DFedReweighting: A Unified Framework for Objective-Oriented Reweighting in Decentralized Federated Learning

Kaichuang Zhang[1], Wei Yin[2], Jinghao Yang[3], Ping Xu[4]
kaichuangzhang@usf.edu, wei.yin01@utrgv.edu, jinghao.yang@utrgv.edu, pxu4@gsu.edu
[1]Department of Electrical Engineering, University of South Florida, Tampa, Florida, USA
[2]Department of Mathematics, The University of Texas Rio Grande Valley, Edinburg, Texas, USA
[3]Department of Electrical and Computer Engineering, The University of Texas Rio Grande Valley, Edinburg, Texas, USA
[4]Department of Computer Science, Georgia State University, Atlanta, Georgia, USA

***Abstract*—Decentralized federated learning (DFL) has emerged as a promising paradigm that enables multiple clients to collaboratively train machine learning models through iterative rounds of local training, communication, and aggregation, without relying on a central server. Nevertheless, DFL systems continue to face a range of challenges, including fairness and Byantine robustness. To address these challenges, we propose DFedReweighting, a unified aggregation framework that achieves diverse learning objectives in DFL via objective-oriented reweighting at the final step of each learning round. Specifically, for each client, the framework first evaluates a target performance metric (TPM) on a compact auxiliary dataset constructed from local data, yielding preliminary aggregation weights, which are subsequently refined by a customized reweighting strategy (CRS) to produce the final aggregation weights. Theoretically, we prove that an appropriate TPM-CRS combination guarantees linear convergence for general L-smoothand strongly convex functions. Empirical results consistently demonstrate that DFedReweighting significantly improves fairness and robustness against Byzantine attacks across diverse settings. Two multi-objective examples, spanning tasks across and within clients, further establish that a broad range of desired learning objectives can be accommodated by appropriately designing the TPM and CRS. Our code is available at https://github.com/KaichuangZhang/DFedReweighting.**

***Index Terms*—Decentralized federated learning, Fairness, Robustness**

I. Introduction

In federated learning (FL), multiple clients collaboratively train a machine learning model by performing local training and synchronizing their updates (gradients or model parameters) with a central server. Consequently, clients maintain their raw data locally, thereby preserving data privacy [1], [2]. In contrast, decentralized federated learning (DFL), where each client independently executes both local model updates and aggregation without relying on a central server, has recently gained attention [3], [4]. A typical DFL system operates iteratively through a three-stage process comprising local training, communication, and aggregation [5], [6]. This process repeats over multiple rounds until convergence, ultimately ensuring that each client obtains an effective local model that can be deployed in real-world scenarios, such as the Internet of Things [7], healthcare [8], etc.

This work was partly supported by the National Science Foundation (Grants #2112650, #2434916). Corresponding author: Ping Xu (email: pxu4@gsu.edu).

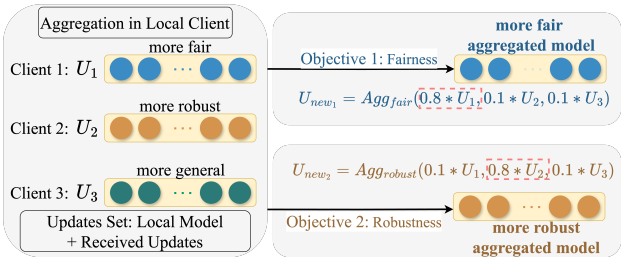

Fig. 1: *Objective-oriented reweighting aggregation.* Each client assigns aggregation weights according to its local objective. For instance, a client seeking to improve fairness may assign a higher weight to update U_1 if it yields superior fairness performance. More generally, each client allocates weights to received updates (U_1, U_2, U_3) through the reweighting aggregation function $Agg(\,)$, enabling diverse objectives to be pursued simultaneously across clients.

Although DFL mitigates issues inherent to centralized FL, such as single point of failure [9] and communication bottlenecks [10], other critical challenges, e.g., robustness, fairness, communication efficiency, etc., persist in DFL [11]. Addressing these challenges typically involves developing dedicated algorithms at any stage of the three-stage (i.e., local computation, communication, local aggregation) process in DFL. However, we argue that the final aggregation stage, characterized by target performance metrics (TPMs) and customized reweighting strategies (CRSs), is sufficient to resolve these issues. In essence, numerous DFL methods ultimately take effect at the final aggregation stage regardless of the specific problems they aim to address. For example, `q-FFL` [12] promotes fairness by modifying the global objective function, a change that directly affects local updates while implicitly

inducing reweighted aggregation. Similarly, FedFV [13] alleviates gradient conflicts through iterative adjustments to both the direction and magnitude of gradients, which constitutes another indirect form of reweighted aggregation.

In contrast to these inherently indirect methods, directly applying objective-oriented reweighting on updates (as shown in Fig. 1) at the final aggregation stage provides a more generalizable and transparent approach to steer the learning process. Rather than relying on implicit or heuristic modifications, this approach explicitly aligns aggregation with the underlying optimization objective — intuitively, when heterogeneous updates are available, effective learning demands prioritizing those most relevant to the target objective. The objective-oriented reweighting thus provides a principled aggregation strategy for DFL systems to distinguish useful received updates from noisy or irrelevant ones, since these updates may differ substantially due to data heterogeneity, varying objectives, or unreliable participants. Specifically, objective-oriented reweighting enables each client to treat received updates as heterogeneous sources, selectively emphasizing those that best support its local objective before integration. The resulting aggregation scheme balances adaptability and robustness while preserving the decentralized nature of the system.

