Distributional and Byzantine Robust Decentralized Federated Learning

Kaichuang Zhang
Dept. of ECE
University of Texas Rio Grande Valley
Edinburg, TX, USA
kaichuang.zhang01@utrgv.edu

Ping Xu
Dept. of ECE
University of Texas Rio Grande Valley
Edinburg, TX, USA
ping.t.xu@utrgv.edu

Zhi Tian
Dept. of ECE
George Mason University
Fairfax, VA, USA
ztian1@gmu.edu

2025 59th Annual Conference on Information Sciences and Systems (CISS) | 979-8-3315-1326-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/CISS64860.2025.10944744

***Abstract*—Decentralized federated learning (DFL) offers enhanced resilience to client failures and potential attacks than its centralized counterpart. This advantage stems from its ability to aggregate learning models from distributed clients requiring centralized server coordination. However, the practical adoption of DFL faces several challenges that threaten the robustness of local models. On one hand, the distribution of data might change over time, degrading the aggregated model's performance on test data. On the other hand, Byzantine attacks, where certain users send malicious updates to their neighbors to spread erroneous knowledge, can compromise the convergence and accuracy of the global model. Notably, no existing work has simultaneously addressed both distributional shifts and Byzantine attacks in decentralized settings. To bridge this gap, we first propose a robust aggregation algorithm, Local Performance Evaluation with Temperature-Scaled Softmax Reweighting (LPE-TSR), to defend against Byzantine attacks. We then integrate Wasserstein distributionally robust optimization with LPE-TSR and develop Distributional and Byzantine Robust Decentralized Stochastic Gradient Descent (DB-Robust DSGD) to tackle both challenges simultaneously. DB-Robust DSGD allows flexible selection of robust aggregation algorithms tailored to specific scenarios. Experimental results show that LPE-TSR achieves optimal performance across diverse attack scenarios, while DB-Robust DSGD effectively mitigates both distributional shifts and Byzantine attacks.**

***Index Terms*—Decentralized Federated Learning, Distributionally Robust Optimization, Byzantine Robustness**

I. Introduction

In decentralized federated learning (DFL), multiple clients collaboratively carry out a common task based on a shared machine learning model. Each client iteratively updates a local model based on its raw data and exchanges model information with neighboring clients within a one-hop communication range. By eliminating the need for central server coordination, DFL offers greater resilience to client failures and potential attacks compared to its centralized counterpart [1], [2]. Moreover, the fully decentralized structure and local training property of DFL inherently allow for efficient allocation and utilization of computational resources. Yet, two major robustness challenges in DFL, namely Byzantine attacks and distributional shifts, must be addressed to ensure model safety before deploying it in modern safety-critical fields such as autonomous driving [3] and medical diagnosis [4], etc.

Byzantine attackers are dishonest clients that send arbitrary malicious information to deliberately disrupt the entire system, potentially compromising convergence or degrading the accuracy of the global model [5]–[7]. To defend against Byzantine attacks, extensive work has been conducted. Statistics-based methods leverage statistical features such as the median, mean, variance, percentile, and distance to exclude potential attackers from aggregation [5], [8]–[11]. Anomaly detection-based approaches identify clients with abnormal updates as anomalies and perform aggregation after excluding these anomalous clients [12]–[15]. Performance evaluation-based methods use an additional evaluation dataset to filter out malicious updates before model aggregation [16]–[18].

Distributional shift refers to the mismatch between the distributions of training data and real-world test data [19], [20]. Traditional empirical risk minimization (ERM) methods often assume that training and test data share the same distribution. However, this assumption frequently fails due to dynamic environmental changes and uncertainty. To address distributional shift, robust optimization assumes no prior knowledge of the data distribution and minimizes the worst-case scenario [21], [22]. However, this approach can lead to over conservative solutions that perform poorly in practice. In contrast, distributionally robust optimization (DRO) assumes that the real-world data distribution lies within an ambiguity set of the training data distribution, which is less conservative in handling distributional ambiguity while still preserving key statistical properties of the data [23], [24]. The ambiguity set can be constructed based on moment conditions [25] or probability distance such as f-divergence [26], [27] and Wasserstein distance [28], [29]. Notably, distributional shift occurs across all machine learning settings, including both centralized and federated learning [30].

