k \in J^*, \\ & \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{w}\right)^\top \nabla F_k(\mathbf{x}) \geq \left(\nabla\mathbf{F}(\mathbf{x})\,\mathbf{a}\right)^\top \nabla F_k(\mathbf{x})\,\mathbb{1}_{J\neq\varnothing}, \quad \forall k \in \bar{J} \setminus J^*. \end{aligned} \tag{17}$$

We refer the reader to Mahapatra and Rajan (2020) for further explanations and insights.

C.3 Preference-based Federated Communication-Efficient MOO

Since exact gradients cannot be calculated and communication is limited in our federated problem setting, Equation (17) cannot be solved exactly. Instead, similar to `FedCMOO`, we first approximate the Gram matrix of the Jacobian, $G \approx \nabla\mathbf{F}(\mathbf{x})^\top \nabla\mathbf{F}(\mathbf{x})$, on the server. Clients also share their loss values, which consist of only M scalar values per client, with the server, allowing the server to estimate the global objective loss values by averaging them. With these approximations, all the steps presented in Appendix C.2 can be performed approximately on the server in a communication-efficient way.

First, `FedCMOO-Pref` uses the same `ApproxGramJacobian` subroutine as `FedCMOO` to obtain $G = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M] \approx \nabla\mathbf{F}(\mathbf{x})^\top \nabla\mathbf{F}(\mathbf{x})$ on the server, where $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M$ are the columns of G. Additionally, using the local loss values of the participating clients, the objective value estimates, $\widetilde{F}_1(\mathbf{x}), \ldots, \widetilde{F}_M(\mathbf{x})$, are computed. The server then calculates the normalized losses using these objective value estimates. Subsequently, the KL divergence $\mu_\mathbf{r}$ and the $\mathbf{a}$ weights are calculated. Note that the sets J, $\bar{J}$, and J^* can be approximated using only the columns of matrix G and the estimated loss values, without needing individual objective gradients, as follows:

$$J = \left\{k \mid \mathbf{a}^\top \mathbf{g}_k > 0\right\}, \qquad \bar{J} = \left\{k \mid \mathbf{a}^\top \mathbf{g}_k \leq 0\right\} \qquad J^* = \left\{k \mid k \in \arg\max_j r_j \widetilde{F}_j(\mathbf{x})\right\} \tag{18}$$

Then the following approximation to Problem (17) is solved on the server to find task weights.

$$\begin{aligned}
&\max_{\mathbf{w}\in\mathcal{S}_M} \ \mathbf{w}^\top G\left(\mathbf{a}\mathbb{1}_\mu + \mathbf{1}(1-\mathbb{1}_\mu)\right) \\
&\text{s.t.} \quad \mathbf{w}^\top \mathbf{g}_k \geq 0, \quad \forall k \in J^*, \\
&\qquad\ \ \mathbf{w}^\top \mathbf{g}_k \geq \mathbf{a}^\top \mathbf{g}_k \mathbb{1}_{J\neq\varnothing}, \quad \forall k \in \bar{J} \setminus J^*.
\end{aligned} \tag{19}$$

We depict the `FedCMOO-Pref` in Algorithm 7, which differs from `FedCMOO` only in how the task weights are obtained. Additionally, we provide a detailed presentation of the `FindWeights-Pref` subroutine in Algorithm 4, with more details available in Algorithm 8.

Algorithm 7 `FedCMOO-Pref`

```
1: Input: client and server learning rates $\eta_c, \eta_s$, number of local steps $\tau$, preference vector $\mathbf{r}$
2: Initialize: global model $\mathbf{x}^{(0)} \in \mathbb{R}^d$, and task weights $\mathbf{w}^{(0)} \leftarrow [1/M, \ldots, 1/M]^\top \in \mathcal{S}_M$
3: for $t = 0, 1, \ldots, T-1$ do
4:     Select the client set $\mathcal{B}^{(t)}$ uniformly at random from $[N]$; send $\mathbf{x}^{(t)}$ to the clients in $\mathcal{B}^{(t)}$
5:     $G^{(t)} \leftarrow$ ApproxGramJacobian($\mathcal{B}^{(t)}, \mathbf{x}^{(t)}$) at the server, where $G^{(t)} \approx \nabla\mathbf{F}(\mathbf{x}^{(t)})^\top \nabla\mathbf{F}(\mathbf{x}^{(t)})$
6:     $\widetilde{\mathbf{F}}^{(t)} \leftarrow$ Average loss values of participating clients in $\mathcal{B}^{(t)}$ for all objectives
7:     Compute $\mathbf{w}^{(t+1)} \leftarrow$ FindWeights-Pref($\mathbf{r}$, $\widetilde{\mathbf{F}}^{(t)}$, $G^{(t)}$) at the server and send to clients in $\mathcal{B}^{(t)}$
8:     for each client $i \in \mathcal{B}^{(t)}$ in parallel do
9:         Initialize local model: $\mathbf{x}_i^{(t,0)} \leftarrow \mathbf{x}^{(t)}$
10:        for $r = 0, \ldots, \tau - 1$ do
11:            $\mathbf{x}_i^{(t,r+1)} \leftarrow \mathbf{x}_i^{(t,r)} - \eta_c \sum_{k=1}^{M} w_k^{(t+1)} \widetilde{\nabla} f_{i,k}(\mathbf{x}_i^{(t,r)})$
12:        end for
13:        Send $\Delta_i^{(t)} \triangleq \frac{1}{\tau\eta_c}\left(\mathbf{x}^{(t)} - \mathbf{x}_i^{(t,\tau)}\right)$ to the server
14:    end for
15:    $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta_s\eta_c\tau \frac{1}{|\mathcal{B}^{(t)}|} \sum_{i\in\mathcal{B}^{(t)}} \Delta_i^{(t)}$
16: end for
17: Return $\mathbf{x}^{(T)}$
```

Algorithm 8 `FindWeights-Pref` in detail

```
1: Input: preferences $\mathbf{r}$, losses $\mathbf{F}$, Gram matrix $G = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_M]$
2: Compute normalized losses $\hat{\mathbf{u}}(\mathbf{r}) \triangleq \frac{\mathbf{r}\odot\mathbf{F}(\mathbf{x})}{\sum_k r_k F_k(\mathbf{x})}$
3: Compute non-uniformity: $\mu_{\mathbf{r}} = \mathrm{KL}\left(\hat{\mathbf{u}}(\mathbf{r}) \mid\mid \frac{\mathbf{1}}{M}\right)$
4: Compute $\mathbf{a} \in \mathbb{R}^M$ such that $a_k = r_k\left(\log\left(\frac{\hat{u}_k}{1/M}\right) - \mu_{\mathbf{r}}\right)$ for all $k \in [M]$
5: $J \leftarrow \left\{k \mid \mathbf{a}^\top \mathbf{g}_k > 0\right\}$
6: $\bar{J} \leftarrow \left\{k \mid \mathbf{a}^\top \mathbf{g}_k \leq 0\right\}$
7: $J^* \leftarrow \left\{k \mid k \in \arg\max_j r_j \widetilde{F}_j\left(\mathbf{x}\right)\right\}$
8: $\mathbf{w} \leftarrow$ a solution to optimization problem (19)
9: Return $\mathbf{w}$
```

Practical improvement. We observe that enforcing a minimum weight on every task accelerates training with the `FedCMOO-Pref` method, even when this minimum weight is small. In the `FedCMOO-Pref` experiments, except for Figure 4, where the goal is to find a Pareto solution aligned with specific preference, we project the output of line 7 in Algorithm 7, $\mathbf{w}^{(t+1)}$, onto a vector where each entry is at least $\frac{1}{5\times M}$, where M is the number of objectives.

D EXPERIMENTAL DETAILS

We conduct simulated experiments in a federated multi-objective optimization setting, primarily using PyTorch and CVXPY packages on our internal clusters equipped with NVIDIA H100 Tensor Core GPUs. In this section, we describe the experiments in detail.

D.1 Datasets

We use six different datasets in the experiments:

1. **MultiMNIST**: We create our MultiMNIST dataset using the well-known MNIST dataset (Deng, 2012). The images are maintained at 28×28 pixels with 1 channel. For each sample, we place two random digits, one at the top-left and the other at the bottom-right, after cropping and resizing them. Each sample is assigned a new label that combines the labels of the two digits, resulting in 100 unique labels. Samples from this dataset are shown in Figure 5.