The contributions of this work are summarized as follows:

- We propose **DFedReweighting**, a unified aggregation framework that addresses a broad range of challenges in DFL via objective-oriented reweighting at the final step of each learning round.
- We provide a theoretical convergence analysis establishing that any appropriate TPM-CRS combination guarantees linear convergence for general L-smooth and strongly convex objective functions.
- We conduct extensive experiments demonstrating that **DFedReweighting** consistently achieves superior performance on client fairness and Byzantine robustness across diverse settings. Additionally, two multi-objective examples illustrate the flexibility of the TPM-CRS design paradigm in accommodating a wider range of learning objectives.

II. Related Work

This section reviews representative works addressing key challenges in DFL systems across different stages of the learning process. Given the structural similarities between FL and DFL, client-side methods originally developed for FL are also discussed where relevant.

Local Training Stage: Some studies pursue specific objectives, notably fairness, by modifying the local objective function or altering the training workflow. AFL [14] promotes fairness by minimizing the worst-case loss across all clients. FedMGDA+ [15] enhances both fairness and robustness through multi-objective optimization over clients' local losses. GIFAIR-FL [16] introduces a regularization term that penalizes dispersion in per-group losses, steering optimization toward fairer solutions. FairFed [17] enforces per-group loss upper bounds via server-side gradient reweighting, improving global fairness without sacrificing accuracy. AgnosticFair [18] assigns sample-level reweighting values through kernel functions applied to both the loss and the fairness constraint.

Communication Stage: A separate line of work targets communication efficiency. [19] propose structured and sketched updates to reduce uplink communication costs by up to two orders of magnitude. [20] leverage lossy compression and dropout to further reduce communication overhead. FedKD [21] achieves communication efficiency through adaptive mutual knowledge distillation combined with dynamic gradient compression. [22] improve efficiency by selectively soliciting updates only from clients with informative gradients while estimating the remainder. [23] investigate structured sparsification techniques to construct compact models with substantially reduced storage and bandwidth requirements.

Aggregation Stage: A third category of methods pursues their objectives at the aggregation stage. For fairness, pFedFair [24] identifies under performing clients, assigns them higher fairness coefficients during aggregation, and subsequently personalizes local fine-tuning, achieving a Pareto-optimal fairness-accuracy trade-off. For robustness, [25] propose a spectral anomaly detection model based on a variational autoencoder with dynamic thresholds, deployed at the central server to identify and suppress malicious updates. FLTrust [26] computes per-worker trust scores using a curated root dataset at the server to filter unreliable updates before aggregation. Siren [27] combines a client-side alarming mechanism with a server-side root-dataset-based defense to identify malicious participants. ByGARS [28] derives reputation scores for each worker using a server-held auxiliary dataset and incorporates them into adaptive gradient aggregation.

III. Problem Statement

A DFL system can be modeled as a decentralized network of N clients interconnected over a fixed undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ denotes the client set and $\mathcal{E}$ denotes the edge set [5]. Any two clients n and m are said to be one-hop neighbors if $(n, m) \in \mathcal{E}$, enabling them to exchange information during communication steps. For client k, its one-hop neighbors are in the set $\mathcal{N}_k = \{m|(m, k) \in \mathcal{E}\}$. Under the assumption that all clients work properly, the learning task of a DFL system is to minimize the weighted average of local clients' losses, given by

$$\min_{\mathbf{W}} F(\mathbf{w}) = \sum_{k=1}^{N} p_k f_k(\mathbf{w}_k, \mathcal{D}_k), \tag{1}$$

where $\mathbf{W} := \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$ denotes the set of all local models, for which a consensus constraint may or may not be imposed depending on the task. $\mathcal{D}_k$ represents client k's local data, and $\mathcal{D} = \bigcup_{k\in\mathcal{N}} \mathcal{D}_k$. p_k is the aggregation weight assigned to client k, which is set to be $p_k = \frac{|\mathcal{D}_k|}{|\mathcal{D}|}$ in DFedAvg [29], with $|\mathcal{D}_k|$ representing the number of local data samples owned by client k.

Various popular decentralized algorithms, such as decentralized stochastic gradient descent and the decentralized al-

ternating direction method of multipliers, can be employed to solve (1). Taking decentralized SGD as an example, at each communication round t, each client k ($k \in \mathcal{N}$) executes the following three-stage learning process multiple times until convergence [30], [31].

Stage 1 (local training): client k takes a local SGD step to obtain an intermediate update $\mathbf{w}_k^{t+\frac{1}{2}}$:

$$\mathbf{w}_k^{t+\frac{1}{2}} = \mathbf{w}_k^t - \eta^t \nabla f_k(\mathbf{w}_k^t; \xi_k^t), \tag{2}$$

where $\mathbf{w}_k^{t+\frac{1}{2}}$ is the locally trained model at iteration t, η^t is the learning rate, and $\nabla f_k(\mathbf{w}_k^t; \xi_k^t)$ represents the stochastic gradient of the local objective function f_k for client k, evaluated using the model parameter $\mathbf{w}_k^t$ on a mini-batch sample ξ_k^t drawn from the client's local dataset $\mathcal{D}_k$.

Stage 2 (communication): client k sends $\mathbf{w}_k^{t+\frac{1}{2}}$ to its neighbors m ($m \in \mathcal{N}_k$) and receives $\mathbf{w}_m^{t+\frac{1}{2}}$ from its neighbors, resulting in a local update set $\mathbf{W}_k^t$ that includes client k's local model and updates from its neighbors:

$$\mathbf{W}_k^t := \{\mathbf{w}_k^{t+\frac{1}{2}}\} \cup \{\mathbf{w}_m^{t+\frac{1}{2}} | m \in \mathcal{N}_k\}. \tag{3}$$

Stage 3 (aggregation): client k follows (4) to get a new local model [29]:

$$\mathbf{w}_k^{t+1} = \sum_{\mathbf{w}_i^{t+\frac{1}{2}} \in \mathbf{W}_k^t} p_i \mathbf{w}_i^{t+\frac{1}{2}}, \tag{4}$$

where $\mathbf{w}_k^{t+1}$ is the new local model for client k at round $t+1$.

