PAwBRoFL: Exploring Performance Aware Byzantine-Robust Federated Learning.

Heshan Fernando, Zhengjie Xiong, Aarnav Patel, Shane Stoll

Abstract

In this project, we explore the feasibility of incorporating some notion of client performance awareness when aggregating client gradients using Byzantine-robust aggregators when malicious clients are suspected to be available in the federated learning (FL) setup. To this end, we introduce the performance-aware Byzantine-robust federated learning (PAwBRoFL) framework and the corresponding algorithm, which attempts to integrate performance awareness and Byzantine-robustness when determining the final gradient to update the global model. We derive the analytical form of the PAwBRoFL gradient and provide experimental results on applying the PAwBRoFL algorithm to a simulated FL setting with malicious clients. Our code is available at `https://github.com/heshandevaka/PAwBRoFL.git`.

1 Introduction

Federated learning (FL) (McMahan et al. (2017)) is a decentralized approach to training machine learning models where data remains on local devices, and only model updates are shared with a central server. This setup enhances privacy by avoiding direct data transfer, making it suitable for applications such as healthcare and finance. However, the distributed nature of FL also presents challenges, particularly in the presence of Byzantine adversaries (Blanchard et al. (2017)): malicious clients or unreliable participants that can corrupt the model by sending incorrect or misleading updates. Designing robust aggregation mechanisms capable of defending against such Byzantine faults is critical for maintaining the integrity and reliability of federated learning systems.

Byzantine-robust federated learning (FL) is a crucial research area focused on developing methods to safeguard distributed learning systems against malicious clients attempting to disrupt the learning process. Most existing protocols (Blanchard et al. (2017); Zhang et al. (2021)) aim to identify and filter out malicious (outlier) gradients submitted by clients, removing them from the global gradient computation. The aggregated global gradient is then returned to clients to update their local models. However, while this approach may be robust to Byzantine attacks, it does not guarantee optimal performance across all clients. For example, the gradient of one client could dominate the learning process, potentially degrading the performance of other clients.

In this work, we address this issue by proposing a novel approach. Clients submit both their gradients and their local objective sub-optimalities to the server. Using existing protocols, the server first identifies the benign clients in the FL system. Next, the server computes a global performance-aware gradient for these benign clients, using both the gradient and suboptimality information. Specifically, we employ a gradient-smoothed max operator applied to the local sub-optimalities of the clients, which is aimed at a balanced and effective contribution from each client, which improves the overall learning process and maintains fairness across clients. The proposed framework is summarized in Figure **??**.

2 Related work

Byzantine-Robust Federated Learning. Zhu et al. (2023) introduced Byzantine-Robust Federated Learning protocols that provide optimal statistical rates and strong privacy guarantees. Furthermore, they demonstrated through benchmark data that these protocols consistently deliver high performance in various attack scenarios. Li et al. (2023) takes a deeper look at how different Byzantine

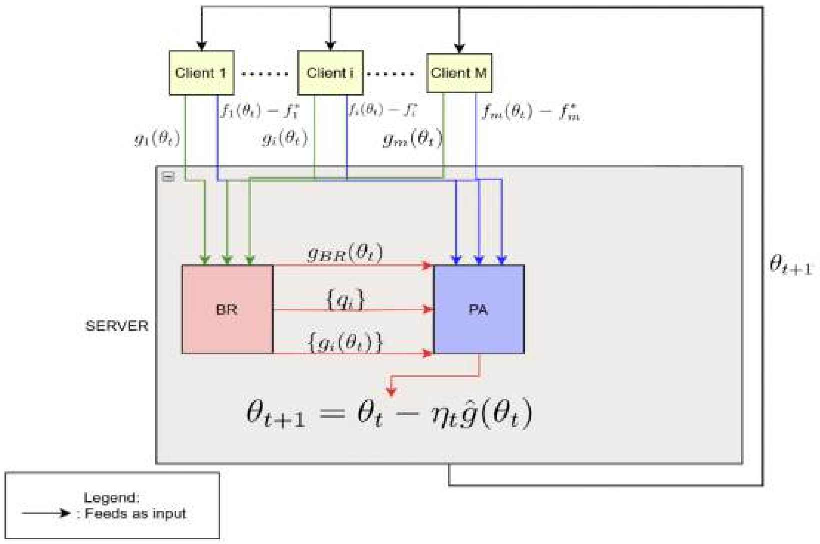

Figure 1: **Overview of the Framework.** Each client learns separately, and feeds the server with their gradient as well as performance. BR block outputs the byzantine robust gradient and corresponding weights. PA block generates the final PAwBRoFL gradient ($\hat{g}$) based on the BR block's outputs