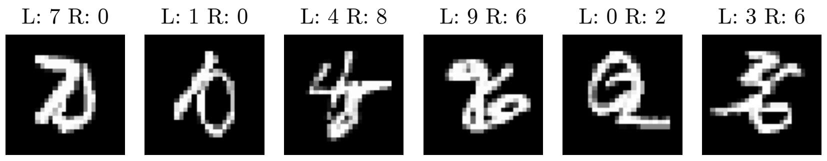

Figure 5: Samples from created MultiMNIST dataset. The captions indicate the individual digits of the left (L) and right (R) samples merged.

2. **MNIST+FMNIST**: Similar to the construction of MultiMNIST, we create another dataset by combining two randomly sampled images: one from the MNIST dataset and one from the FashionMNIST dataset (Xiao et al., 2017). The labeling process follows the same approach as in MultiMNIST. Samples from this dataset are shown in Figure 6.

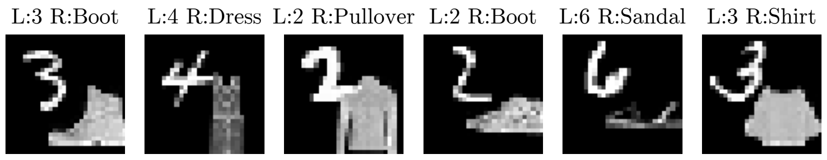

Figure 6: Samples from created MNIST+FMNIST dataset. The captions indicate the individual labels of the MNIST (L) and FashionMNIST (R) samples merged.

3. **CIFAR10+FMNIST**: We use three-channel images from the CIFAR-10 dataset (Krizhevsky et al., 2009) and randomly selected MNIST digits. Each digit is cropped, resized, and placed in the center of all channels. The labeling process is similar to that of MultiMNIST and MNIST+FMNIST. Samples from this dataset are shown in Figure 7.

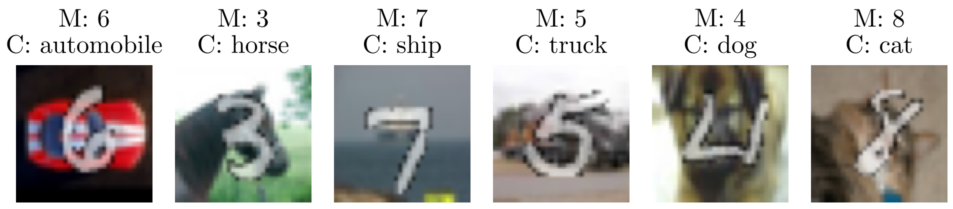

Figure 7: Samples from created CIFAR10+FMNIST dataset. The captions indicate the individual labels of MNIST (M) and CIFAR-10 (C) samples merged.

4. **CelebA** and **CelebA-5**: CelebA is a large-scale face dataset (Liu et al., 2015), where each sample contains 40 binary attributes, resulting in a 40-objective problem. We also construct the CelebA-5 dataset with 5

objectives by partitioning the 40 attributes of CelebA into 5 groups of 8 attributes each. The attributes are grouped based on visual and facial characteristics, ensuring a balanced representation of different facial features in each group. The groups are as follows:
Group 1: 5_o_Clock_Shadow, Blond_Hair, Bags_Under_Eyes, Eyeglasses, Mustache, Wavy_Hair, Oval_Face, Rosy_Cheeks,
Group 2: Arched_Eyebrows, Black_Hair, Big_Nose, Blurry, Goatee, Straight_Hair, Pale_Skin, Wearing_Earrings,
Group 3: Attractive, Brown_Hair, Bushy_Eyebrows, Double_Chin, Mouth_Slightly_Open, Receding_Hairline, Wearing_Hat, Young,
Group 4: Bald, Big_Lips, Chubby, Heavy_Makeup, High_Cheekbones, Male, Sideburns, Smiling,
Group 5: Bangs, Gray_Hair, Narrow_Eyes, No_Beard, Pointy_Nose, Wearing_Lipstick, Wearing_Necklace, Wearing_Necktie.

In both datasets, we assign each sample with new composite labels using the binary value combinations of attributes 'Attractive', 'Male', 'Mouth_Slightly_Open'. In total there are 8 possible new composite labels.

5. **QM9**: QM9 is a widely used 11-task regression benchmark (Ramakrishnan et al., 2014). We follow the implementations in Lin and Zhang (2023); Zhu et al. (2022). The QM9 dataset consists of graph data for molecules, with 40k samples used for training and 20k samples for validation and testing, randomly split. The training samples are distributed to the clients uniformly at random. All 11 tasks involve predicting continuous-valued molecular properties. A graph neural network model as described in Lin and Zhang (2023) is used as the backbone, and linear models are employed as decoders. Mean-squared error is used both as the loss function for training and as the evaluation metric.

D.2 Federated Setting Details and Data Partitioning

Our simulated federated setting consists of $N = 100$ clients, except in QM9 experiments, where we use $N = 20$ clients. To create data heterogeneity across clients in the experiments, we use a Dirichlet distribution over the newly assigned composite labels. We follow this distribution to determine the number of samples each client receives from each label. We follow the implementation from Acar et al. (2021). In QM9 experiments, data samples are distributed uniformly at random across clients. All clients are assigned an equal number of data points.

D.3 Models, Training, and Hyperparameter Selection

We use convolutional neural network (CNN) models of varying sizes in all experiments, except for QM9, where we use a graph neural network (GNN). For MultiMNIST and MNIST+FMNIST, we employ a LeNet-like CNN encoder (Lecun et al., 1998; Sener and Koltun, 2018) with two linear layers as the decoders, resulting in 34.6k parameters. In CIFAR10+FMNIST, we use a CNN encoder adapted from Acar et al. (2021) with a single linear layer as the decoders, totaling 1.73M parameters. For CelebA and CelebA-5, we utilize a ResNet-based encoder adapted from Jhunjhunwala et al. (2023), paired with one linear layer as the decoders, resulting in 11.2M parameters. In QM9, we use a GNN model as described in Lin and Zhang (2023) as the backbone, with linear decoders, totaling 617k parameters.

We use negative log-likelihood loss for MultiMNIST, MNIST+FMNIST, and CIFAR10+FMNIST, and binary cross-entropy loss for the CelebA experiments unless otherwise specified. For MultiMNIST, MNIST+FMNIST, and CIFAR10+FMNIST, we apply random rotations as data augmentation during training. The batch size is fixed at 128, and we perform 10 local training steps per round unless otherwise stated. Stochastic Gradient Descent (SGD) without momentum is used as the optimizer. For QM9, mean-squared error is used as the loss function, Adam optimizer is employed, and 14 local training steps are performed per round.

All datasets except for QM9 have predefined training and test splits, which we use along with the default validation splits, when available. If a validation split is not provided, we randomly split the training set, using 80% for training and 20% for validation. For QM9, we use 40k training samples and 20k validation and test samples selected randomly.

We use constant learning rates (without changing during the training) in experiments. To select the learning rates, we run the methods on each experimental setting and evaluate them on the validation set.

For the CelebA experiment, the global learning rate (η_s) is set to 1, and the local learning rate (η_c) is 0.2. In the CelebA5 experiment, the global learning rate is 1.6, and the local learning rate is 0.3.

For the MultiMNIST and MNIST+FMNIST experiments, the learning rates are the same but vary depending on the algorithm: for `FSMGDA`, the global learning rate is 2, and the local learning rate is 0.1; for `FedCMOO`, the global learning rate is 1.2, and the local learning rate is 0.5; for `FedCMOO-Pref`, the global learning rate is 1.6, and the local learning rate is 0.3.

For, the CIFAR10+FMNIST experiment, the global learning rate is 1.6, and the local learning rate is 0.3 for all methods. Finally, for the QM9 experiment, the global learning rate is set to 1, and the local learning rate is 0.01 for all methods.

E ADDITIONAL EXPERIMENTAL RESULTS

We present the additional experimental results here which are not included in the main text due to space limitation.

E.1 The Training Curves Of The Experiments

We show the mean test accuracy and mean test loss curves of the compared methods (similar to the experiment in Figure 1 in the main text) in Figures 8 and 9.