The DFL system repeats this 3-stage learning process until all local models converge, at which point they can be deployed for practical use.

IV. DFedReweighting

We propose **DFedReweighting**, a unified aggregation framework (Algorithm 1) to address key challenges in DFL systems through objective-oriented reweighting at the final stage of each learning round. The core insight is that each client can steer its own learning behavior by assigning aggregation weights that reflect its local objective, rather than relying on default, objective-agnostic schemes. Specifically, for each client kk k, the framework first evaluates a *target performance metric* (TPM) on a compact auxiliary dataset constructed from local data, yielding preliminary aggregation weights. These weights are subsequently refined by a *customized reweighting strategy* (CRS) to produce the final aggregation weights. The framework proceeds in four steps:

Step 1 (Construct Auxiliary Dataset $\mathcal{A}_k$) Unlike conventional FL, where a central server aggregates client updates without access to any local data [32], DFL enables each client to validate received updates against its own local training data, a capability that is fundamental to the objective-oriented reweighting framework proposed in this work. Nevertheless, utilizing the entire local dataset for this validation process introduces unnecessary computational overhead. To address this, we construct a compact auxiliary dataset $\mathcal{A}_k$ by *subsampling* a representative subset of the local training data, which is sufficient to serve as the basis for update evaluation while remaining computationally efficient.

Algorithm 1 DFedReweighting runs at client k.

Input: $\mathbf{W}_k^t = \{\mathbf{w}_k^{t+\frac{1}{2}}\} \cup \{\mathbf{w}_m^{t+\frac{1}{2}} \mid m \in \mathcal{N}_k\}$, local training dataset $\mathcal{D}_k$, sampling ratio α, reweighting strategy s.
Output: $\mathbf{w}_k^{t+1}$

1: Subsample a fraction α of $\mathcal{D}_k$ to form auxiliary dataset $\mathcal{A}_k$.
2: **for** each $\mathbf{w}_i^{t+\frac{1}{2}} \in \mathbf{W}_k^t$ **do**
3: Calculate the performance score m_i for update $\mathbf{w}_k^{t+\frac{1}{2}}$ using $\mathcal{A}_k$ to form M_k.
4: **end for**
5: **for** each $m_i \in M_k$ **do**
6: Compute the corresponding final aggregation weight $v_i^t = s(m_i, M_k)$ based on performance score m_o and reweighting strategy s to form V_k.
7: **end for**
8: Update local model: $\mathbf{w}_k^{t+1} = \sum_{\mathbf{w}_i^{t+\frac{1}{2}} \in \mathbf{W}_k^t} v_i^t \mathbf{w}_i^{t+\frac{1}{2}}$.
9: **Return** $\mathbf{w}_k^{t+1}$.

Step 2 (Obtain Preliminary Weights M_k). A key advantage of DFL is that each client retains full autonomy over its local aggregation process, enabling the incorporation of objective-aware criteria directly into the update selection mechanism. Leveraging this flexibility, the proposed framework computes preliminary aggregation weights based on a TPM evaluated on the auxiliary dataset $\mathcal{A}_k$. Specifically, upon constructing $\mathcal{A}_k$in Step 1, client k evaluates the TPM against all received model updates $\mathbf{W}_k^t$ as well as its own locally trained model, producing a performance score set M_k. The TPM can be instantiated as accuracy, loss, or any other objective-specific measure, providing the framework with the generality to accommodate diverse learning goals.

Step 3 (Calculate Final Aggregation Weights V_k). Reweighting-based aggregation offers a principled solution, enabling each client in a DFL system to selectively prioritize updates from neighboring peers according to its own objective, rather than relying on default weights derived from dataset sizes. Specifically, after obtaining M_k the framework applies a CRS $s(m_i, M_k)$ to produce a new weight set V_k tailored to the client's goal — for instance, upweighting the local model to promote personalization, or assigning near-zero weights to suspected Byzantine participants to enhance robustness. The flexibility of this design lies in the choice of s by appropriately pairing the TPM with a suitable CRS, the framework can accommodate a broad range of learning objectives, as demonstrated through concrete TPM-CRS design examples in the Experiments section.

Step 4 (Aggregation.) After obtaining final aggregation weights V_k from Step 3, each client performs aggregation using

$$\mathbf{w}_k^{t+1} = \sum_{\mathbf{w}_i^{t+\frac{1}{2}} \in \mathbf{W}_k^t} v_i^t \mathbf{w}_i^{t+\frac{1}{2}}. \tag{5}$$