Although extensive efforts have been made to address Byzantine attacks and distributional shifts separately, few works have tackled both issues simultaneously. [30] attempts to address both challenges; however, their federated learning setting is less challenging than our DFL setting, where no central server exists for model aggregation. In this work, we aim to bridge this gap by developing a distributional

This work was partly supported by the National Science Foundation (Grants #2434916, #1939553).

and Byzantine robust decentralized learning algorithm. We first leverage the unique nature of DFL, where each client acts as a central server with its own evaluation dataset, an integral component of performance evaluation-based robust aggregation mechanism, and propose a novel approach for performance evaluation and attacker detection in DFL systems. We then integrate Wasserstein DRO with various robust aggregation techniques to address both distributional shifts and Byzantine attacks simultaneously. Our main contributions are summarized as follows.

- We propose the Local Performance Evaluation with Temperature-Scaled Softmax Reweighting (LPE-TSR) algorithm to defend against Byzantine attackers. LPE-TSR allows each client to utilize its local training dataset as a verification dataset to identify malicious and benign updates. The influence or importance of benign updates on the new aggregated model is controlled by a Temperature-scaled Softmax Reweighting scheme, leading to a more adaptive and potentially robust aggregated model.
- We adopt the Wasserstein distance to construct the ambiguity set in DRO and integrate it with Byzantine robust aggregation methods to form the Distributional and Byzantine Robust Decentralized Stochastic Gradient Descent (DB-Robust DSGD) algorithm. Wasserstein distance captures the geometric relationship between distributions and directly measures the minimum transporting cost from one distribution to another. Additionally, it is more sensitive to local perturbations than mean, variance, or KL divergence methods. The integrated DB-Robust DSGD algorithm address distributional shift and Byzantine attack issues at the same time, which, to the best of our knowledge, is the first work that tackles both robustness issues at the same time.
- We conduct extensive experiments to validate the effectiveness of the proposed algorithms on real-world datasets. With the Fashion MNIST dataset, we show that LPE-TSR is more robust than classic robust aggregation algorithms (median, Trimmed mean, Krum) in most scenarios. Additionally, experiments on the Spambase dataset reveal that DB-Robust DSGD exhibits superior distributional and Byzantine robustness compared to the conventional ERM algorithm.

II. Problem Statement

Consider a fully decentralized network of N clients interconnected over a fixed undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the client set and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ denotes the edge set. Two clients n and m are said to be one-hop neighbors if $(n, m) \in \mathcal{E}$, which can exchange model information during the communication stage. By the symmetry of the network, $(m, n) \in \mathcal{E}$. For client n, its one-hop neighbors are in the set $\mathcal{N}_n = \{m | (m, n) \in \mathcal{E}\}$. Under the assumption that all clients work properly, the learning task of DFL is to minimize the average of local clients' loss, given by

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{x_n \sim \mathcal{D}_n} \, f_n(\boldsymbol{w}; x_n), \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the model parameter to be learned and $f_n(\boldsymbol{w}; x_n)$ is the local loss of client n with respect to data x_n sampled from local data distribution $\mathcal{D}_n$, i.e., $x_n \sim \mathcal{D}_n$.

Various popular decentralized algorithms such as decentralized stochastic gradient descent (DSGD) and decentralized alternating direction method of multipliers (ADMM) can be utilized to solve (1). Take DSGD as an example, at each communication round t, each client n ($n \in \mathcal{N}$) performs the following three-stage operations [31], [32].

Stage 1 (local training): a local SGD step is taken to obtain an intermediate update $\boldsymbol{w}_n^{t+\frac{1}{2}}$:

$$\boldsymbol{w}_n^{t+\frac{1}{2}} = \boldsymbol{w}_n^t - \eta^t \nabla f_n(\boldsymbol{w}_n^t; x_n^t), x_n^t \sim \mathcal{D}_n, \tag{2}$$

where η^t is the learning rate, $\boldsymbol{w}_n^t$ is the local model of client n at communication round t, x_n^t is the data sample of client n at communication round t, and $\nabla f_n(\boldsymbol{w}_n^t; x_n^t)$ is the gradient obtained from x_n^t at communication round t.