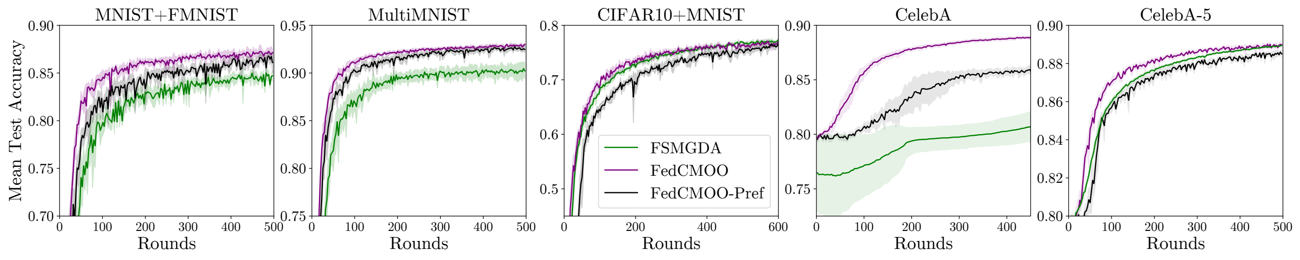

Figure 8: Mean test accuracy in MNIST+FMNIST, MultiMNIST, CIFAR10+FMNIST, CelebA, and CelebA-5 datasets. `FedCMOO` outperforms the `FSMGDA` in training speed and final accuracy. `FedCMOO-Pref` with uniform preference either outperforms or is surpassed by `FSMGDA` in terms of mean accuracy, but `FedCMOO-Pref` trains the model for all objectives more uniformly across all tasks (see Table 1 in the main text).

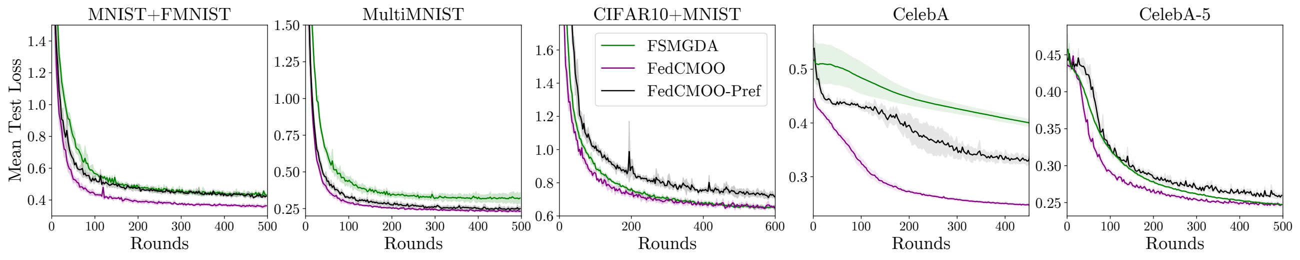

Figure 9: Mean test loss in MNIST+FMNIST, MultiMNIST, CIFAR10+FMNIST, CelebA, and CelebA-5 datasets. `FedCMOO` outperforms the `FSMGDA` in training speed and final loss.

E.2 The Final Loss Values Of CelebA And CelebA-5

We present the final loss values for each task in the CelebA and CelebA-5 experiments shown in Figure 9, using radial plots in Figure 10. The performance of `FSMGDA` worsens relative to other methods as the number of objectives increases due to local training drift.

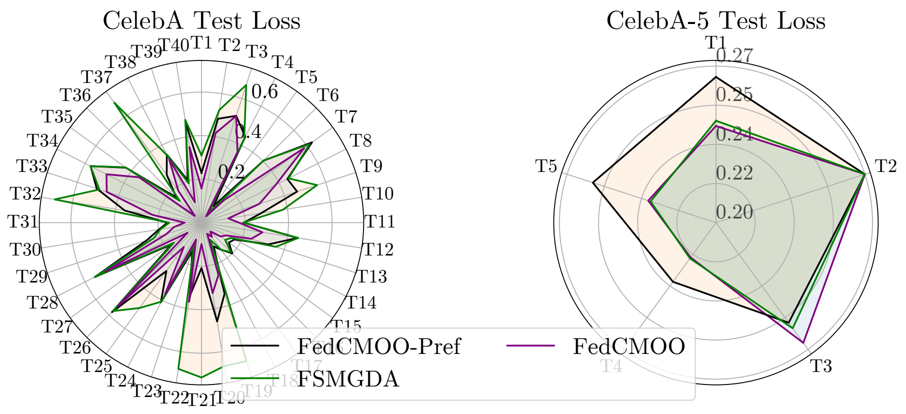

Figure 10: Final test loss values of all objectives in the CelebA and CelebA-5 experiments. `FSMGDA` performs poorly as the number of objectives increases.

E.3 Δ_M Metric Results of the Main Experiments in Table 1

Δ_M is a widely used metric in MOO literature (Fernando et al., 2022), measuring the average percentage performance loss of multi-objective training compared to single-objective training. It is defined for a method $\mathcal{A}$ relative to a baseline $\mathcal{B}$ as:

$$\Delta_M = \frac{1}{M}\sum_{m=1}^{M}(-1)^{\ell_m}\frac{S_{\mathcal{A},m} - S_{\mathcal{B},m}}{S_{\mathcal{B},m}},$$

where $S_{\mathcal{A},m}$ and $S_{\mathcal{B},m}$ denote the performance of objective m under methods $\mathcal{A}$ and $\mathcal{B}$, respectively. M is the number of objectives, and ℓ_m is 0 if lower values of S indicate better performance and 1 otherwise (Fernando et al., 2022).
In Table 4, we report Δ_M for the accuracy results of the experiments shown Table 1. In this calculation, $S_{\mathcal{A},m}$ corresponds to the accuracy of objective m in multi-objective training with algorithm $\mathcal{A}$, while $S_{\mathcal{B},m}$ is the accuracy when training the objective individually. For example, Δ_M for `FedCMOO` on the MultiMNIST dataset is computed as:

$$\frac{1}{2}\Bigg(\frac{\text{Single-task Digit 1 accuracy} - \texttt{FedCMOO}\text{ Digit 1 accuracy}}{\text{Single-task Digit 1 accuracy}} + \frac{\text{Single-task Digit 2 accuracy} - \texttt{FedCMOO}\text{ Digit 2 accuracy}}{\text{Single-task Digit 2 accuracy}}\Bigg).$$

Table 4: The Δ_M values for the test accuracy levels in experiments in Table 1 in the main text. Smaller values are better.

Experiment	MNIST+ FMNIST	Multi MNIST	CIFAR10+ MNIST
`FSMGDA` (Yang et al., 2023)	10.22%	4.26%	5.63%
`FedCMOO`	**7.04**%	**0.79**%	5.93%
`FedCMOO-Pref`	7.60%	1.91%	**5.12**%

While this metric reflects average performance, Table 1 in the main text provides detailed task-specific results. The highest achievable accuracies for the two-objective federated settings in Table 4 are as follows: For MNIST+FMNIST, 97.0% (MNIST) and 90.1% (FashionMNIST); for MultiMNIST, 95.4% (Digit 1) and 93.1% (Digit 2); and for CIFAR10+FMNIST, 64.9% (CIFAR-10) and 97.1% (MNIST).

E.4 Further Experiments Comparing The Local Training Performance Of FedCMOO And FSMGDA

To further support our claim that local drift across objectives reduces the performance of `FSMGDA`, we designed the following experiment. During `FedCMOO` training in the MultiMNIST and MNIST+FMNIST experiments, we also evaluate the local training performance of `FSMGDA` using the same global models. We assess the clients' local training performance with both `FedCMOO` and `FSMGDA` by measuring accuracy and loss on local datasets at the beginning and end of each local training session. The difference in these values gives us the loss reduction and accuracy improvement during local training. These results are averaged over clients and plotted across rounds. The results, shown in Figure 11, indicate that clients achieve greater local training progress with `FedCMOO` than with `FSMGDA`. Drift across objectives in `FSMGDA` hinders local training performance.

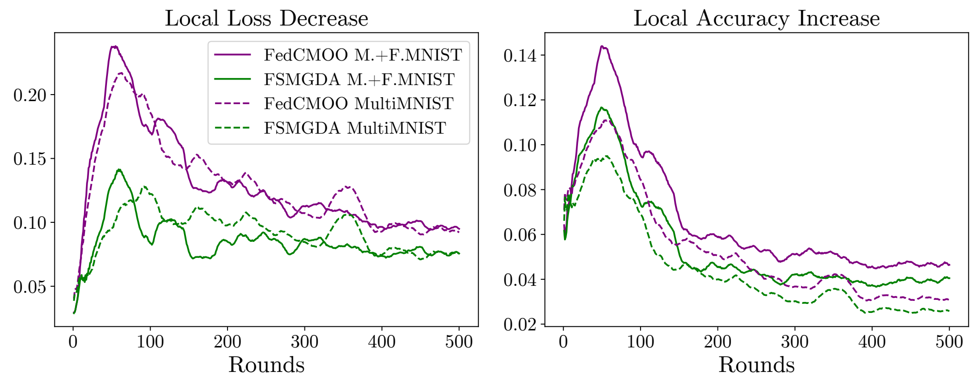

Figure 11: Comparison of average local training progress (loss ↓ and accuracy ↑) across clients between `FedCMOO` and `FSMGDA`. Drifting gradients during local training lead to poor performance in `FSMGDA`. For better visualization, the results are smoothed with a running average over 40 rounds.