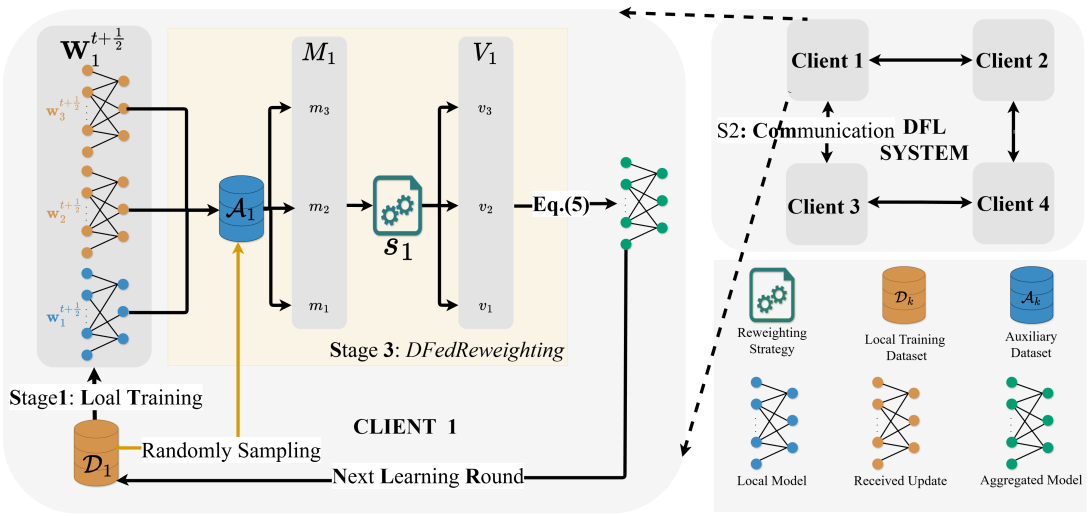

Fig. 2: *Overview of* ***DFedReweighting*** *in a four-client DFL system (each client has two neighbors).* Each client proceeds in four steps: (i) constructs an auxiliary dataset $\mathcal{A}_k$ from local data; (ii) evaluate the TPM on $\mathcal{A}_k$ to obtain a preliminary performance score set M_k; (iii) apply the CRS to M_k to derive the final aggregation weight set V_k; and (iv) aggregates models using V_k.

Fig. 2 shows an example of DFedReweighting in a ring-structured DFL system with four clients. All clients follow a three-stage learning process.

V. Convergence Analysis

Building upon the definitions and notations introduced in Section III, we analyze the convergence properties of **DFedReweighting**. Since each client independently maintains its own local model in DFL, our analysis focuses on the convergence behavior of each client's local model $\mathbf{w}_k$. The analysis adopts the following standard assumptions.

Assumption 1: L-smoothness [31], [33], [34]. Each local objective function f_k has L-Lipschitz continuous gradients, i.e., for all $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$, $\|\nabla f_k(\mathbf{w}) - \nabla f_k(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$.

Assumption 2: Strong convexity [35], [36]. All local objective functions $\{f_k, k \in \mathcal{N}\}$ are strongly convex.

Additional standard assumptions, including bounded second moment [31], [33], [34], connected graph $\mathcal{G}_\mathcal{B}$ [30], [37], and unbiased stochastic gradients with bounded variance [34], are adopted without restatement.

In **DFedReweighting**, the aggregation weights satisfy $v_i^t \geq 0$ for any TPM-CRS combination, as defined in (5). For any pair $(k, i) \notin \mathcal{E}$, i.e., when no direct connection exists between clients k and i, we set $v_i^t = 0$.

Our analysis begins by applying the descent lemma [35], [36] for L-smooth, strongly convex functions:

$$\mathbb{E}\left[F(\mathbf{w}_k^{t+1})\right] - F(\mathbf{w}_k^*) \leq \frac{L}{2} \cdot \mathbb{E}\left[\|\mathbf{w}_k^{t+1} - \mathbf{w}_k^*\|^2\right], \tag{6}$$

where $\mathbf{w}_k^*$ is a deterministic optimal point for client k.

Substituting the aggregation rule (5) and the local update rule (2) into (6), the squared error decomposes into three terms:

$$\begin{aligned}\|\mathbf{w}_k^{t+1} - \mathbf{w}_k^*\|^2 &= \Big\|\sum_{i=1}^N v_i^t(\mathbf{w}_i^t - \mathbf{w}_k^*) - \eta^t \sum_{i=1}^N v_i^t \nabla f_i(\mathbf{w}_i^t; \xi_i^t)\Big\|^2 \\ &= \underbrace{\Big\|\sum_{i=1}^N v_i^t(\mathbf{w}_i^t - \mathbf{w}_k^*)\Big\|^2}_{\text{Term 1}} \\ &\quad + \underbrace{(\eta^t)^2\Big\|\sum_{i=1}^N v_i^t \nabla f_i(\mathbf{w}_i^t; \xi_i^t)\Big\|^2}_{\text{Term 2}} \\ &\quad - \underbrace{2\eta^t\Big\langle\sum_{i=1}^N v_i^t(\mathbf{w}_i^t - \mathbf{w}_k^*), \sum_{i=1}^N v_i^t \nabla f_i(\mathbf{w}_i^t; \xi_i^t)\Big\rangle}_{\text{Term 3}}.\end{aligned} \tag{7}$$

Each term is bounded as follows. For Term 1, by Jensen's inequality:

$$\Big\|\sum_{i=1}^N v_i^t(\mathbf{w}_i^t - \mathbf{w}_k^*)\Big\|^2 \leq \sum_i v_i^t \|\mathbf{w}_i^t - \mathbf{w}_k^*\|^2. \tag{8}$$

For Term 2, using the bounded gradient assumption:

$$(\eta^t)^2\Big\|\sum_{i=1}^N v_i^t \nabla f_i(\mathbf{w}_i^t; \xi_i^t)\Big\|^2 \leq (\eta^t)^2 G^2 \Rightarrow (\eta^t \cdot G)^2. \tag{9}$$

For Term 3:

$$\left\langle \sum_{i=1}^{N} v_i^t(\mathbf{w}_i^t - \mathbf{w}_k^*), \sum_{i=1}^{N} v_i^t \nabla f_i(\mathbf{w}_i^t; \xi_i^t) \right\rangle \leq \frac{3L}{2} \sum_{i=1}^{N} v_i^t \|\mathbf{w}_i^t - \mathbf{w}_k^*\|^2. \tag{10}$$

Combining these bounds establishes the following convergence theorem.