Stage 2 (communication): client n sends $\boldsymbol{w}_n^{t+\frac{1}{2}}$ to its neighbors m ($m \in \mathcal{N}_n$) and receives $\boldsymbol{w}_m^{t+\frac{1}{2}}$ from its neighbors m ($m \in \mathcal{N}_n$), resulting in a local updates set $\boldsymbol{W}_n$ that includes client n's and its neighbors' updates, i.e.,

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

Stage 3 (aggregation): client n aggregates all local models to get a new local model as

$$\boldsymbol{w}_n^{t+1} = \sum_{\boldsymbol{w}_i \in \boldsymbol{W}_n} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \boldsymbol{w}_i^{t+\frac{1}{2}}, \tag{4}$$

where $\mathcal{D} := \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_N$ is the total data sample set.

These operations iterate until all local models converge to the global model $\boldsymbol{w}^T$, i.e., $\boldsymbol{w}_1^T = \boldsymbol{w}_2^T = \ldots, = \boldsymbol{w}_N^T$, which can then be deployed for practical usage. However, two safety issues must be addressed before implementing DFL in practice. First, unexpected behaviors such as data corruption, device malfunctioning, or even malicious attacks, if not treated properly, can lead to the failure of the whole system. Denote $\mathcal{M}$ as the malicious clients set containing clients that send omniscient arbitrarily malicious updates, and correspondingly $\mathcal{B}$ as the benign clients set. Clearly, $\mathcal{M} \cup \mathcal{B} = \mathcal{N}$. Under Byzantine attacks, the local training in **Stage 1** follows:

$$\boldsymbol{w}_n^{t+\frac{1}{2}} = \begin{cases} \star, & n \in \mathcal{M} \\ (2) & n \in \mathcal{B} \end{cases}, \tag{5}$$

where $\star$ denotes arbitrarily malicious updates.

To eliminate the negative impact of malicious attacks, instead of naive aggregation (4), Byzantine robust aggregation algorithms (denoted as **BRAgg()**) need to be equipped for all honest clients n ($n \in \mathcal{B}$) to perform local aggregation:

$$\boldsymbol{w}_n^{t+1} = \textbf{BRAgg}(\boldsymbol{w}_i^{t+\frac{1}{2}} | \boldsymbol{w}_i^{t+\frac{1}{2}} \in \mathcal{M}_n \cup \mathcal{B}_n), \tag{6}$$

where $\mathcal{M}_n$ and $\mathcal{B}_n$ denotes the malicious neighbor set and the benign neighbor set of client n, respectively.

Secondly, the assumption that training data $\mathcal{D}_n$ $(\forall n)$ shares the same distribution as real-world data usually fails in practice due to the dynamic change of environments as well as environment uncertainty. To combat distributional shifts, distributional robust optimization (DRO) is required. Specifically, for each client n, DRO assumes that data x_n^t is sampled from $\mathcal{Q}_n$, the worst-case distribution in an ambiguity set Ω_n that encompasses a cluster of data distributions. In that way, the local training step in (2) changes to

$$\boldsymbol{w}_n^{t+\frac{1}{2}} = \boldsymbol{w}_n^t - \eta^t \nabla f_n(\boldsymbol{w}_n^t; x_n^t), x_n^t \sim \mathcal{Q}_n. \tag{7}$$

In the next section, we develop a novel robust aggregation scheme and describe how to construct the ambiguity set, based on which we develop a framework to address distributional shift and Byzantine attack issues simultaneously.

III. Proposed Algorithms

We first introduce a novel robust aggregation algorithm to mitigate the impact of malicious attacks and adaptively assign weights to benign updates based on their importance. The developed Local Performance Evaluation with Temperature-Scaled Softmax Reweighting (LPE-TSR) algorithm enhances resilience against adversarial influences. We then develop the Distributional and Byzantine Robust Decentralized Stochastic Gradient Descent (DB-Robust DSGD) algorithm by integrating Wasserstein DRO with various robust aggregation techniques to ensure robustness against both distributional shifts and Byzantine attacks.