E.5 The Final Test Loss Values of All Tasks in the QM9 Experiment

We present the radial plot of the final test accuracies of all individual tasks in QM9 experiment for each method in Figure 12. It further demonstrates the superiority of `FedCMOO` over the baseline on every task.

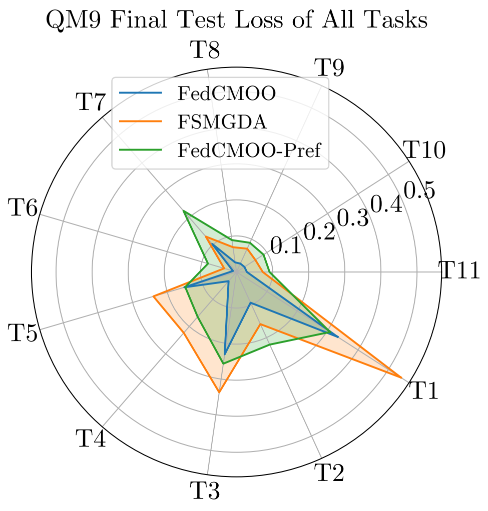

Figure 12: Radial plot showing the final test loss of `FedCMOO`, `FedCMOO-Pref` (with a uniform preference), and `FSMGDA` for all tasks in the QM9 experiment. `FedCMOO` achieves superior performance across tasks.

F THE IMPRACTICAL ASSUMPTION IN THE THEORETICAL ANALYSIS OF Yang et al. (2023)

In this section, we provide a detailed explanation expanding upon the discussion at the end of Section 5, focusing on the nonstandard assumption in Yang et al. (2023). We specifically refer to Assumption 4, Theorem 5, and Corollary 6, all of which are from Yang et al. (2023).

Assumption 4 in Yang et al. (2023) combines Lipschitz continuity and bounded gradient variance by assuming $\mathbb{E}\left[||\nabla f(\mathbf{x},\xi)-\nabla f(\mathbf{y},\xi')||^2\right] \leq \alpha||\mathbf{x}-\mathbf{y}||^2+\beta\sigma^2$. This assumption is reasonable, as there always exists a sufficiently large α and β. However, in Theorem 5, their bound on $\mathbb{E}||\bar{\mathbf{d}}_t||$ contains a $\mathcal{O}(\beta\sigma^2)$ term. From Assumption 4, this term cannot always be arbitrarily small. To ensure its convergence to zero, Corollary 6 imposes the assumption $\beta = \mathcal{O}(\eta)$. This is unusual, as β is a constant that depends on the data distribution and is not a tunable parameter. Since $\eta = \mathcal{O}(1/\sqrt{T})$ is chosen, this assumption effectively forces the $\beta\sigma^2$ term in Assumption 4 to be $\mathcal{O}(\sigma^2/\sqrt{T})$, which is a very strong requirement. One way to satisfy this condition would be to make the batch size at each step depend on T. However, this would significantly worsen the convergence rate with respect to the number of data samples.

G PROOF OF CONVERGENCE THEOREM

We provide the mathematical proofs of the claims in the paper in this section.

G.1 Definitions

The norms are Frobenius norm unless otherwise specified. $\|A\|_{\text{op}}$ denotes the operator norm of matrix A. The notation ζ with subscript is used to clarify the source of randomness when multiple sources could cause confusion.

G.2 `ApproxGramJacobian` and `FedCMOO` Algorithm with All Simplifications for Theoretical Analysis

As explained in Section 5 of the main text, we prove the convergence of `FedCMOO` with slight modifications to `ApproxGramJacobian` subroutine. The resulting algorithm, with the incorporated changes, is presented in Algorithm 9. This algorithm takes as input the model at round t, $\mathbf{x}^{(t)}$, and returns an unbiased approximation of the Gram matrix of the Jacobian of the global loss vector, $\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)$. First, two independent sets of n' clients are sampled. The clients in each set calculate local stochastic Jacobian matrices, compressing the columns (i.e., individual objectives' stochastic gradients) with an unbiased compression operator $\mathcal{Q}$ before sending them to the server. On the server, the collected stochastic Jacobian matrices from both sets of clients are averaged. Using these two averaged matrices, the subroutine returns an unbiased estimate of the Gram matrix of the Jacobian of the global loss vector.

Algorithm 9 `ApproxGramJacobian-T`: `ApproxGramJacobian` with incorporated changes for theoretical analysis

```
1: **Input:** model $\mathbf{x}^{(t)}$
2: $\mathcal{A}_1^{(t)} \leftarrow$Sample $n'$ clients uniformly at random
3: $\mathcal{A}_2^{(t)} \leftarrow$Sample $n'$ clients uniformly at random
4: for $j \in \{1,2\}$ in parallel do
5:     for each client $i \in \mathcal{A}_j^{(t)}$ in parallel do
6:         $\widetilde{H}_i\left(\mathbf{x}^{(t)},\zeta_j\right) \leftarrow \left[\widetilde{\nabla} f_{i,1}\left(\mathbf{x}^{(t)},\zeta_j\right),\ldots,\widetilde{\nabla} f_{i,M}\left(\mathbf{x}^{(t)},\zeta_j\right)\right] \in \mathbb{R}^{d\times M}$
7:         Compress each column of $\widetilde{H}_i\left(\mathbf{x}^{(t)},\zeta_j\right)$ with an unbiased compression operator $\mathcal{Q}$, and send
           $\mathcal{Q}\left(\widetilde{H}_i\left(\mathbf{x}^{(t)},\zeta_j\right)\right) = \left[\mathcal{Q}\left(\widetilde{\nabla} f_{i,1}\left(\mathbf{x}^{(t)},\zeta_j\right)\right),\ldots,\mathcal{Q}\left(\widetilde{\nabla} f_{i,M}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)\right]$ to the server
8:     end for
9:     $Y\left(\mathbf{x}^{(t)},\zeta_j\right) \leftarrow \frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}} \mathcal{Q}\left(\widetilde{H}_i\left(\mathbf{x}^{(t)},\zeta_j\right)\right)$ at the server
10: end for
11: **Return** $Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)$
```

We will use $Y\left(\mathbf{x}^{(t)}, \zeta_j\right)$, as defined in Algorithm 9, throughout the theoretical analysis. For the sake of completeness, we also present the full algorithm, for which the convergence guarantee is proven, in Algorithm 10.

Algorithm 10 `FedCMOO` with `ApproxGramJacobian-T` whose convergence guarantee is shown

```
1:  Input: client and server learning rates $\eta_c, \eta_s$, number of local steps $\tau$
2:  Initialize: global model $\mathbf{x}^{(0)} \in \mathbb{R}^d$, and task weights $\mathbf{w}^{(0)} \leftarrow [1/M, \dots, 1/M]^\top \in \mathcal{S}_M$
3:  for $t = 0, 1, \dots, T-1$ do
4:      Select the client set $\mathcal{B}^{(t)}$ uniformly at random from $[N]$; send $\mathbf{x}^{(t)}$ to the clients in $\mathcal{B}^{(t)}$
5:      $G^{(t)} \triangleq Y\left(\mathbf{x}^{(t)}, \zeta_1\right)^\top Y\left(\mathbf{x}^{(t)}, \zeta_2\right) \leftarrow$ ApproxGramJacobian-T($\mathbf{x}^{(t)}$) at the server, where
        $G^{(t)} \approx \nabla\mathbf{F}(\mathbf{x}^{(t)})^\top \nabla\mathbf{F}(\mathbf{x}^{(t)})$
6:      Compute $\mathbf{w}^{(t+1)} \leftarrow$ FindWeights($\mathbf{w}^{(t)}, G^{(t)}, K = 1$)    # $\mathbf{w}^{(t+1)} \leftarrow \text{proj}_{\mathcal{S}_M}\left(\mathbf{w}^{(t)} - \beta G^{(t)}\mathbf{w}^{(t)}\right)$
        at server and send $\mathbf{w}^{(t+1)}$ to clients in $\mathcal{B}^{(t)}$
7:      for each client $i \in \mathcal{B}^{(t)}$ in parallel do
8:          Initialize local model: $\mathbf{x}_i^{(t,0)} \leftarrow \mathbf{x}^{(t)}$
9:          for $r = 0, \dots, \tau - 1$ do
10:             $\mathbf{x}_i^{(t,r+1)} \leftarrow \mathbf{x}_i^{(t,r)} - \eta_c \sum_{k=1}^M w_k^{(t+1)} \widetilde{\nabla} f_{i,k}(\mathbf{x}_i^{(t,r)})$
11:         end for
12:         Send $\Delta_i^{(t)} \triangleq \frac{1}{\tau\eta_c}\left(\mathbf{x}^{(t)} - \mathbf{x}_i^{(t,\tau)}\right)$ to the server
13:     end for
14:     $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \eta_s\eta_c\tau\frac{1}{|\mathcal{B}^{(t)}|}\sum_{i\in\mathcal{B}^{(t)}} \Delta_i^{(t)}$
15: end for
16: Return $\mathbf{x}^{(T)}$
```

G.3 Restatement of Theoretical Claims

We first restate the main theorem.