models can be classified into three different categories based on their approach to Byzantine attacks. Models that utilize redundancy, like Chen et al. (2018), which spreads redundant gradients on each training datapoint across the clients, in which the system is able to use a subset of these additional gradients to add up to a standard gradient update, limiting the impact that adversarial clients could have on the model across iterations; Models that assume trust from certain clients Konstantinov & Lampert (2019), which learns to trust certain clients based on early updates to utilize as a base; and models that utilize robust aggregation algorithms, which is the category our paper aims to utilize to combat Byzantine attacks. They also propose their model *ClippedClustering* to try to combat 'A little is enough' (ALIE) attacks, in which attackers inject a small amount of poisoned data into the training data; however, Li notes that current Byzantine models need to improve when training on non-IID client datasets.

Performance aware Learning. In this article, we refer to performance-aware learning as the learning that takes into consideration the performance (objective value) of a set of objectives in a multi-objective setting, when trying to optimize all the objectives simultaneously (e.g. a weighted combination of the objectives). To this end, Tchebyshev scalarization Miettinen (1999) provides a way to reduce the multiple sets of objectives into a single objective by optimizing the objective with the maximum optimality gap. More recently, Lin et al. (2024) introduced a smoothed version of the aforementioned Tchebyshev scalarization, which we leverage in this paper to compute a performance-aware gradient in the FL setting. Other work in the federated learning setting aggregates the model parameters considering different "performance" metrics of the clients, such as quality Deng et al. (2021) and computing power (Seo et al., 2022), but these notions of performance awareness are out of the scope of this paper.

3 PRELIMINARIES

MODEL POISONING AND ATTACKS

Model poisoning attacks are significant threats in federated learning environments, where adversaries attempt to corrupt the global model by manipulating local model updates. There are different kinds of model poisoning attacks, including backdoor attacks (Nguyen et al. (2024)) and untargeted model poisoning (Al Mallah et al. (2023)). In backdoor attacks, malicious clients intentionally manipulate their local model updates to embed a hidden functionality (backdoor) into the global model, while ensuring that the model behaves normally on standard data. This can cause the model to produce targeted, incorrect outputs when specific triggers are present, allowing adversaries to compromise the system's integrity without detection. Untargeted model poisoning, on the other hand, aims to degrade the overall performance of the global model without specific triggers. In this project, we apply **Distributed Backdoor Attack** (DBA) (Xie et al. (2019)). DBA decomposes a global trigger pattern into distinct local patterns, embedding them into the training sets of different adversarial parties. Compared to standard centralized backdoors, DBA is significantly more persistent and stealthy against FL across diverse datasets, such as financial and image data. The algorithm is also cited in (Zhu et al. (2023)) for implementation.

Robust aggregation

To defend against model poisoning attacks, robust aggregation mechanisms are essential in federated learning. A robust aggregator aims to aggregate local model updates while minimizing the influence of malicious clients. Given local gradients, a robust aggregation function computes the global gradient while ensuring that the aggregated result is not significantly affected by outliers or adversarial contributions. In this project, we will apply the following two robust estimators:
No-Regret Algorithm (Hopkins et al. (2020); Zhu et al. (2022)) and **Filtering Algorithm** (Diakonikolas et al. (2017); Li (2018)) and the two algorithms are also cited in Zhu et al. (2023) for implementation. According to Zhu et al. (2022):

Definition 3.1 (Generalized quasi-gradient)**.** In the optimization problem $\min_{q\in A} F(q)$, we say $g(q)$ is a generalized quasi-gradient with parameter $C \geq 1$ if the following holds for all $q, p \in A$:

$$\langle g(q), q-p\rangle \leq 0 \Rightarrow F(q) \leq C \cdot F(p) \tag{1}$$

and $g(q; \mathbf{x})$ is defined with closed form expression under different circumstances (See Zhu et al. (2022)).