A. LPE-TSR algorithm

The LPE-TSR method consists of two components: Local Performance Evaluation (LPE) and Temperature-Scaled Softmax Reweighting (TSR). The LPE component allows each client to randomly sample from its local training data to obtain an unbiased verification dataset. The classification/prediction accuracy on this dataset, computed from the received updates, is then used to determine whether the updates are malicious or benign. The TSR component utilizes a parameter $\mathcal{T}$ to enable adaptive aggregation based on the importance of benign updates. The LPE-TSR algorithm is presented in Algorithm 1, which consists of the following four steps.

Step 1 (Build the performance verification dataset $\mathcal{PVD}_n$). One of the defining characteristics of LPE-TSR is the random sampling of the local training dataset to form the performance verification dataset. Random sampling ensures that the distributions of the performance verification dataset and the local training dataset are consistent, thereby enabling a more confident performance evaluation.

Step 2 (Perform performance evaluation). After obtaining $\mathcal{PVD}_n$ in **Step 1**, we evaluate the performance of all models updates $\boldsymbol{w}_m$ that client n receives from its neighbors m ($m \in \mathcal{N}_n$) based on $\mathcal{PVD}_n$. The performance metric here can be accuracy, loss, or other metrics that can determine whether $\boldsymbol{w}_m$ is malicious or benign.

Step 3 (Determine the filtering threshold and filter out malicious updates). Once the performance of all updates $\boldsymbol{W}_n$ (including the local model $\boldsymbol{w}_n$) has been obtained, a filtering threshold will be calculated to classify benign and malicious updates. In this paper, we use average accuracy as the threshold, which is straightforward and effective according to the experimental results.

Step 4 (Aggregate model with temperature-scaled softmax reweighting (TSR)). After obtaining the potential benign neighbors, the local model is updated with TSR. Specifically, each neighbor's contribution is weighted according to its importance using a softmax function with a tunable temperature parameter $\mathcal{T}$. TSR ensures that the aggregation step assigns weights dynamically based on the performance scores a_j of each model in the set $\bar{W}_n$. By adjusting the temperature $\mathcal{T}$, the influence of each model can be controlled, leading to a more adaptive and potentially robust model update.

Algorithm 1 Local Performance Evaluation with Temperature-scaled Softmax Reweighting (LPE-TSR) at client n

Input: $\boldsymbol{W}_n = \{\boldsymbol{w}_n^{t+\frac{1}{2}}, \boldsymbol{w}_m^{t+\frac{1}{2}} (\forall m \in \mathcal{N}_n)\}$, sampling fraction α, temperature scale $\mathcal{T}$.
Output: $\boldsymbol{w}_n^{t+1}$.

1: sample α of the local training data to form the local performance verification dataset $\mathcal{PVD}_n$.
2: **for all** $\boldsymbol{w}_i^{t+\frac{1}{2}} \in \boldsymbol{W}_n$ **do**
2: get the performance a_i use $\mathcal{PVD}_n$.
3: **end for**
4: calculate the filtering threshold $\beta = \frac{1}{|\mathcal{N}_n|} \sum_{i=1}^{|\mathcal{N}_n|} a_i$.
5: filter $\boldsymbol{w}_i^{t+\frac{1}{2}}$ if $a_i \leq \beta$ to get a new update set $\bar{\boldsymbol{W}}_n$ and the new index set $\bar{\mathcal{N}}_n := \{i | a_i > \beta\}$.
6: update local model $\boldsymbol{w}_n^{t+1} = \sum_{j \in \bar{\mathcal{N}}_n} \frac{e^{a_j/\mathcal{T}}}{\sum_{k \in \bar{\mathcal{N}}_n} e^{a_k/\mathcal{T}}} \boldsymbol{w}_j$.
7: return $\boldsymbol{w}_n^{t+1}$.

B. DB-Robust DSGD algorithm

To tackle Byzantine attacks and distributional shifts simultaneously, we present the DB-Robust DSGD algorithm. Motivated by the successful implementation of distributed Wasserstein DRO in [30], we start by constructing ambiguity sets based on Wasserstein distance under the decentralized setting. For each client n, the ambiguity set Ω_n is chosen as a Wasserstein ball $\Omega_n := B_\rho(\mathcal{D}_n) = \{\mathcal{Q}_n : W_c(\mathcal{Q}_n, \mathcal{D}_n) \leq \rho_n\}$, with the empirical distribution $\mathcal{D}_n$ being the center and ρ_n being the radius, and $W_c(\cdot, \cdot)$ is the Wasserstein distance between two probability distributions $\mathcal{Q}_n$ and $\mathcal{D}_n$ with $c(\cdot, \cdot)$ being the transportation cost between two data points.