Theorem 2 (Restatement of Theorem 1: Convergence of `FedCMOO`)**.** *Suppose Assumptions 1-5 hold, the client learning rate satisfies $\eta_c \leq \frac{1}{2L\tau}$, and the server selects $|\mathcal{B}^{(t)}| = n$ clients in every round. Then the iterates of `FedCMOO` (Algorithm 10) satisfy,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\sum_k w_k^{(t)}\nabla F_k(\boldsymbol{x}^{(t)})\right\|^2 \leq \underbrace{\mathcal{O}\left(\beta M C_1^2\right)}_{\textit{MOO Weight Error}} + \underbrace{\mathcal{O}\left(\frac{1}{T\eta_s\eta_c\tau} + \frac{L\eta_s\eta_c\sigma_l^2}{n}\right)}_{\substack{\textit{Centralized Optimization Error}\\ \textit{for Scalarized Loss}}} + \underbrace{\mathcal{O}\left(L\eta_s\eta_c\tau b^2\right)}_{\substack{\textit{Partial Participation}\\ \textit{Error}}} + \underbrace{\mathcal{O}\left(L\eta_c(\tau b^2 + \sqrt{\tau}b\sigma_l)\right)}_{\textit{Local Drift Error}},$$

where $C_1 \triangleq \mathcal{O}\left(\frac{q+1}{n'}\sigma_l^2 + \frac{qb^2}{n'} + \frac{N-n'}{n'(N-1)}\sigma_g^2 + b^2\right)$ is a constant independent of T and M.

Also, by noticing that $\left\|\sum_k w_k^{*(t)}\nabla F_k(\mathbf{x}^{(t)})\right\|^2 \leq \left\|\sum_k w_k^{(t)}\nabla F_k(\mathbf{x}^{(t)})\right\|^2$ for all t and $\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)$ where $\mathbf{w}^{*(t)} \triangleq \arg\min_{\mathbf{w}\in\mathcal{S}_M}\left\|\sum_k w_k\nabla F_k(\mathbf{x}^{(t)})\right\|^2$, the following holds:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\sum_k w_k^{*(t)}\nabla F_k(\mathbf{x}^{(t)})\right\|^2 \leq \mathcal{O}\left(\beta M C_1^2\right) + \mathcal{O}\left(\frac{1}{T\eta_s\eta_c\tau} + \frac{L\eta_s\eta_c\sigma_l^2}{n}\right) + \mathcal{O}\left(L\eta_s\eta_c\tau b^2\right) + \mathcal{O}\left(L\eta_c(\tau b^2 + \sqrt{\tau}b\sigma_l)\right).$$

Corollary 2.1 (Restatement of Corollary 1.1: Convergence Rate)**.** *With the client and server learning rates $\eta_c = \frac{1}{L\tau\sqrt{\tau T}}$, $\eta_s = \sqrt{\tau}$, and step size of `FindWeights` $\beta = \frac{1}{M\sqrt{T}}$, the bound on $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\sum_k w_k^{(t)}\nabla F_k\right\|^2$ in (6) reduces to,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla\mathbf{F}\left(\boldsymbol{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \leq \mathcal{O}\left(\frac{b^2 + C_1^2}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{\sigma_l^2}{\tau\sqrt{T}}\left(\frac{1}{n} + \frac{1}{\tau}\right)\right).$$

G.4 Intermediate Lemmas

We first provide the intermediate lemmas used throughout the proof. The expectation used in this section is expectation conditioned on $\mathbf{x}^{(t)}$ and $\mathbf{w}^{(t)}$. We drop the condition notation for brevity.

Lemma 1 (Linear Algebra Tools)**.** For a matrix $A \in \mathbb{R}^{d\times M}$ with columns $\mathbf{a}_1, \ldots, \mathbf{a}_M$ such that $\|\mathbf{a}_k\| \le b$ for all $k \in [M]$, and a vector $\mathbf{w} \in \mathcal{S}_M$ where $\mathcal{S}_M$ is M-probability simplex, the followings hold.

1. $||A\mathbf{w}|| \le b$. *Proof*: $||A\mathbf{w}|| = ||\sum_k w_k \mathbf{a}_k|| \le \sum_k w_k ||\mathbf{a}_k|| \le \sum_k w_k b = b$.
2. $\|A\|_{\text{op}} \le b\sqrt{M}$. *Proof:* $\|A\|_{\text{op}} \le ||A||_2 = \sqrt{\sum_k ||\mathbf{a}_k||^2} \le \sqrt{\sum_k b^2} = b\sqrt{M}$.
3. $\left\|A^\top A\mathbf{w}\right\| \le \sqrt{M}b^2$. *Proof:* $\left\|A^\top A\mathbf{w}\right\| \le \|A\|_{\text{op}} \|A\mathbf{w}\| \le b^2\sqrt{M}$ using the first two inequalities.

Lemma 2. Suppose Assumptions 2 - 5 hold. Then, the expected value norm difference of averaged compressed stochastic Jacobian matrices across clients and exact Jacobian is bounded. Recall that $Y\left(\mathbf{x}^{(t)}, \zeta_j\right)$ is defined in Algorithm 9 as the averaged compressed stochastic Jacobian estimates from sampled clients.

$$\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2 \le M\left(\frac{1+q}{n'} + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2\right),$$

and,

$$\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\text{op}} \le \sqrt{M}\sqrt{\frac{q+1}{n'}\sigma_l^2 + \frac{qb^2}{n'} + \frac{N-n'}{n'(N-1)}\sigma_g^2}.$$

The proof is presented in Appendix G.6.

Lemma 3. Suppose Assumptions 2 - 5 hold. Then, the expected value norm difference of the weighted average of averaged compressed stochastic gradients across clients and exact gradients is bounded.

$$\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\| \le \sqrt{\frac{q+1}{n'}\sigma_l^2 + \frac{qb^2}{n'} + \frac{N-n'}{n'(N-1)}\sigma_g^2},$$

and,

$$\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|^2 \le \frac{1+q}{n'}\sigma_l^2 + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2.$$

The proof is presented in Appendix G.6.

Lemma 4. Suppose Assumptions 2 - 5 hold. Then, the update on task weights (line 6 in Algorithm 10) is bounded.

$$\mathbb{E}\left\|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\right\| \le \beta\sqrt{M}C_1, \tag{20}$$

where $C_1 \triangleq \left(\sqrt{\frac{1+q}{n'} + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2} + b\right)^2$ is a constant independent of T and M. Also, note that $C_1 \le \mathcal{O}\left(\frac{1+q}{n'} + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2 + b^2\right)$ by AM-GM inequality.

The proof is presented in Appendix G.6.

Lemma 5. Suppose Assumptions 2 - 5 hold. Then we can show the following inequality,

$$\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \le \frac{\mathbb{E}\left\|\mathbf{w}^{(t)} - \mathbf{w}\right\|^2 - \mathbb{E}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}\right\|^2}{2\beta} + \frac{\beta M C_1^2}{2},$$

where $C_1 \triangleq \left(\sqrt{\frac{1+q}{n'} + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2} + b\right)^2$ is a constant independent of T and M.

The proof is presented in Appendix G.6.

Lemma 6. Suppose Assumptions 2 and 4 hold. The aggregated update of clients is bounded in expectation such that,

$$\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \leq b^2 + \frac{\sigma_l^2}{n\tau}. \tag{21}$$

The proof is presented in Appendix G.6.