Algorithm 1 No-Regret (D_m, ϵ, σ^2, ξ, η)

```
1: Input: $D_m = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$: The dataset of $m$ samples in $d$-dimensional space;
   $\epsilon$: The fraction of samples that may be corrupted;
   $\sigma^2$, An upper bound on the spectral norm of the true covariance matrix;
   $\eta \in (0,1)$: A step size parameter for the weight updates.
2: Set $d_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ for all $i, j \in [m]$          ▷ Compute pairwise distances
3: Remove all $\mathbf{x}_i$ with
   $\left|\{j \in [m] : d_{i,j} > \Omega(\sqrt{d\log(m)})\}\right| > 2\epsilon m$          ▷ Remove outliers
4: Let $\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_{m'}$ be the remaining samples
5: Initialize: $q_i^{(0)} = \frac{1}{m'}, i \in [m']$          ▷ Initialize weights
6: for $k = 0, 1, \dots$ do
7:     if $\|\mathrm{Cov}_{q^{(k)}}(\mathbf{x})\|_2 \leq \xi$ then
8:         Break          ▷ Check spectral norm
9:     else
10:        for $i \in [m']$ do
11:            Compute $g_i^{(k)} = g(q^{(k)}; \mathbf{x}_i)$          ▷ Compute generalized quasi-gradients
12:            $\tilde{q}_i^{(k+1)} = q_i^{(k)} \cdot \left(1 - \frac{\eta\epsilon}{2\sigma^2 d} \cdot g_i^{(k)}\right)$          ▷ Update weights $\tilde{q}$
13:        end for
14:        Update $q^{(k+1)} = \arg\min_{q\in\Delta_{m',\epsilon}} D_{KL}(q\|\tilde{q}^{(k+1)})$          ▷ Project onto the feasible set, update weights $q$
15:        where $\Delta_{m',\epsilon} = \{q \mid \sum q_i = 1, q_i \leq \frac{1}{(1-\epsilon)m'}\}$
16:    end if
17: end for
18: Output: $\mathbb{E}_{q^{(k)}}[\mathbf{x}] = \sum_{i=1}^{m'} q_i^{(k)}\mathbf{x}_i$, $\{q_i\}$, $\{\mathbf{x}_i\}$          ▷ Robust mean estimate
```

Algorithm 2 Filtering (D_m, ξ)

```
1: Input: $D_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$,
   threshold for termination $\xi$.
2: Initialize $q_i^{(0)} = \frac{1}{m}$, $i \in [m]$                                  ▷ Initialize weights
3: for $k = 0, 1, \ldots$ do
4:     if $\|\mathrm{Cov}_{q^{(k)}}(\mathbf{x})\|_2 \le \xi$ then
5:         Break                                                         ▷ Check spectral norm
6:     else
7:         for $i \in [m]$ do
8:             Compute
               $g_i^{(k)} = g(q^{(k)}; \mathbf{x}_i)$                    ▷ Compute generalized quasi-gradients
9:             $\tilde{q}_i^{(k+1)} = q_i^{(k)} \cdot \left(1 - \frac{g_i^{(k)}}{\max_{j\in[m]} g_j^{(k)}}\right)$        ▷ Update weights $\tilde{q}$
10:        end for
11:        Update $q^{(k+1)} = \mathrm{Proj}_{\Delta_m}^{KL}(\tilde{q^{k+1}}) = \tilde{q}^{(k+1)} / \sum_{i\in[m]} \tilde{q}_i^{(k+1)}$      ▷ Update weights $q$
           where $\Delta_m = \{q \mid \sum q_i = 1\}$
12:        Remove samples with $q_i^{(k+1)} = 0$                          ▷ Prune zero-weight samples
13:    end if
14: end for
15: Output: $\mathbb{E}_{q^{(k)}}[\mathbf{x}] = \sum_{i=1}^{m'} q_i^{(k)} \mathbf{x}_i$, $\{q_i\}$, $\{\mathbf{x}_i\}$        ▷ Robust mean estimate
```

Defense objective

The defense objective aims to design a robust protocol that preserves an optimal rate of convergence even with malicious clients involved. By leveraging algorithms like No-Regret and Filtering, the protocol iteratively updates weights assigned to each client's data and removes outliers that deviate significantly from the norm. These algorithms compute generalized quasi-gradients to adjust the weights, effectively mitigating the influence of corrupted samples. This ensures that, throughout training, the model's parameters stay close to the optimal solution within a set statistical error bound, despite adversarial interference. Additionally, the protocol establishes a "breakdown point"—the maximum proportion of corrupted clients the system can tolerate before the estimation becomes unbounded—thus securing the protocol's resilience up to a critical limit. Through these robust mean estimation methods, the defense strategy maintains the integrity and performance of the model, ensuring convergence rates are preserved even in adversarial settings.