Theorem 1: For any L-smooth, strongly convex local objective function at client k, **DFedReweighting** with a dynamic learning rate $\eta^t < 1$ satisfies the following recursive bound:

$$\|\mathbf{w}_k^{t+1} - \mathbf{w}_k^*\|^2 \leq \Big(\prod_{h=0}^{t}\big(1 - 3\eta^h L\big)\Big)\Big(\sum_{i=1}^{N} v_i^0 \|\mathbf{w}_i^0 - \mathbf{w}_k^*\|^2\Big) + \sum_{j=0}^{t}\Big(\prod_{h=j+1}^{t}\big(1 - 3\eta^h L\big)\Big)\big(\eta^j G\big)^2. \tag{11}$$

In particular, setting $\eta^t < 1/(3L)$ at all rounds ensures that the first term contracts geometrically, and the expected squared error converges linearly to a neighborhood of order $\eta G^2/L$.

VI. Experiments

To validate the effectiveness and broad applicability of **DFedReweighting**, we conduct simulation experiments targeting two representative challenges in DFL: client fairness and Byzantine robustness. Beyond these primary evaluations, we further demonstrate the framework's versatility through multi-objective tasks, both across and within clients, illustrating how the TPM-CRS design can be readily adapted to address a wider range of learning objectives in DFL.

A. Experimental Settings

We begin by summarizing the key experimental settings adopted throughout this work.

Network Topologies. We model the communication network as a random Erdős–Rényi graph with $N = |\mathcal{N}|$ nodes and a connection probability ρ. For Byzantine attack scenarios, following [5], [38], data samples are allocated exclusively to benign nodes while malicious nodes hold no data, with $|\mathcal{N}| = |\mathcal{B}| + |\mathcal{M}|$ denoting the total, benign, and malicious node counts, respectively. Unless otherwise specified, we set $|\mathcal{B}| = 10$, $|\mathcal{M}| = 2$, and $\rho = 0.7$.

Data Heterogeneity (DH). Data heterogeneity is a critical factor in DFL, spanning a spectrum from IID to non-IID distributions. In the IID setting, each client's local dataset contains all classes with equal sample counts. For non-IID settings, we adopt two strategies: LabelSkew (h) [39], which assigns h classes to each node with equal samples per class, and Diri (α) [40], which allocates samples across clients according to a Dirichlet distribution parameterized by α.

Evaluation Metric. For classification tasks, we report the average test accuracy across all clients, $\mathrm{Acc} = \frac{1}{N}\sum_{k=1}^{N} \mathrm{Acc}_k$, where Acc_k is the test accuracy of client k on its local or global test dataset, depending on the task. All results are averaged over four runs with different random seeds ($seed = 43, 44, 45, 46$).

B. Experimental Results

We conduct two sets of experiments to evaluate the proposed framework on client fairness and Byzantine robustness, followed by two illustrative examples demonstrating how to design TPM-CRS combinations for specific objectives.

*1) **Client Fairness**:* Client fairness in DFL aims to ensure comparable model performance across clients on their respective local test datasets. We quantify fairness using the variance of test accuracies across all N clients, defined as $\mathrm{Var} = \frac{1}{N}\sum_{k=1}^{N}\big(\mathrm{Acc}_k - \overline{\mathrm{Acc}}\big)^2$, where Acc_k is the local test accuracy of client k and $\overline{\mathrm{Acc}}$ is the mean accuracy across clients. All clients train a softmax regression model with cross-entropy loss over $T = 3000$ rounds on MNIST and Fashion-MNIST, with a fixed learning rate 0.01 a minibatch size of 32. We compare against FedAvg [1] and q-FFL [12] adapted to the DFL setting, referred to as DFedAvg and q-FDFL, respectively.

Since this is an image classification task, the local test accuracy serves as the TPM for client k. To promote fairness, the CRS adopts a temperature-scaled softmax reweighting strategy:

$$\begin{cases} m_k = \mathrm{Acc}_k \\ s(m_i, M_k) = \frac{e^{m_i/\mathcal{T}}}{\sum_{m_j \in M_k} e^{m_j/\mathcal{T}}},\ i \in \mathcal{N}_k \end{cases}, \tag{12}$$

where a smaller temperature $\mathcal{T}$ yields a sharper, more concentrated weight distribution, while a larger $\mathcal{T}$ produces a softer, more uniform one.

Results. As shown in Table I, under IID conditions, **DFedReweighting** achieves fairness comparable to the baselines while maintaining higher accuracy. Under non-IID conditions, it attains the highest accuracy while reducing variance to approximately 1% to 50% of that observed in the baselines, depending on the specific scenario.

*2) **Byzantine Attack Robustness**:* Byzantine attack robustness in DFL aims to eliminate the adverse impact of malicious clients on the learning process [41]. We adopt the same image classification setup as in the client fairness experiments and simulate three representative Byzantine attacks: Gaussian Attack (GA) [30], [38], which injects updates sampled from $\mathbb{N}(0, 30^2)$; Sign-Flipping Attack (S-F) [38], which corrupts updates by multiplying local model weights by -10; and A Little Is Enough Attack (ALIE) [30], which constructs malicious updates using the mean and standard deviation of benign gradients.

In this setting, training loss is used as the TPM. To suppress the influence of Byzantine clients during aggregation, the CRS adopts a loss-clipping strategy that assigns negligible weight

(a) MNIST.

DH \ Methods	DFedAvg	q-FDFL ($q=0.1$)	q-FDFL ($q=0.01$)	q-FDFL ($q=0.5$)	DFedReweighting ($\mathcal{T}=0.01$)	DFedReweighting ($\mathcal{T}=0.1$)	DFedReweighting ($\mathcal{T}=0.5$)
IID	0.807(91.867)	0.964(91.760)	0.787(91.873)	**0.608**(91.254)	0.896(91.846)	0.850(92.120)	0.838(91.996)
Diri(0.1)	380.611(79.454)	95.408(84.953)	110.003(84.573)	29.335(88.512)	9.191(95.994)	**8.731**(95.900)	56.058(92.576)
Diri(1.0)	9.712(91.339)	8.030(91.628)	8.023(91.656)	8.539(91.457)	9.383(90.941)	**5.870**(92.512)	7.099(92.006)
LabelSkew(4)	107.049(85.449)	220.539(80.629)	188.987(79.877)	13.646(91.036)	5.725(95.598)	**4.125**(95.414)	6.468(95.328)

(b) Fashion MNIST.