Lemma 7. Suppose Assumptions 1, 2 and 4 hold, and $\eta_c \leq \frac{1}{2L\tau}$. Then, we can show the following inequality,

$$\mathbb{E}\left[\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] \leq L\eta_c\sqrt{\tau}\sigma_l b + L\eta_c b^2\sqrt{2\tau(\tau-1)}. \tag{22}$$

The proof is presented in Appendix G.6.

G.5 Proofs of Main Statements

We present the proofs of Theorem 1 and Corollary 1.1. We are inspired by some proof techniques in Xiao et al. (2023); Jhunjhunwala et al. (2022); Askin et al. (2024) in lemmas and main proof.

G.5.1 Proof of Theorem 1

The expectation used in this section is expectation conditioned on $\mathbf{x}^{(t)}$ and $\mathbf{w}^{(t)}$. We drop the condition notation for brevity. Let us fix some $\mathbf{w} \in \mathcal{S}_M$. Using the update rule in Algorithm 1 and Assumption 2,

$$\begin{aligned}
\mathbf{F}\left(\mathbf{x}^{(t+1)}\right)\mathbf{w} &\leq \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w} + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\rangle + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&= \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w} + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle \\
&\quad + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right\rangle.
\end{aligned}$$

Then, taking conditional expectation on $\mathbf{x}^{(t)}$ and $\mathbf{w}^{(t)}$,

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{F}\left(\mathbf{x}^{(t+1)}\right)\mathbf{w}\right] - \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w} \leq \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] \\
&\quad + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbb{E}\left[\mathbf{w}^{(t+1)}\right]\right\rangle \\
&= \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&\quad + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbb{E}\left[\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\right]\right\rangle + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \\
&= \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&\quad + \eta_c\eta_s\tau\left\langle \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbb{E}\left[\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\right]\right\rangle \mp \eta_c\eta_s\tau\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2
\end{aligned}$$

$$
\begin{aligned}
&+ \eta_c\eta_s\tau \left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \\
&= \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\right\rangle\right] - \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau \left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \\
&\overset{\substack{\text{Cauch-Schwarz}\\ \text{inequality}}}{\leq} \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}\right\|\mathbb{E}\left\|\mathbf{w}^{(t)} - \mathbf{w}^{(t+1)}\right\| - \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau \left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle.
\end{aligned}
$$

Then using Lemmas 1 and 4, and defining $C_1 \triangleq \left(\sqrt{\frac{1+q}{n'} + \frac{q}{n'}b^2 + \frac{N-n'}{n'(N-1)}\sigma_g^2} + b\right)^2$,

$$
\begin{aligned}
&\mathbb{E}\left[\mathbf{F}\left(\mathbf{x}^{(t+1)}\right)\mathbf{w}\right] - \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w} \leq \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] \\
&+ \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 + \eta_c\eta_s\tau\beta M C_1 b^2 + \eta_c\eta_s\tau \left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \\
&- \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&\overset{\text{Lemma 5}}{\leq} \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau\beta M C_1 b^2 + \eta_c\eta_s\tau\frac{\left\|\mathbf{w}^{(t)} - \mathbf{w}\right\|^2 - \mathbb{E}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}\right\|^2}{2\beta} + \eta_c\eta_s\tau\frac{\beta M C_1^2}{2} - \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&\overset{\substack{\text{Lemma 6 and}\\ \text{assuming } \eta_c\leq\frac{1}{2L\tau}}}{\leq} \eta_c\eta_s\tau\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] - \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau\beta M\left(C_1 b^2 + \frac{C_1^2}{2}\right) + \eta_c\eta_s\tau\frac{\left\|\mathbf{w}^{(t)} - \mathbf{w}\right\|^2 - \mathbb{E}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}\right\|^2}{2\beta} + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\left(b^2 + \frac{\sigma_l^2}{n\tau}\right) \\
&\overset{\text{Lemma 7}}{\leq} \eta_c\eta_s\tau\left(L\eta_c\sqrt{\tau}\sigma_l b + L\eta_c b^2\sqrt{2\tau(\tau-1)}\right) - \eta_c\eta_s\tau\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \\
&+ \eta_c\eta_s\tau\beta M\left(C_1 b^2 + \frac{C_1^2}{2}\right) + \eta_c\eta_s\tau\frac{\left\|\mathbf{w}^{(t)} - \mathbf{w}\right\|^2 - \mathbb{E}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}\right\|^2}{2\beta} + \frac{L\eta_c^2\eta_s^2\tau^2}{2}\left(b^2 + \frac{\sigma_l^2}{n\tau}\right).
\end{aligned}
$$

Arranging the terms, we have

$$
\begin{aligned}
&\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \leq \frac{\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w} - \mathbb{E}\left[\mathbf{F}\left(\mathbf{x}^{(t+1)}\right)\mathbf{w}\right]}{\eta_c\eta_s\tau} + \frac{\left\|\mathbf{w}^{(t)} - \mathbf{w}\right\|^2 - \mathbb{E}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}\right\|^2}{2\beta} + \beta M\left(C_1 b^2 + \frac{C_1^2}{2}\right) \\
&+ L\eta_c\sqrt{\tau}\sigma_l b + L\eta_c b^2\sqrt{2\tau(\tau-1)} + \frac{L\eta_c\eta_s\tau}{2}\left(b^2 + \frac{\sigma_l^2}{n\tau}\right)
\end{aligned}
$$

We choose $\mathbf{w} = \mathbf{w}^{(0)}$. Define $\delta = \sum_{k\in[M]} w_k^{(0)}\left(F_k\left(\mathbf{x}^{(0)}\right) - \min_{\mathbf{x}} F_k\left(\mathbf{x}\right)\right)$ Telescoping the terms through

$t = 0, \dots, T-1$, and taking the unconditional expectation by using the tower property,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \leq \frac{\delta}{T\eta_c\eta_s\tau} + \beta M\left(C_1 b^2 + \frac{C_1^2}{2}\right) + L\eta_c\sqrt{\tau}\sigma_l b + L\eta_c b^2\sqrt{2\tau(\tau-1)} + \frac{L\eta_c\eta_s\tau}{2}\left(b^2 + \frac{\sigma_l^2}{n\tau}\right)$$

$$\leq \underbrace{\mathcal{O}\left(\beta M C_1^2\right)}_{\textit{MOO Weight Error}} + \underbrace{\mathcal{O}\left(\frac{1}{T\eta_s\eta_c\tau} + \frac{L\eta_s\eta_c\sigma_l^2}{n}\right)}_{\substack{\textit{Centralized Optimization Error}\\ \textit{for Scalarized Loss}}} + \underbrace{\mathcal{O}\left(L\eta_s\eta_c\tau b^2\right)}_{\substack{\textit{Partial Participation}\\ \textit{Error}}} + \underbrace{\mathcal{O}\left(L\eta_c(\tau b^2 + \sqrt{\tau}b\sigma_l)\right)}_{\textit{Local Drift Error}}, \tag{23}$$

where $\mathcal{O}(\cdot)$ swallows numerical constant dependencies and δ is the optimization gap of the initial model.

G.5.2 Proof of Corollary 1.1

Let $\mathcal{O}(\cdot)$ swallow smoothness constant L as well for brevity. Now, inserting client and server learning rates $\eta_c = \frac{1}{L\tau\sqrt{\tau T}}$, $\eta_s = \sqrt{\tau}$, and step size of `FindWeights` $\beta = \frac{1}{M\sqrt{T}}$ in Equation (23), we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2 \leq \mathcal{O}\left(\frac{C_1^2}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{1}{\sqrt{T}} + \frac{\sigma_l^2}{\sqrt{T}\tau n}\right) + \mathcal{O}\left(\frac{b^2}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{b^2}{\sqrt{T}\tau} + \frac{b\sigma_l}{\tau\sqrt{T}}\right)$$

$$\overset{\text{by AM-GM inequality}}{=} \mathcal{O}\left(\frac{b^2 + C_1^2}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{\sigma_l^2}{\tau\sqrt{T}}\left(\frac{1}{n} + \frac{1}{\tau}\right)\right).$$

G.6 Proofs of Intermediate Lemmas

Below, we provide the proofs of intermediate lemmas presented in Appendix G.4.