Performance awareness

Definition 3.2 (Performance aware objective)**.** Given M objectives $g_m(\theta)$ for $m = 1, \ldots, M$, we define the performance aware objective as

$$g(\theta) = \max_{m\in[M]} g_m(\theta). \tag{2}$$

We term this as the performance-aware objective since the objective to be minimized $g(\theta)$ dynamically chooses the objective with the largest value (i.e. the worst-performing objective).

Definition 3.3 (Smoothness and smoothing function (Lin et al., 2024))**.** A differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if the gradient of g satisfies the condition

$$\|\nabla g(\theta_1) - \nabla g(\theta_2)\| \le L\|\theta_1 - \theta_2\|, \ \forall \theta_1, \theta_2 \in \mathbb{R}^d. \tag{3}$$

For a continuous function $g : \mathbb{R}^d \to \mathbb{R}$, we call $g_\mu : \mathbb{R}^d \to \mathbb{R}$ a smoothing function of g if for any $\mu > 0$, g_μ is continuously differentiable in $\mathbb{R}^d$ and satisfies the conditions

- $\lim_{\phi\to\theta,\mu\downarrow 0} g_\mu(\phi) = g(\theta)$.
- There exists constants L and $\alpha > 0$ independent of μ, such that g_μ is $\left(L + \frac{\alpha}{\mu}\right)$-smooth.

Note that the smoothing function of the performance aware objective function can be given by Lin et al. (2024)

$$g_\mu(\theta) := \mu \log\left(\sum_{m=1}^{M} \exp\left(\frac{g_m(\theta)}{\mu}\right)\right), \tag{4}$$

where $\mu > 0$ is the smoothing parameter that controls the smoothness of g_μ, $\log$ is the natural logarithm function. For the performance aware objective g_μ is $\left(L + \frac{\alpha}{\mu}\right)$-smooth with $L := \sum_{m=1}^{M} L_m$ and $\alpha := \sum_{m=1}^{M} L_{m,1}^2$, where L_m and $L_{m,q}$ are the smoothness and Lipchitz constants of objective g_m, respectively.

4 Method

In this section, we introduce our proposed incorporation of performance awareness into Byzantine-robust aggregation schemes. To this end, we first introduce the performance-aware objective as follows. Consider the setting where the model to be optimized is given by θ, local client objectives are given by $f_i(\theta)$ for $i \in \{1, \dots, m\}$, where m is the number of clients. Then, as introduced in Section 3, the smoothed performance-aware objective for this setting can be naturally given by

$$f_{\text{PA}}(\theta) = \mu \log \left(\sum_{i=1}^{m} \exp\left(\frac{\lambda_i \left(f_i(\theta) - f_i^*\right)}{\mu} \right) \right), \tag{5}$$

where $f_i^* = \operatorname{argmin}_\theta f_i(\theta)$, λ_i are preference coefficients (probably based on prior knowledge on f_i) with $\sum_i^m \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i \in \{1, \dots, m\}$, and μ is a temperature parameter to be determined. The intuition of the objective is, given a set of objectives, give a larger weight in optimizing the objectives that have a larger optimality gap $f_i(\theta) - f_i^*$, i.e. the objectives that are performing poorly. Note that we also take into consideration the preference we give to each optimality gap λ_i. The focus on the poorly performing objectives increases when the temperature parameter μ is reduced. The gradient of the objective f_{PA} can be given by

$$\nabla f_{\text{PA}}(\theta) = \sum_{i=1}^{m} \lambda_i \frac{\exp(y_i)}{\sum_{j=1}^{m} \exp(y_j)} \nabla f_i(\theta), \tag{6}$$

where $y_i = \frac{\lambda_i}{\mu}\left(f_i(\theta) - f_i^*\right)$ for all $i = \{1, \dots, m\}$. Thus, it can be seen that this gradient is essentially a weighted combination of the gradients, weighted by the preference weights λ_i, and the softmax weights of the weighted optimality gaps y_i. If no particular preference is not known apriori, one can set $\lambda_i = 1/m$.