DH \ Methods	DFedAvg	q-FDFL ($q=0.1$)	q-FDFL ($q=0.01$)	q-FDFL ($q=0.5$)	DFedReweighting ($\mathcal{T}=0.01$)	DFedReweighting ($\mathcal{T}=0.1$)	DFedReweighting ($\mathcal{T}=0.5$)
IID	1.455(81.597)	1.237(83.040)	1.151(81.757)	1.153(83.947)	**1.005**(82.460)	1.243(82.147)	1.196(81.653)
Diri(0.1)	360.772(76.972)	335.066(76.112)	429.660(76.493)	414.974(74.970)	16.657(95.881)	**13.422**(95.390)	161.933(86.862)
Diri(1.0)	22.610(83.172)	18.020(83.204)	18.012(82.923)	20.125(83.686)	14.253(84.003)	**10.452**(85.549)	11.976(83.995)
LabelSkew(4)	184.043(75.315)	199.828(73.005)	147.575(72.308)	215.891(74.968)	17.589(91.773)	**15.876**(92.182)	35.302(89.697)

TABLE I: *Client fairness comparison (Var(Acc)) on MNIST and Fashion-MNIST.* For **DFedReweighting**, results are obtained using the TPM-CRS combination defined in (12). **Boldface** denotes the lowest variance within each scenario.

to updates with anomalously high loss:

$$\begin{cases} m_k = f(\mathbf{w}_k, \mathcal{A}_k) \\ s(m_i, M_k) = \frac{m_i}{\sum_{m_j \in M_k} m_j}, m_i = \begin{cases} m_i, & m_i \le \mu \\ 0, & m_j > \mu \end{cases}, \ i \in \mathcal{N}_k \end{cases} \tag{13}$$

where $f(\cdot)$ denotes the training loss and μ is the mean loss across received updates.

Results. We compare **DFedReweighting** against Median [42], mKrum, Trimmed Mean [42], Flame [43], and BALANCE [30]. As shown in Table II, our method achieves the best performance in the majority of settings and remains competitive in the remainder. Notably, it occasionally surpasses the theoretical upper bounds of baseline methods, an effect we attribute to the regularizing influence of benign noise during training.

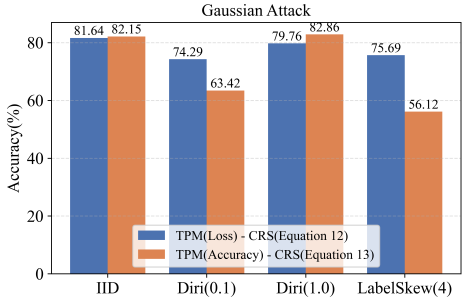

Fig. 3: *Impact of TPM-CRS combination choice on Byzantine attack robustness.* The loss-based combination (TPM: Loss, CRS: (13)) consistently outperforms the accuracy-based alternative (TPM: Accuracy, CRS: (14)). Results under S-F and ALIE attacks follow the same trend.

*3) **Another Alternative TPM-CRS Combination**:* While the TPM-CRS design space is flexible, different combinations yield different performance outcomes. To illustrate this, we evaluate an alternative combination for Byzantine robustness in which accuracy replaces loss as the TPM, paired with an accuracy-based clipping rule as the CRS:

$$\begin{cases} m_k = Acc_k \\ s(m_i, M_k) = \frac{m_i}{\sum_{j \in M_k} m_j}, m_i = \begin{cases} m_i, & m_i \ge \mu, \\ 0, & m_i < \mu. \end{cases} \end{cases}, \tag{14}$$

where μ is the mean accuracy serving as the clipping threshold. As shown in Fig. 3, the loss-based combination (13) consistently outperforms the accuracy-based alternative (14) under non-IID settings (Diri(0.1), Diri(1.0), and LabelSkew(4)), while achieving comparable performance under IID conditions, demonstrating its superior robustness to data heterogeneity.

*4) **Multi-objective Task across Clients**:* Multi-objective tasks across clients refer to settings where different clients pursue distinct learning objectives within the same DFL system. For instance, some clients may prioritize fairness while others seek to maximize accuracy. A natural solution is to equip each client with a different TPM-CRS combination aligned to its individual objective, as illustrated in Table III.

*5) **Multi-objective Task within Each Client**:* Multi-objective tasks within each client refer to settings where every client simultaneously optimizes multiple objectives [44], a scenario common in realistic DFL deployments. Unlike the across-client case, all clients here share the same multi-criteria learning goal and solve an identical multi-objective optimization problem. We thus propose the following TPM-CRS combination:

$$\begin{cases} m_k = \frac{1}{2} Acc_k + \frac{1}{2} f(\mathbf{w}_k, \mathcal{A}_k) \\ s(m_i, M_k) = f_e \end{cases}, \tag{15}$$

where the TPM is a combination of local accuracy Acc_k and training loss $f(\mathbf{w}_k, \mathcal{A}_k)$, and f_e is a monotonically increasing function that assigns higher aggregation weights to updates with greater TPM scores.

VII. Conclusion

This paper proposes DFedReweighting, a unified aggregation framework for decentralized federated learning that

(a) MNIST.