Proof of Lemma 2. To prove Lemma 2, we first show that $\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2$ is bounded, which is the first part of the lemma.

$$\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2$$

$$= \mathbb{E}\left\|\left[\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla}f_{i,1}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right) - \nabla F_1\left(\mathbf{x}^{(t)}\right)\right), \dots, \frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla}f_{i,M}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right) - \nabla F_M\left(\mathbf{x}^{(t)}\right)\right)\right]\right\|^2$$

$$= \sum_{k=1}^{M}\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right) - \nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2 \tag{24}$$

$$= \sum_{k=1}^{M}\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right) \mp \widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right) \mp \nabla f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2$$

$$= \sum_{k=1}^{M}\mathbb{E}\left[\frac{1}{(n')^2}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\left\|\mathcal{Q}\left(\widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right) - \widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right\|^2 + \left\|\widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right)\right\|^2\right)\right]$$

$$+ \sum_{k=1}^{M}\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\nabla f_{i,k}\left(\mathbf{x}^{(t)}\right) - \nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2 \tag{25}$$

$$\leq \sum_{k=1}^{M}\left[\frac{1}{n'N}\sum_{i\in[N]}\left(q\mathbb{E}\left\|\widetilde{\nabla}f_{i,k}\left(\mathbf{x}^{(t)}, \zeta_j\right) \mp \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right\|^2 + \sigma_l^2\right) + \mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\nabla f_{i,k}\left(\mathbf{x}^{(t)}\right) - \nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2\right] \tag{26}$$

$$\leq \sum_{k=1}^{M}\left[\frac{1}{n'N}\sum_{i\in[N]}\left(q\mathbb{E}\left\|\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right\|^2+q\left\|\nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right\|^2+\sigma_l^2\right)+\frac{N-n'}{n'(N-1)}\sigma_g^2\right] \tag{27}$$

$$\leq \sum_{k=1}^{M}\left[\frac{1}{n'N}\sum_{i\in[N]}\left(q\sigma_l^2+qb^2+\sigma_l^2\right)+\frac{N-n'}{n'(N-1)}\sigma_g^2\right] \tag{28}$$

$$\leq M\left(\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2\right), \tag{29}$$

where Equation (25) follows observing that cross-terms will be zero in the expansion since the randomness across compression, stochastic gradients, and client selection is independent. Equation (26) holds by Assumptions 2 and 5. Equation (27) follows the independence of stochastic gradients, Assumption 3, and Lemma 4 by Jhunjhunwala et al. (2022). Equation (28) uses Assumptions 2 and 4. This proves the first part of the lemma. Now,

$$\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\mathrm{op}}\leq\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_2$$
$$=\mathbb{E}\sqrt{\left\|Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2}\overset{\substack{\textit{Jensen's}\\ \textit{Inequality}}}{\leq}\sqrt{\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2}$$
$$\overset{\substack{\text{Using}\\ \text{Equation (29)}}}{\leq}\sqrt{M}\sqrt{\frac{q+1}{n'}\sigma_l^2+\frac{qb^2}{n'}+\frac{N-n'}{n'(N-1)}\sigma_g^2}.$$

Proof of Lemma 3. To prove the lemma using the Assumptions 2 - 5, we first observe that for any $k\in[M]$,

$$\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2\leq\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2, \tag{30}$$

following the same steps after Equation (24) in Lemma 2 without the outermost summation. Then,

$$\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|\overset{\substack{\text{Triangle}\\ \text{inequality}}}{\leq}\sum_{k\in[M]}w_k^{(t)}\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|$$
$$=\sum_{k\in[M]}w_k^{(t)}\mathbb{E}\sqrt{\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2}$$
$$\overset{\substack{\text{Jensen's}\\ \text{inequality}}}{\leq}\sum_{k\in[M]}w_k^{(t)}\sqrt{\mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2}$$
$$\overset{\text{Equation (30)}}{\leq}\sum_{k\in[M]}w_k^{(t)}\sqrt{\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2}$$
$$\overset{\sum_k w_k^{(t)}=1}{=}\sqrt{\frac{q+1}{n'}\sigma_l^2+\frac{qb^2}{n'}+\frac{N-n'}{n'(N-1)}\sigma_g^2}.$$

This proves the first part of the lemma. Then, we prove the second part,

$$\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_j\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|^2=\mathbb{E}\left\|\sum_{k\in[M]}w_k^{(t)}\left(\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right)\right\|^2$$

$$\overset{\substack{\text{Jensen's}\\ \text{inequality}}}{\leq} \sum_{k\in[M]} w_k^{(t)} \mathbb{E}\left\|\frac{1}{n'}\sum_{i\in\mathcal{A}_j^{(t)}}\left(\mathcal{Q}\left(\widetilde{\nabla} f_{i,k}\left(\mathbf{x}^{(t)},\zeta_j\right)\right)-\nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\|^2$$

$$\overset{\text{Equation (30)}}{\leq} \sum_{k\in[M]} w_k^{(t)}\left(\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2\right) \overset{\sum_k w_k^{(t)}=1}{=} \frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2.$$

Proof of Lemma 4.

$$\mathbb{E}\left\|\mathbf{w}^{(t)}-\mathbf{w}^{(t+1)}\right\| = \mathbb{E}\left\|\mathrm{proj}_{\mathcal{S}_M}\mathbf{w}^{(t)}-\mathrm{proj}_{\mathcal{S}_M}\left(\mathbf{w}^{(t)}-\beta Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right)\right\|$$

$$\leq \mathbb{E}\left\|\mathbf{w}^{(t)}-\left(\mathbf{w}^{(t)}-\beta Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right)\right\| \tag{31}$$

$$= \beta\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|$$

$$= \beta\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_1\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)^\top\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|$$

$$\overset{\substack{\text{Triangle}\\ \text{inequality}}}{\leq} \beta\mathbb{E}\Bigg[\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_1\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)^\top\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|+\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|$$

$$+\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_1\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)^\top\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|+\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|\Bigg]$$

$$\overset{\substack{\text{Cauchy-Schwarz}\\ \text{inequality}}}{\leq} \beta\Bigg[\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\mathrm{op}}\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|$$

$$+\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\mathrm{op}}\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|+\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\mathrm{op}}\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|$$

$$+\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|_{\mathrm{op}}\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|$$

$$\overset{\substack{\textit{Lemma 1}\\ \textit{Lemma 2}\\ \textit{Lemma 3}}}{\leq} \beta\sqrt{M}\left(\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2+2b\sqrt{\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2}+b^2\right)$$

$$= \beta\sqrt{M}\left(\sqrt{\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2}+b\right)^2 = \beta\sqrt{M}C_1.$$

where Equation (31) follows the contraction property of the projection onto the convex sets, e.g., $\mathcal{S}_M$, and $C_1 \triangleq \left(\sqrt{\frac{1+q}{n'}\sigma_l^2+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2}+b\right)^2$.

Proof of Lemma 5. To prove the lemma, we first show $\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2$ is bounded.

$$\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2 = \mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_1\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)^\top\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|^2$$

$$\overset{\substack{\text{Cauchy-Schwarz and}\\ \text{independence}\\ \text{of } \zeta_1 \text{ and } \zeta_2}}{\leq} \mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mp\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|^2$$

$$= \left(\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2+\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right\|^2\right)\left(\mathbb{E}\left\|\left(Y\left(\mathbf{x}^{(t)},\zeta_2\right)-\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\right)\mathbf{w}^{(t)}\right\|^2 \right. \tag{32}$$

$$\left. +\mathbb{E}\left\|\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\|^2\right)$$

$$\overset{\substack{\textit{Lemma 1}\\ \textit{Lemma 2}\\ \textit{Lemma 3}}}{\leq} M\left(\frac{q+1}{n'}\sigma_l^2+\frac{qb^2}{n'}+\frac{N-n'}{n'(N-1)}\sigma_g^2+b^2\right)^2 \leq MC_1^2, \tag{33}$$

where $C_1 \triangleq \left(\sqrt{\frac{1+q}{n'}+\frac{q}{n'}b^2+\frac{N-n'}{n'(N-1)}\sigma_g^2}+b\right)^2$. Then,

$$\mathbb{E}\left\|\mathbf{w}^{(t+1)}-\mathbf{w}\right\|^2 \leq \mathbb{E}\left\|\operatorname{proj}_{\mathcal{S}_M}\left(\mathbf{w}^{(t)}-\beta Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right)-\operatorname{proj}_{\mathcal{S}_M}\mathbf{w}\right\|^2$$

$$\overset{\substack{\text{Contraction}\\ \text{property of}\\ \text{projection}}}{\leq} \mathbb{E}\left\|\mathbf{w}^{(t)}-\mathbf{w}-\beta Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2$$