On the other hand, there are existing Byzantine-robust aggregators that, in general, calculate a weighted combination of the local gradients of the clients, based on some measure of 'maliciousness' of the client. Specifically, these aggregators will assign a larger weight for gradients of clients that seem not malicious, and a smaller (or zero) weight for that of seemingly malicious clients. More concretely, this kind of aggregation can be given by

$$g_{\text{BR}}(\theta) = \sum_{i=1}^{m} q_i g_i(\theta), \tag{7}$$

where g_i is the update difference sent by clients to server, $\sum_{i=1}^{m} q_i = 1$, $q_i \geq 0$, and $q_i = 0$ for possible malicious clients. Now, note that q_i's can be viewed as preference given to client i (or equidistantly objective f_i) based on their maliciousness. This allows us to seamlessly integrate the performance-aware gradient, to obtain the performance-aware Byzantine-robust gradient, given by

$$g_{\text{PABR}}(\theta) = \sum_{i=1}^{m} q_i \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)} g_i(\theta), \tag{8}$$

where $z_i = \frac{q_i}{\mu}\left(f_i(\theta) - f_i^*\right)$. While this gradient seems to combine both the advantages of ∇f_{PA} and g_{BR}, the weight $q_i \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)}$ applied in front of each gradient tend to make the norm of g_{BR} be too small, which tends to make the convergence slow, and the test performance of the obtained model poor (Wei & Hu, 2024). Thus, we obtain the final PAwBRoFL gradient, adjusting for the magnitude of the gradient as follows:

$$\hat{g}(\theta) = \|g_{\text{BR}}(\theta)\| \frac{g_{\text{PABR}}(\theta)}{\|g_{\text{PABR}}(\theta)\|}. \tag{9}$$

The intuition behind the PAwBRoFL gradient is that, we are preserving the desirable descent direction given by g_{PABR} while retaining the gradient magnitude of g_{BR} which is desirable for fast convergence and test performance. We summarize the complete PAwBRoFL algorithm in Algorithm 3.

Algorithm 3 PAwBRoFL

```
1: Input: Initialize parameter $\theta_0 \in \Theta$, step size $\eta_t$, local model update interval $H$, and total
   iteration $T$ .
2: Server: send $\theta_0$ to all clients.
3: for $t = 1, 2 \ldots, T$ do
4:     for $i \in [m]$ do
5:         Client $i$: Update local parameter $\theta_t^{(i)} = \theta_{t+1}^{(i)} - \eta \nabla f_i(\theta_t^{(i)})$
6:         if $t \mod H = 0$ then
7:             send local model update $g_i(\theta_t) = \theta_t^{(i)} - \theta_{t-H}^{(i)}$ to server.
8:         end if
9:     end for
10:    if $t \mod H$ then
11:        Server:
12:        $g_{\text{BR}}(\theta_t), \{q_i\}, \{g_i(\theta_t)\} \leftarrow$ BR-Aggregator($\{g_i(\theta_t)\}$)    ▷ e.g., No-Regret, Filtering
13:        Compute $\hat{g}(\theta_t)$ given in (9)
14:        Update $\theta_{t+1} = \theta_t - \eta_t \hat{g}(\theta_t)$
15:        Send $\theta_{t+1}$ clients, setting $\theta_{t+1}^{(i)} = \theta_{t+1}$
16:    end if
17: end for
18: Output: $\theta_T$
```

5 Numerical experiments

5.1 Experiment setup

To evaluate the performance of our proposed Performance-Aware Byzantine-Robust Federated Learning (PAwBRoFL) framework, we conducted experiments using the well-established MNIST and Fashion-MNIST datasets. Both datasets were divided into training and testing sets according to their standard configurations. The training data was distributed among 100 simulated clients to mimic a federated learning environment. Each client was assigned an equal portion of the training data to simulate decentralized data storage.

Model Architecture and Training Configuration. We implemented a lightweight convolutional neural network (CNN) with two convolutional layers followed by two fully connected layers. This architecture was chosen to balance computational efficiency and model performance, making it suitable for deployment in federated settings. Clients trained their local models using stochastic gradient descent (SGD) with a learning rate of 0.01. The models were trained for 5 local epochs with a batch size of 10 before sending updates to the server.

The server aggregated the client updates using both the proposed PAwBRoFL method and a baseline vanilla federated learning (FL) approach. PAwBRoFL incorporated client sub-optimality metrics to weigh updates, using different temperature parameter (μ) values to evaluate performance under varying levels of performance awareness.

Attack Simulation. To evaluate the robustness of our PAwBRoFL framework, we conducted simulations of Byzantine adversarial attacks. Specifically, we implemented model poisoning attacks, including Distributed Backdoor Attacks (DBA), following the methodology outlined by Xie et al. (2019).