Attack	DH	No-AT	Median	mKrum	TM	Flame	BALANCE	Ours
GA	IID	92.30±0.06	91.79±0.20	89.43±1.97	**92.39**±0.07	91.98±0.66	91.19±0.87	91.44±0.92
	Diri(0.1)	79.67±4.26	68.18±5.82	62.57±5.35	77.85±3.32	68.26±17.99	**79.50**±1.63	76.40±0.87
	Diri(1.0)	91.04±0.31	90.50±0.54	86.69±1.44	90.53±0.53	90.47±0.40	**90.99**±0.49	89.98±0.16
	LabelSkew(4)	81.32±0.30	71.55±0.43	61.42±0.00	70.42±3.29	68.49±1.41	79.95±0.77	**80.55**±2.18
S-F	IID	92.30±0.06	90.56±0.56	91.40±0.00	91.29±0.00	92.35±0.10	91.19±0.87	**92.38**±0.14
	Diri(0.1)	79.67±4.26	51.27±13.16	78.79±0.00	72.69±9.27	87.35±2.23	79.50±1.63	**88.37**±0.14
	Diri(1.0)	91.04±0.31	90.00±0.60	88.99±0.00	89.65±0.68	**91.26**±0.14	90.99±0.49	90.45±1.13
	LabelSkew(4)	81.32±0.30	65.49±11.64	**89.55**±0.10	84.06±0.00	84.96±0.72	79.95±0.77	89.21±0.40
ALIE	IID	92.30±0.06	91.29±0.58	90.42±1.56	**92.37**±0.08	92.33±0.11	91.19±0.87	92.18±0.01
	Diri(0.1)	79.67±4.26	83.47±1.88	69.29±7.10	83.55±2.96	**87.84**±1.84	82.41±0.54	73.21±2.06
	Diri(1.0)	91.04±0.31	90.45±0.23	87.57±1.01	90.46±0.58	**91.26**±0.16	90.99±0.49	90.66±0.82
	LabelSkew(4)	81.32±0.30	81.65±1.58	**89.55**±0.10	87.36±1.52	86.30±0.64	79.95±0.77	88.90±0.28

(b) Fashion MNIST.

Attack	DH	No-AT	Median	mKrum	TM	Flame	BALANCE	Ours
GA	IID	81.60±0.68	81.52±0.33	81.28±0.86	81.21±0.80	**81.87**±0.52	81.70±0.47	81.32±0.52
	Diri(0.1)	71.82±1.69	66.31±2.10	57.93±1.98	**67.99**±1.63	58.48±3.59	67.39±3.12	65.19±6.46
	Diri(1.0)	82.77±0.59	82.10±0.90	78.99±1.76	82.34±0.76	82.27±0.78	**82.65**±0.71	76.61±2.68
	LabelSkew(4)	71.88±3.17	67.96±0.48	62.71±6.21	67.59±1.75	67.26±4.63	**73.97**±3.64	72.31±2.81
S-F	IID	81.60±0.68	83.22±0.63	82.71±0.35	83.04±0.73	**83.93**±0.34	83.50±0.82	83.48±0.98
	Diri(0.1)	71.82±1.69	63.01±2.72	64.00±3.28	63.67±5.07	76.29±0.70	**79.92**±0.94	78.31±1.05
	Diri(1.0)	82.77±0.59	81.84±1.00	80.15±0.64	81.75±0.50	82.96±0.36	**83.34**±0.23	82.50±0.15
	LabelSkew(4)	71.88±3.17	62.55±7.90	76.66±4.77	71.34±3.32	77.34±2.17	**82.50**±0.23	81.95±0.30
ALIE	IID	81.60±0.68	**83.79**±0.20	82.71±0.35	83.63±0.60	83.59±0.30	55.76±4.89	83.49±0.94
	Diri(0.1)	71.82±1.69	72.54±1.83	64.00±3.28	69.83±3.95	77.83±1.66	**78.72**±2.10	59.14±4.00
	Diri(1.0)	82.77±0.59	81.75±0.87	80.15±0.64	81.47±1.73	**82.74**±0.51	63.78±8.45	80.70±2.70
	LabelSkew(4)	71.88±3.17	69.18±2.49	76.66±4.77	76.15±1.08	76.95±1.43	30.71±8.00	**81.97**±0.21

TABLE II: *Byzantine attack robustness comparison (Acc±std) on MNIST and Fashion MNIST.* For BALANCE, hyperparameters are tuned to optimal performance ($\alpha = 0.1$, $\gamma = 2$ and $\kappa = 1$). For **DFedReweighting**, results are obtained using the TPM-CRS combination defined in (13). **Boldface** and grey denote the best and second-best accuracy within each scenario, excluding **No-AT**.

achieves diverse learning objectives via objective-oriented reweighting at the final step of each training round. The framework is broadly extensible: by appropriately designing the target performance metric (TPM) and customized reweighting strategy (CRS), it can be readily adapted to a wide range of DFL challenges. Theoretical analysis establishes linear convergence guarantees for general L-smooth and strongly convex functions under suitable TPM-CRS combinations. Empirical evaluations on client fairness and Byzantine robustness demonstrate consistent improvements over competitive baselines, and two multi-objective examples further illustrate the flexibility of the TPM-CRS design paradigm in accommodating diverse learning objectives.

Clients	Objective	TPM (m_k)	CRS ($s(m_i, M_k)$)
Client 1	Client Fairness	Acc_k	$\frac{e^{m_i/\mathcal{T}}}{\sum_{m_j \in M_k} e^{m_j/\mathcal{T}}}$
Client 2	Group Fairness	Acc_k	$\frac{m_i}{\sum_{m_j \in M_k} m_j}$
Client 3	Byzantine Robustness	$f(\mathbf{w}_k, \mathcal{A}_k)$	$\frac{m_i}{\sum_{m_j \in M_k} m_j}$
Client 4	Fast Convergence	$f(\mathbf{w}_k, \mathcal{A}_k)$	$\mathrm{clip}(m_i, 0, t)$

TABLE III: Multi-objective task across clients in a four-client DFL system, readily scalable to larger networks and more diverse TPM-CRS combinations.