The optimization problem under distributional shifts and Byzantine attacks is then formulated as

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{B}|} \sup_{\mathcal{Q}_n : W_c(\mathcal{Q}_n, \mathcal{D}_n) \leq \rho_n} \mathbb{E}_{x_n \sim \mathcal{Q}_n} f_n(\boldsymbol{w}; x_n). \tag{8}$$

However, solving (8) exactly with all prespecified ρ_n's is infeasible. Instead, we follow the duality proof in

[33] and the distributed Wasserstein DRO [30] and derive that $\sup_{\mathcal{Q}_n:W_c(\mathcal{Q}_n,\mathcal{D}_n)\leq\rho_n}\mathbb{E}_{x_n\sim\mathcal{Q}_n}f_n(\boldsymbol{w};x_n) = \inf_{\lambda_n\geq 0}\{\lambda_n\rho_n+\mathbb{E}_{x_n\sim\mathcal{D}_n}\phi_{\lambda_n}(\boldsymbol{w}_n;x_n)\}$, where $\phi_{\lambda_n}(\boldsymbol{w}_n;x_n)=\sup_{z_n}\{f_n(\boldsymbol{w}_n;z_n)-\lambda_n c(z_n,x_n)\}$ represents the robust surrogate of $f_n(\boldsymbol{w}_n;x_n)$ and λ_n is the dual variable for agent n. In this way, given some fixed $\lambda_n\geq 0$, we turn to solve a easier problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^d}\frac{1}{|\mathcal{B}|}\sum_{n=1}^{|\mathcal{B}|}\mathbb{E}_{x_n\sim\mathcal{D}_n}\ \phi_{\lambda_n}(\boldsymbol{w};x_n). \tag{9}$$

The key to solving (9) is to find the best z_n^* that maximizes $f_n(\boldsymbol{w}_n;z_n)-\lambda_n c(z_n,x_n)$. In most cases, the maximizer $z_n^*(\boldsymbol{w}_n)$ does not have a closed-form solution, and thus can only be solved to a certain precision via iterative methods, which will be detailed in the experimental part. After obtaining the ϵ-optimal maximizer $z_n^\epsilon(\boldsymbol{w}_n)$, DSGD can be utilized to update the model following steps listed in Section II.

We summarize the proposed DB-Robust DSGD algorithm in Algorithm 2, where **BRAgg()** in (6) can be any Byzantine robust aggregation algorithms, including our proposed LPE-TSR scheme. In this way, DB-Robust DSGD is robust to both Byzantine attackers and distributional shifts.

Algorithm 2 Distributional and Byzantine Robust Decentralized Stochastic Gradient Descent (DB-Robust DSGD)

Input: local model $\boldsymbol{w}^0$, local training datasets $\mathcal{D}_n$, dual variable λ_n, total communication round T.
Output: $\boldsymbol{w}_n^T$

Initialize all local models $\boldsymbol{w}_n^0=\boldsymbol{w}^0$, $n\in\mathcal{N}$.
for $t=0,1,\ldots,T-1$ **do**
  for $n=1,2,\ldots,|\mathcal{N}|$ **do**
    sample data x_n^t from $\mathcal{D}_n$.
    obtain $z_n^\epsilon(\boldsymbol{w}_n^t)$ for each local sample x_n^t by solving $\sup_{z_n}\{f_n(\boldsymbol{w}_n^t;z_n)-\lambda c(z_n,x_n^t)\}$ to ϵ precision.
    compute local gradient and get $\boldsymbol{w}_n^{t+\frac{1}{2}}$ by

$$\boldsymbol{w}_n^{t+\frac{1}{2}}=\begin{cases}\star, & n\in\mathcal{M}\\ \boldsymbol{w}_n^t-\eta^t\nabla f_n(\boldsymbol{w}_n^t;z_n^\epsilon(\boldsymbol{w}_n^t)), & n\in\mathcal{B}\end{cases}.$$

    send $\boldsymbol{w}_n^{t+\frac{1}{2}}$ to neighbors in $\mathcal{N}_n$ and receive $\boldsymbol{w}_m^{t+\frac{1}{2}}$ from neighbors in $\mathcal{N}_n$.
    perform robust local aggregation with (6).
  end for
end for
return $\boldsymbol{w}_n^T$.