The attacks were generated by manipulating a subset of client updates before aggregation at the server. Malicious clients adjusted their local updates to introduce targeted biases into the global model, either to degrade overall performance (untargeted poisoning) or to embed specific hidden functionalities (backdoor attacks).

We varied the strength of the attacks by controlling the proportion of malicious clients in the system, ranging from 10% to 40% of the total number of clients. Additionally, we adjusted the scale of the injected malicious updates to evaluate the framework's resilience against subtle versus aggressive attack strategies.

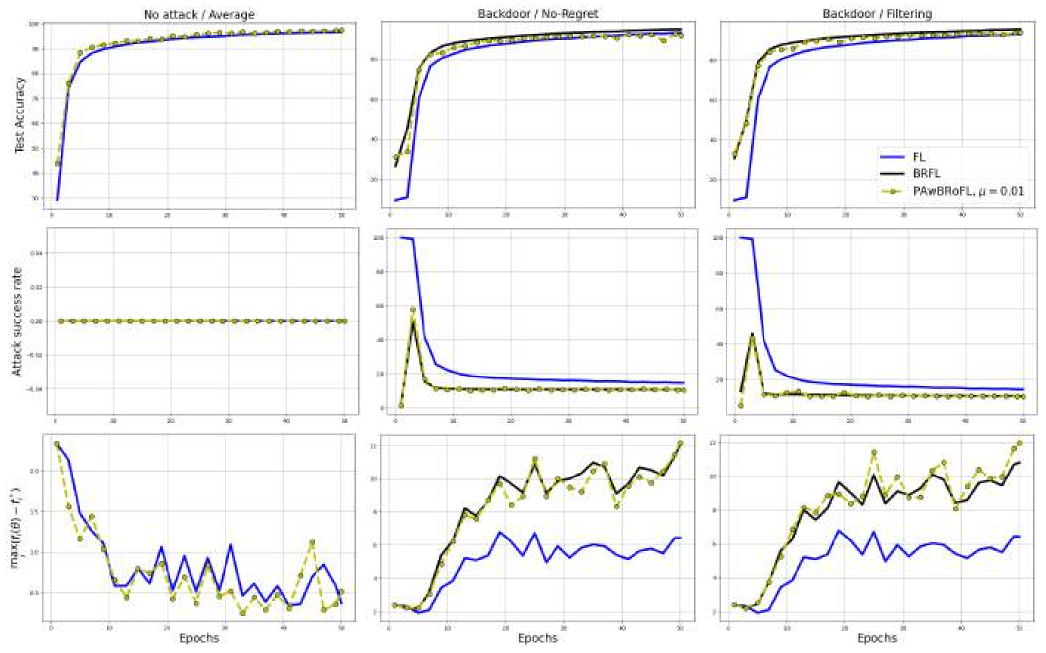

Figure 2: **MNIST experiments.** The first row represent the test accuracy performance (higher the better) of vanilla FL (solid blue), vanilla BRFL (solid black), and the proposed PAwBRoFL method (dashed yellow with circle marker). Second row represent the attack success rates (if any, lower the better) which indicate the robustness of each method to attacks. Third row gives the maximum local optimality gap (lower the better), the objective given in 2.

For benchmarking, we compared PAwBRoFL's performance against standard Byzantine-robust FL algorithms using No-Regret and Filtering aggregators, as described in the original research by Zhu et al. (2023). This comprehensive setup allowed us to assess the effectiveness of PAwBRoFL in mitigating adversarial influence under varying levels of attack severity.

Metrics for Evaluation. The metrics used to assess the experiments were test accuracy, attack success rate, and the worst client optimality gap. The test accuracy was measured after each training epoch to monitor the model's performance over time, while the attack success rate was tracked to evaluate the framework's resilience against adversarial manipulation. Worst client optimality gap was computed by taking the difference between the current client performance and the (approximate) best client objective value. Best client objective value was first computed by running local updates for a longer number of iterations (250 epochs).