REFERENCES

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

[2] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

[3] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "Spreadgnn: Serverless multi-task federated learning for graph neural networks," *arXiv preprint arXiv:2106.02743*, 2021.

[4] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third Workshop on Bayesian Deep Learning (NeurIPS)*, vol. 2, 2018.

[5] K. Zhang, A. Basharat, and P. Xu, "Byzantine-robust decentralized federated learning via local performance checking," in *31st International Conference on Neural Information Processing*, 2024.

[6] K. Zhang, P. Xu, and Z. Tian, "Distributional and byzantine robust decentralized federated learning," in *2025 59th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2025, pp. 1–6.

[7] V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, "Federated learning for connected and automated vehicles: A survey of existing approaches and challenges," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 119–137, 2023.

[8] S. Baghersalimi, T. Teijeiro, A. Aminifar, and D. Atienza, "Decentralized federated learning for epileptic seizures detection in low-power wearable systems," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 6392–6407, 2023.

[9] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *arXiv preprint arXiv:2101.03300*, 2021.

[10] K. Le, N. Luong-Ha, M. Nguyen-Duc, D. Le-Phuoc, C. Do, and K.-S. Wong, "Exploring the practicality of federated learning: A survey towards the communication perspective," *arXiv preprint arXiv:2405.20431*, 2024.

[11] L. Yuan, Z. Wang, L. Sun, P. S. Yu, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet of Things Journal*, 2024.

[12] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," *arXiv preprint arXiv:1905.10497*, 2019.

[13] Z. Wang, X. Fan, J. Qi, C. Wen, C. Wang, and R. Yu, "Federated learning with fair averaging," *arXiv preprint arXiv:2104.14937*, 2021.

[14] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 4615–4625.

[15] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Fedmgda+: Federated learning meets multi-objective optimization," *arXiv preprint arXiv:2006.11489*, 2020.

[16] X. Yue, M. Nouiehed, and R. Al Kontar, "Gifair-fl: A framework for group and individual fairness in federated learning," *INFORMS Journal on Data Science*, vol. 2, no. 1, pp. 10–23, 2023.

[17] Y. H. Ezzeldin, S. Yan, C. He, E. Ferrara, and S. Avestimehr, "Fairfed: Enabling group fairness in federated learning," *arXiv preprint arXiv:2110.00857*, 2021.

[18] W. Du, D. Xu, X. Wu, and H. Tong, "Fairness-aware agnostic federated learning," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 181–189.

[19] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[20] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[21] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, "Communication-efficient federated learning via knowledge distillation," *Nature Communications*, vol. 13, no. 1, p. 2032, 2022.

[22] M. Ribero and H. Vikalo, "Reducing communication in federated learning via efficient client sampling," *Pattern Recognition*, vol. 148, p. 110122, 2024.

[23] S. M. Shah and V. K. Lau, "Model compression for communication efficient federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 9, pp. 5937–5951, 2021.

[24] H. Lei, S. Gong, Q. Dou, and F. Farnia, "pfedfair: Towards optimal group fairness-accuracy trade-off in heterogeneous federated learning," *arXiv preprint arXiv:2503.14925*, 2025.

[25] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[26] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[27] H. Guo, H. Wang, T. Song, Y. Hua, Z. Lv, X. Jin, Z. Xue, R. Ma, and H. Guan, "Siren: Byzantine-robust federated learning via proactive alarming," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 47–60.

[28] J. Regatti, H. Chen, and A. Gupta, "Bygars: Byzantine sgd with arbitrary number of attackers," *arXiv preprint arXiv:2006.13421*, 2020.

[29] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4289–4301, 2022.

[30] M. Fang, Z. Zhang, P. Khanduri, S. Lu, Y. Liu, N. Gong *et al.*, "Byzantine-robust decentralized federated learning," *arXiv preprint arXiv:2406.10416*, 2024.

[31] Z. Wu, T. Chen, and Q. Ling, "Byzantine-resilient decentralized stochastic optimization with robust aggregation rules," *IEEE Transactions on Signal Processing*, vol. 71, pp. 3179–3195, 2023.

[32] A. E. Abyane, S. Drew, and H. Hemmati, "Mda: Availability-aware federated learning client selection," *arXiv preprint arXiv:2211.14391*, 2022.

[33] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[34] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 487–500, 2022.

[35] C. Jiang, J. Fan, T. Halabi, and I. Haque, "Performance analysis of decentralized federated learning deployments," 2025. [Online]. Available: https://arxiv.org/abs/2503.11828

[36] Y. Li and X. Lyu, "Convergence analysis of sequential federated learning on heterogeneous data," *Advances in Neural Information Processing Systems*, vol. 36, pp. 56 700–56 755, 2023.

[37] C. Yang and J. Ghaderi, "Byzantine-robust decentralized learning via remove-then-clip aggregation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 19, 2024, pp. 21 735–21 743.

[38] H. Ye, H. Zhu, and Q. Ling, "On the tradeoff between privacy preservation and byzantine-robustness in decentralized learning," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 9336–9340.

[39] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[40] S. Guo, X. Yang, J. Feng, Y. Ding, W. Wang, Y. Feng, and Q. Liao, "Fedgr: Federated learning with gravitation regulation for double imbalance distribution," in *International Conference on Database Systems for Advanced Applications*. Springer, 2023, pp. 703–718.

[41] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[42] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. Pmlr, 2018, pp. 5650–5659.

[43] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[44] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Federated learning meets multi-objective optimization," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2039–2051, 2022.