IV. Experiments

To evaluate the effectiveness of the proposed LPE-TSR and DB-Robust DSGD methods, we conduct several group experiments under various scenarios.

A. Experiment settings

Hyperparameters. For LPE-TSR, all clients are trained using a softmax regression model with cross-entropy loss over $10,000$ communication rounds on the Fashion MNIST dataset [34]. The learning rate decays as $\eta^t=0.9\times\frac{0.9}{\sqrt{t+1}}$ as the communication rounds t increases. The temperature scaling factor is set to be $\mathcal{T}=0.1$. For DB-Robust DSGD, all clients are trained using a logistic regression model over $T=3,000$ communication rounds on the Spambase dataset [35]. The learning rate gradually decays as $\eta^t=0.01\times 0.9^{\lfloor\frac{t}{100}\rfloor}$ as the communication rounds t increases.

Network topologies. Our study utilizes a random Erdős–Rényi graph containing $|\mathcal{B}|+|\mathcal{M}|$ nodes ($|\mathcal{B}|$ benign nodes and $|\mathcal{M}|$ malicious nodes), characterized by a connection probability of 0.7. Following the settings in [36], benign nodes will hold data samples, whereas the malicious nodes will not hold any data samples.

Data perturbation in training. The perturbed sample is obtained by approximately solving $\sup_z\{f(\boldsymbol{w};z)-\lambda_n c(z,x)\}$ with $c(z,x)=\frac{1}{2}\|z-x\|^2$ [30]. In accordance with the adversarial perturbation on the test data, we only perturb the feature vector x into z without changing the label y. For logistic regression, the objective is $g(z)=-y\ln\hat{y}-(1-y)\ln(1-\hat{y})-\frac{\lambda}{2}\|z-x\|^2$ with $\hat{y}=\frac{1}{1+e^{-\boldsymbol{w}^Tz}}$, which has no closed-form solution. Therefore, we calculate the approximate maximizer via gradient ascent $z^{t+1}=z^t+\eta_z\nabla g(z^t)$ using T_z iterations initialized at $z^0=x$. The values for λ_n, η_z, and T_z are indicated in Table II.

Distributional shifts. The distributional shifts are simulated by adding perturbations to the test data as [30]:

$$\begin{cases}L_1\text{ shift: }||z-x||\leq q\\ L_2\text{ shift: }||z-x||_2\leq q\end{cases}, \tag{10}$$

where x is the original data sample in test dataset, z is the shifted data sample after adding perturbations, and q is the controlled budget for distributional shifts. Larger q indicates more data distribution deviations.

Byzantine attacks. We use 4 model poisoning attacks: Gaussian Attack (GA) [36], [37], Sign-Flipping Attack (S-F) [36], A Little Is Enough Attack (ALIE) [37]–[39], and Same-Value Attack (SA) [40], [41]. GA sends an update conforming to Gaussian distribution $\mathbb{N}(0,30^2)$. S-F flips the local model weight's sign by multiplying by -10. ALIE leverages the mean and standard deviation to generate malicious updates. SA sends malicious information with a constant value 0.

Benchmark robust aggregation algorithms. To compare the Byzantine robustness of LPE-TSR and select an appropriate **BRAgg()** for DB-Robust DSGD, the following classic robust aggregation methods are evaluated: Median [11], Trimmed Mean (TM) [11], and Krum [8].

Evaluation metric. The primary evaluation metric for both Byzantine robustness and distributional shift robustness is the average test accuracy (**Acc.**) of all local models on the test dataset. A higher **Acc.** indicates greater robustness of the algorithm.

B. Experimental results

Byzantine robustness of LPE-TSR. We compare the proposed algorithm, LPE-TSR, with Median, TM, and Krum

TABLE I
ACC. (%) COMPARISON OF DIFFERENT AGGREGATION RULES WITH LPE-TSR UNDER VARIOUS SCENARIOS ON FASHION MNIST. $|\mathcal{B}| = 10$, $|\mathcal{M}| = 2$.