5.2 Experiment Results

We provide experiment results for our PAwBRoFL implementation in Figures 2 and 3. We use $\mu = 0.01$ as the choice of temperature parameter for PAwBRoFL. Let's first Consider the MNIST experiments in Figure 2. We can see that PAwBRoFL outperforms vanilla FL slightly in no attack case (first column), in terms of the test accuracy (first row). Furthermore, PAwBRoFL seems to perform well in terms of the worst performing optimality gap in most rounds, compared to that of vanilla FL (third row). There is zero attack success rate (second row) for both methods in this setting, since no attack is used for this setup. Next, considering the backdoor attack case (last two columns), it can be seen that PAwBRoFL slightly outperform vanilla FL, and slightly under perform BRFL, in terms of test accuracy. In the No-regret aggregator case, PAwBRoFL seems to slightly outperform BRFL, but heavily under perform vanilla FL in terms of worst optimality gap. This suggest that applying Byzantine robust approaches in general tend to enlarge the worst performance gap, which is expected since Byzantine robust approaches largely excludes suspected malicious clients from the learning process. Furthermore, both BRFL and PAwBRoFL outperform vanilla FL in terms of attack success rates in attack scenarios, which is also expected. Overall, the advantage

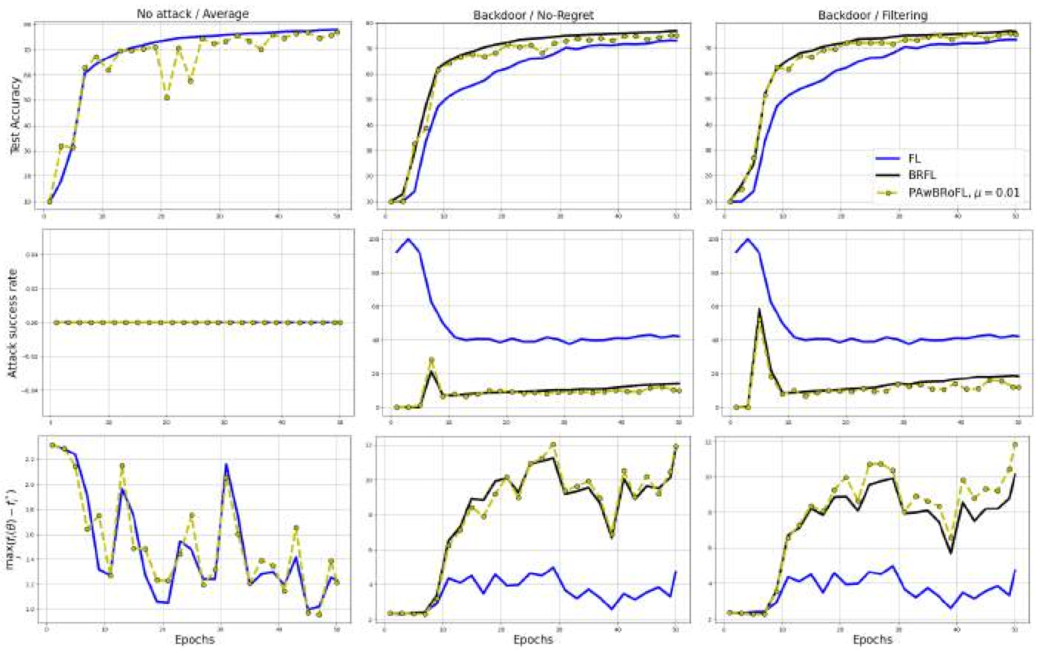

Figure 3: **Fashion-MNIST experiments.** The first row represent the test accuracy performance (higher the better) of vanilla FL (solid blue), vanilla BRFL (solid black), and the proposed PAwBRoFL method (dashed yellow with circle marker). Second row represent the attack success rates (if any, lower the better) which indicate the robustness of each method to attacks. Third row gives the maximum local optimality gap (lower the better), the objective given in 2.

of PAwBRoFL seems to be in better test accuracy and attack success rate compared to vanilla FL in attack/no attack settings, and better fairness (in terms of worst performing client) in the no attack case.

Next, let's consider Fashion-MNIST experiment results given in Figure 3. It can be seen that, unlike in MNIST dataset experiments, PAwBRoFL seems to under perform vanilla FL in the no-attack case. Furthermore, the test accuracy performance seems more sporadic across communication rounds, compared to that in MNIST experiments. However, unlike in MNIST experiments, the attack success rate is better for PAwBRoFL compared to that of both BRFL and vanilla FL, and the vanilla FL seems to suffer significantly larger attack success rates compared to that in MNIST experiments. In Fashion-MNIST experiments too, it can be seen that the worst performing optimality gap is still smaller for vanilla FL (compared to Byzantine robust methods). Overall, PAwBRoFL seems to outperform other methods in terms of attack success rate, while under performing in test accuracy and worst optimality gap. It is unclear why PAwBRoFL slightly under perform vanilla FL in terms of worst optimality gap, but we suspect this is due to choice of μ being sub-optimal for this dataset. This shows the sensitivity of μ parameter for different problem setups. Further tuning of this parameter seems to be needed for Fashion-MNIST experiments.