$$=\mathbb{E}\left\|\mathbf{w}^{(t)}-\mathbf{w}\right\|^2-2\beta\mathbb{E}\left\langle\mathbf{w}^{(t)}-\mathbf{w},Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\rangle+\beta^2\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2$$

$$\overset{\substack{\text{Using the in-}\\ \text{dependence}\\ \text{of } \zeta_1 \text{ and } \zeta_2}}{=} \left\|\mathbf{w}^{(t)}-\mathbf{w}\right\|^2-2\beta\left\langle\mathbf{w}^{(t)}-\mathbf{w},\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)^\top\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle+\beta^2\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2$$

$$=\left\|\mathbf{w}^{(t)}-\mathbf{w}\right\|^2-2\beta\left\langle\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\left(\mathbf{w}^{(t)}-\mathbf{w}\right),\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle+\beta^2\mathbb{E}\left\|Y\left(\mathbf{x}^{(t)},\zeta_1\right)^\top Y\left(\mathbf{x}^{(t)},\zeta_2\right)\mathbf{w}^{(t)}\right\|^2$$

$$\overset{\textit{Equation } (33)}{\leq} \left\|\mathbf{w}^{(t)}-\mathbf{w}\right\|^2-2\beta\left\langle\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\left(\mathbf{w}^{(t)}-\mathbf{w}\right),\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle+\beta^2MC_1^2.$$

Then, arranging the terms, we get

$$\left\langle\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\left(\mathbf{w}^{(t)}-\mathbf{w}\right),\nabla\mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t)}\right\rangle \leq \frac{\mathbb{E}\left\|\mathbf{w}^{(t)}-\mathbf{w}\right\|^2-\mathbb{E}\left\|\mathbf{w}^{(t+1)}-\mathbf{w}\right\|^2}{2\beta}+\frac{\beta MC_1^2}{2}.$$

Proof of Lemma 6.

$$\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2=\mathbb{E}\left[\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)}\right\|^2 \mid \mathbf{w}^{(t+1)}\right]\right]$$

$$=\mathbb{E}\left[\mathbb{E}\left[\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\sum_{k\in[M]}w_k^{(t+1)}\widetilde{\nabla}f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\right\|^2 \mid \mathbf{w}^{(t+1)}\right]\right]$$

$$\overset{\substack{\text{Jensen's}\\ \text{inequality}}}{\leq} \mathbb{E}\left[\mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\widetilde{\nabla}f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\right\|^2 \mid \mathbf{w}^{(t+1)}\right]\right]$$

$$=\mathbb{E}\left[\mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\widetilde{\nabla}f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\mp\nabla f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\right\|^2 \mid \mathbf{w}^{(t+1)}\right]\right]$$

$$\overset{\substack{\text{Independence}\\ \text{of SGD and}\\ \text{Assump. 2}}}{\leq} \mathbb{E}\left[\mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\nabla f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\right\|^2 \mid \mathbf{w}^{(t+1)}\right]\right]+\mathbb{E}\left[\mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\frac{\sigma_l^2}{n\tau} \mid \mathbf{w}^{(t+1)}\right]\right]$$

$$\overset{\substack{\text{Jensen's}\\ \text{inequality}}}{\leq} \mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\left\|\nabla f_{i,k}\left(\mathbf{x}_i^{(t,k)}\right)\right\|^2\right]+\frac{\sigma_l^2}{n\tau}$$

$$\overset{\text{Assump. 4}}{\leq} \mathbb{E}\left[\sum_{k\in[M]}w_k^{(t+1)}\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\frac{1}{\tau}\sum_{r=0}^{\tau-1}b^2\right]+\frac{\sigma_l^2}{n\tau}=b^2++\frac{\sigma_l^2}{n\tau}.$$

Proof of Lemma 7.

$$
\begin{aligned}
&\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{n}\sum_{i\in\mathcal{B}^{(t)}}\Delta_i^{(t)} - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle \mid \mathbf{w}^{(t+1)}\right]\right] \qquad (34)\\
&\overset{\substack{\text{Expectation}\\ \text{over SGD}\\ \text{and } \mathcal{B}^{(t)}}}{=} \mathbb{E}\left[\mathbb{E}\left[\left\langle \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}, -\left(\frac{1}{N}\sum_{i\in[N]}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\sum_{k\in[M]} w_k^{(t+1)}\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}^{(t+1)}\right)\right\rangle \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\overset{\text{Cauchy-Schwarz}}{\leq} \mathbb{E}\left[\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}\right\| \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i\in[N]}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\sum_{k\in[M]} w_k^{(t+1)}\left(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla F_k\left(\mathbf{x}^{(t)}\right)\right)\right\| \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\overset{F_k=\frac{1}{N}\sum_i f_{i,k}}{=} \mathbb{E}\left[\left\|\nabla \mathbf{F}\left(\mathbf{x}^{(t)}\right)\mathbf{w}\right\| \mathbb{E}\left[\left\|\frac{1}{N}\sum_{i\in[N]}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\sum_{k\in[M]} w_k^{(t+1)}\left(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right)\right\| \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\overset{\substack{\text{Assump. 4}\\ \text{Lemma 1}}}{\leq} b\mathbb{E}\left[\mathbb{E}\left[\left\|\frac{1}{N}\sum_{i\in[N]}\frac{1}{\tau}\sum_{r=0}^{\tau-1}\sum_{k\in[M]} w_k^{(t+1)}\left(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right)\right\| \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\overset{\substack{\text{Triangular}\\ \text{inequality}}}{\leq} b\mathbb{E}\left[\mathbb{E}\left[\frac{1}{N}\sum_{i\in[N]}\sum_{k\in[M]} w_k^{(t+1)}\left\|\frac{1}{\tau}\sum_{r=0}^{\tau-1}(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right))\right\| \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\leq b\mathbb{E}\left[\frac{1}{N}\sum_{i\in[N]}\sum_{k\in[M]} w_k^{(t+1)}\mathbb{E}\left[\sqrt{\left\|\frac{1}{\tau}\sum_{r=0}^{\tau-1}(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right))\right\|^2} \mid \mathbf{w}^{(t+1)}\right]\right]\\
&\overset{\substack{\text{Jensen's}\\ \text{inequality}}}{\leq} b\mathbb{E}\left[\frac{1}{N}\sum_{i\in[N]}\sum_{k\in[M]} w_k^{(t+1)}\sqrt{\mathbb{E}\left[\left\|\frac{1}{\tau}\sum_{r=0}^{\tau-1}(\nabla f_{i,k}\left(\mathbf{x}^{(t,r)}\right) - \nabla f_{i,k}\left(\mathbf{x}^{(t)}\right))\right\|^2 \mid \mathbf{w}^{(t+1)}\right]}\right]\\
&\overset{\substack{\text{Lemma 2 in}\\ \text{Askin et al. (2024)}}}{\leq} b\mathbb{E}\left[\frac{1}{N}\sum_{i\in[N]}\sum_{k\in[M]} w_k^{(t+1)}\sqrt{\frac{L^2\eta_c^2\tau}{2(1-L^2\eta_c^2\tau(\tau-1))}\sigma_l^2 + \frac{L^2\eta_c^2\tau(\tau-1))}{1-L^2\eta_c^2\tau(\tau-1))}\left\|\nabla f_{i,k}\left(\mathbf{x}^{(t)}\right)\right\|^2}\right]\\
&\overset{\text{Assump. 4}}{\leq} b\mathbb{E}\left[\frac{1}{N}\sum_{i\in[N]}\sum_{k\in[M]} w_k^{(t+1)}\sqrt{\frac{L^2\eta_c^2\tau}{2(1-L^2\eta_c^2\tau(\tau-1))}\sigma_l^2 + \frac{L^2\eta_c^2\tau(\tau-1)}{1-L^2\eta_c^2\tau(\tau-1)}b^2}\right]\\
&= b\sqrt{\frac{L^2\eta_c^2\tau}{2(1-L^2\eta_c^2\tau(\tau-1))}\sigma_l^2 + \frac{L^2\eta_c^2\tau(\tau-1)}{1-L^2\eta_c^2\tau(\tau-1)}b^2}\\
&\overset{\substack{\eta_c\leq\frac{1}{2L\tau}\\ \text{(assumption}\\ \text{of the lemma)}}}{\leq} b\sqrt{L^2\eta_c^2\tau\sigma_l^2 + 2L^2\eta_c^2\tau(\tau-1)b^2} \leq L\eta_c\sqrt{\tau}\sigma_l b + L\eta_c b^2\sqrt{2\tau(\tau-1)}.
\end{aligned}
$$