AT	Data	ERM	Median	TM	Krum	LPE-TSR
GA	i.i.d.	82.32	82.21	82.31	82.31	**82.41**
	non-i.i.d.	82.27	82.01	82.15	81.83	**82.20**
S-F	i.i.d.	82.32	79.71	79.20	82.31	**82.40**
	non-i.i.d.	82.27	74.62	73.20	81.83	**82.20**
ALIE	i.i.d.	82.32	82.19	82.33	82.15	**82.37**
	non-i.i.d.	82.27	81.10	**81.85**	79.88	81.12
SA	i.i.d.	82.32	79.79	79.22	82.19	**82.40**
	non-i.i.d.	82.27	74.82	73.29	10.00	**82.20**

across various scenarios. Additionally, we evaluate the performance of Empirical Risk Minimization (ERM) under non-attack settings, which serves as the theoretical upper bound. The independent and identically distributed (i.i.d.) setting indicates that each local dataset contains the same number of classes and has the same number of data samples. For the non-i.i.d setting, we let each client hold a subset of classes of data with 4 classes [36], [42]. As shown in Table I, LPE-TSR achieves the best performance in all scenarios except in the non-i.i.d. setting under the ALIE attack, although its performance in that setting is comparable to the best-performing algorithm TM. In addition, LPE-TSR outperforms the theoretical upper bound in the i.i.d. setting, which may be attributed to the benign noise.

Robustness of DB-Robust DSGD. We compare the performance of DB-Robust DSGD, which integrates various Byzantine-robust aggregation algorithms, with that of the ERM. From the first row of Table II, we observe that in the absence of Byzantine attacks and distributional shifts, the performance of ERM and DB-Robust DSGD is nearly identical. When L1 or L2 distributional shifts are present but no Byzantine attacks exist (row 6 and 11 in Table II), DB-Robust(*) demonstrates a certain higher level of robustness, outperforming ERM due to its integration of decentralized Wasserstein DRO. In scenarios with only Byzantine attacks (row 2, 3, 4, and 5 in Table II), DB-Robust(*) achieves significantly higher accuracy, which also indicates the severe impact of Byzantine attacks on the learning process in DFL systems compared to distributional shifts. Finally, when both L1 or L2 shifts and Byzantine attacks are present (remaining rows in Table II), DB-Robust(*) substantially outperforms ERM, further validating the superior robustness of our algorithm.

V. CONCLUSION

This paper proposes the first framework that addresses both Byzantine attacks and distributional shifts simultaneously. We started by proposing the Local Performance Evaluation with Temperature-Scaled Softmax Reweighting algorithm to mitigate the negative impact of Byzantine clients in DFL systems. We then integrate decentralized Wasserstein distributionally robust optimization with a plugable robust aggregation modular to adaptively mitigate distributional shifts and Byzantine attacks. Experimental results show that the proposed algorithms achieve superior performance compared with benchmark methods.

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] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2018.

[4] I. Kononenko, "Machine learning for medical diagnosis: history, state of the art and perspective," *Artificial Intelligence in Medicine*, vol. 23, no. 1, pp. 89–109, 2001.

[5] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[6] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

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

[8] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] C. Fang, Z. Yang, and W. U. Bajwa, "Bridge: Byzantine-resilient decentralized gradient descent," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 610–626, 2022.

[10] 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.

[11] 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.

[12] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[13] 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.

[14] L. Meng, Y. Wei, R. Pan, S. Zhou, J. Zhang, and W. Chen, "Vadaf: visualization for abnormal client detection and analysis in federated learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–23, 2021.

[15] L. Zhao, S. Hu, Q. Wang, J. Jiang, C. Shen, X. Luo, and P. Hu, "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2029–2041, 2020.

[16] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *ArXiv*, vol. abs/2012.13995, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:229678616

[17] 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.

[18] 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.

[19] A. Malinin, N. Band, G. Chesnokov, Y. Gal, M. J. Gales, A. Noskov, A. Ploskonosov, L. Prokhorenkova, I. Provilkov, V. Raina *et al.*, "Shifts: A dataset of real distributional shift across multiple large-scale tasks," *arXiv preprint arXiv:2107.07455*, 2021.