6 Discussion

In this project, we explored the feasibility of incorporating some notion of performance-awareness into Byzantine-robust federated learning setting. We derived the gradient of the corresponding server-side aggregated gradient analytically, and provided experiment results for the performance of the proposed PAwBRoFL method, compared to the vanilla Byzantine-robust methods and vanilla federated learning. It can be seen that merits of PAwBRoFL seems to vary depending on the dataset (hence the experiment setup). For MNIST dataset, we see that the performance of PAwBRoFL seems to be better in terms of test accuracy, attack success rate, and worst performing optimality gap compared to vanilla FL. However, PAwBRoFL seems to under perform BRFL slightly in terms of test accuracy and worst performing optimality gap in the case of filtering aggregation while performing better in no-regret aggregation, which suggests that the method of BR aggregation affect the performance of performance awareness component of PAwBRoFL. Considering Fashion-MNIST dataset, we can see that PAwBRoFL seems perform well in terms of attack success rate, but perform poorly in terms of test accuracy and worst optimality gap. We suspect this might be due to sub-optimal choice of hyper-parameters, and further hyper-parameter tuning might be needed for Fashion-MNIST dataset. We also note that when the n μ parameter is very large, PAwBRoFL is close to vanilla Byzantine-robust methods. It would be interesting to see whether the best of both worlds (better accuracy while better attack resistance) can be achieved by varying the μ parameter throughout the training, rather than keeping it constant. Also, it would be interesting to look into other metrics of performance of validation loss, and gradient norm, in addition to the training loss gap, which might affect the test accuracy more favorably.

REFERENCES

Ranwa Al Mallah, David Lopez, Godwin Badu-Marfo, and Bilal Farooq. Untargeted poisoning attack detection in federated learning via behavior attestation. *IEEE Access*, 2023.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients, 2018. URL `https://arxiv.org/abs/1803.09877`.

Yongheng Deng, Feng Lyu, Ju Ren, Yi-Chao Chen, Peng Yang, Yuezhi Zhou, and Yaoxue Zhang. Fair: Quality-aware federated learning with precise user incentive and model aggregation. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10. IEEE, 2021.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. In *International Conference on Machine Learning*, pp. 999–1008. PMLR, 2017.

Sam Hopkins, Jerry Li, and Fred Zhang. Robust and heavy-tailed mean estimation made simple, via regret minimization. *Advances in Neural Information Processing Systems*, 33:11902–11912, 2020.

Nikola Konstantinov and Christoph Lampert. Robust learning from untrusted sources, 2019. URL `https://arxiv.org/abs/1901.10310`.

Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.

Shenghui Li, Edith C.-H. Ngai, and Thiemo Voigt. An experimental study of byzantine-robust aggregation schemes in federated learning. *IEEE Transactions on Big Data*, pp. 1–13, 2023. doi: 10.1109/TBDATA.2023.3237397.

Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Smooth tchebycheff scalarization for multi-objective optimization. *arXiv preprint arXiv:2402.19078*, 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

Thuy Dung Nguyen, Tuan Nguyen, Phi Le Nguyen, Hieu H Pham, Khoa D Doan, and Kok-Seng Wong. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions. *Engineering Applications of Artificial Intelligence*, 127:107166, 2024.

Sangwon Seo, Jaewook Lee, Haneul Ko, and Sangheon Pack. Performance-aware client and quantization level selection algorithm for fast federated learning. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1892–1897. IEEE, 2022.

Yake Wei and Di Hu. Mmpareto: boosting multimodal learning with innocent unimodal assistance. In *International Conference on Machine Learning*, 2024.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*, 2019.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

Banghua Zhu, Jiantao Jiao, and Jacob Steinhardt. Robust estimation via generalized quasi-gradients. *Information and Inference: A Journal of the IMA*, 11(2):581–636, 2022.

Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I. Jordan. Byzantine-robust federated learning with optimal statistical rates. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 3151–3178. PMLR, 25–27 Apr 2023. URL `https://proceedings.mlr.press/v206/zhu23b.html`.