TABLE II
ACC. (%) COMPARISON OF DB-ROBUST UNDER VARIOUS SCENARIOS ON SPAMBASE DATASET. $|\mathcal{B}| = 100$, $|\mathcal{M}| = 10$, $\lambda_n = 3$, $T_z = 5$, $\eta_z = 0.05$.

SCENARIOS \ METHODS	ERM	**DB-Robust**(median)	**DB-Robust**(TM)	**DB-Robust**(Krum)	**DB-Robust**(LPE-TSR)
1. No-Shifts & No-ATT	93.48	92.11	**93.02**	92.83	92.89
2. No-Shifts & GA	54.35	91.92	92.44	**93.94**	92.60
3. No-Shifts & S-F	52.55	91.59	90.95	91.20	**92.57**
4. No-Shifts & ALIE	92.63	91.26	92.24	91.20	**92.57**
5. No-Shifts & SA	91.12	91.66	91.59	91.79	**92.11**
6. L1 Shifts & No-ATT	90.92	90.60	91.20	**92.21**	91.20
7. L1 Shifts & GA	54.95	90.61	90.89	**91.10**	90.59
8. L1 Shifts & S-F	52.45	90.60	90.05	**91.41**	90.51
9. L1 Shifts & ALIE	67.95	90.13	90.69	91.08	**91.18**
10. L1 Shifts & SA	89.94	90.50	90.60	**91.12**	89.89
11. L2 Shifts & No-ATT	68.03	71.54	71.28	68.70	**71.69**
12. L2 Shifts & GA	52.90	71.39	**71.62**	71.24	71.50
13. L2 Shifts & S-F	50.65	**71.63**	71.59	71.25	71.07
14. L2 Shifts & ALIE	67.95	**71.84**	71.80	71.18	71.81
15. L2 Shifts & SA	71.13	71.83	**72.19**	71.23	71.51

[20] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5637–5664.

[21] H.-G. Beyer and B. Sendhoff, "Robust optimization–a comprehensive survey," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 33-34, pp. 3190–3218, 2007.

[22] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM review*, vol. 53, no. 3, pp. 464–501, 2011.

[23] P. Mohajerin Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, no. 1, pp. 115–166, 2018.

[24] R. Chen, I. C. Paschalidis *et al.*, "Distributionally robust learning," *Foundations and Trends® in Optimization*, vol. 4, no. 1-2, pp. 1–243, 2020.

[25] J. Nie, L. Yang, S. Zhong, and G. Zhou, "Distributionally robust optimization with moment ambiguity sets," *Journal of Scientific Computing*, vol. 94, no. 1, p. 12, 2023.

[26] R. Xie and W. Wei, "f-divergence distributionally robust optimization," in *Distributionally Robust Optimization and its Applications in Power System Energy Storage Sizing*. Springer, 2024, pp. 171–211.

[27] D. Bauso, J. Gao, and H. Tembine, "Distributionally robust games: f-divergence and learning," in *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, 2017, pp. 148–155.

[28] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh, "Wasserstein distributionally robust optimization: Theory and applications in machine learning," in *Operations Research & Management Science in the Age of Analytics*. Informs, 2019, pp. 130–166.

[29] M.-C. Yue, D. Kuhn, and W. Wiesemann, "On linear optimization over wasserstein balls," *Mathematical Programming*, vol. 195, no. 1, pp. 1107–1122, 2022.

[30] G. Zhou, P. Xu, Y. Wang, and Z. Tian, "Robust distributed learning against both distributional shifts and byzantine attacks," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

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

[32] 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.

[33] A. Sinha, H. Namkoong, R. Volpi, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, 2017.

[34] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[35] A. Asuncion, D. Newman *et al.*, "Uci machine learning repository," 2007.

[36] 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.

[37] 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.

[38] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] X. Yueqi, M. Fang, and N. Z. Gong, "Fedredefense: Defending against model poisoning attacks for federated learning using model update reconstruction error," in *Forty-first International Conference on Machine Learning*.

[40] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[41] C. Che, X. Li, C. Chen, X. He, and Z. Zheng, "A decentralized federated learning framework via committee mechanism with convergence guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4783–4800, 2022.

[42] 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.