PAwBRoFL : Exploring Performance Aware Byzantine-Robust Federated Learning

CSCI 6962 Fall 2024 Group Project

Heshan Fernando, Zhengjie Xiong, Aarnav Patel, Shane Stoll

Content

- Introduction
- Motivation
- Proposed Framework
- Related Work
- Technical Foundations
- Methodology
- Experimental Setup
- Results
- Discussion
- References

Introduction

- What is Federated Learning (FL)?
  - Decentralized ML training; data remains to local devices.
  - Key advantages: Enhanced privacy and reduced data transfer.

- Challenges in FL:
  - Vulnerable to Byzantine adversaries: malicious or unreliable clients corrupting updates.

- Our objective:
  - Introduce **PAwBRoFL**: Integrating performance-awareness into Byzantine-robust FL.

Motivation

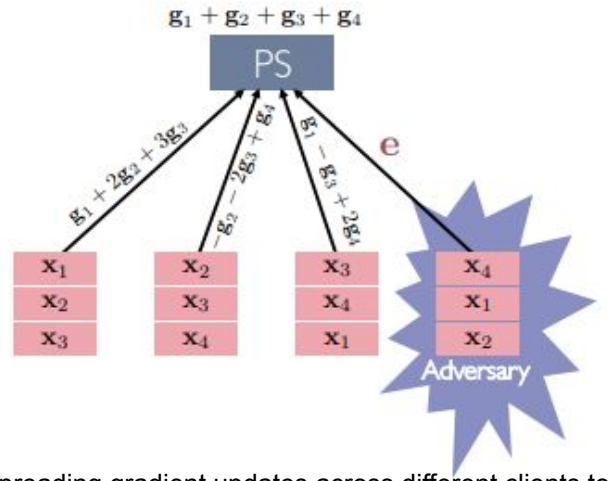
- Byzantine adversaries can:
  - Disrupt global models through malicious updates.
  - Lead to performance degradation across clients.

- Current Limitations:
  - Existing Byzantine-robust methods focus solely on attack resistance.
  - Lack of client performance-awareness in global model updates.

Related work (Byzantine)

3 main strategies to counter Byzantine attacks

- Redundancy
  - Ensure that malicious clients effects are minor compared to good clients
- Trusting a few select clients each iteration
  - Compare gradients to known non-adversarial data/clients[2]
- Robust Aggregation Algorithms



Spreading gradient updates across different clients to reduce the impact of an adversarial client[1]

[1]Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients
[2] Nikola Konstantinov and Christoph Lampert. Robust learning from untrusted sources, 2019

Related work (Performance Awareness)

- In this project, "performance awareness" refers to the process of optimizing the worst-performing objective among a set of objectives to be optimized (details provided later) (Lin et al. 2024).
- In federated learning, some studies aggregate model parameters based on client performance metrics like
  - Quality (Deng et al., 2021)
  - Computing power (Seo et al., 2022),

  but these performance concepts are beyond the scope of this project.

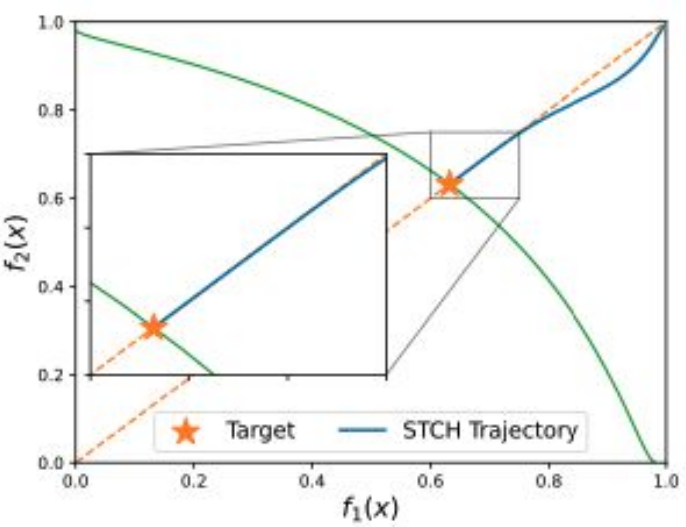
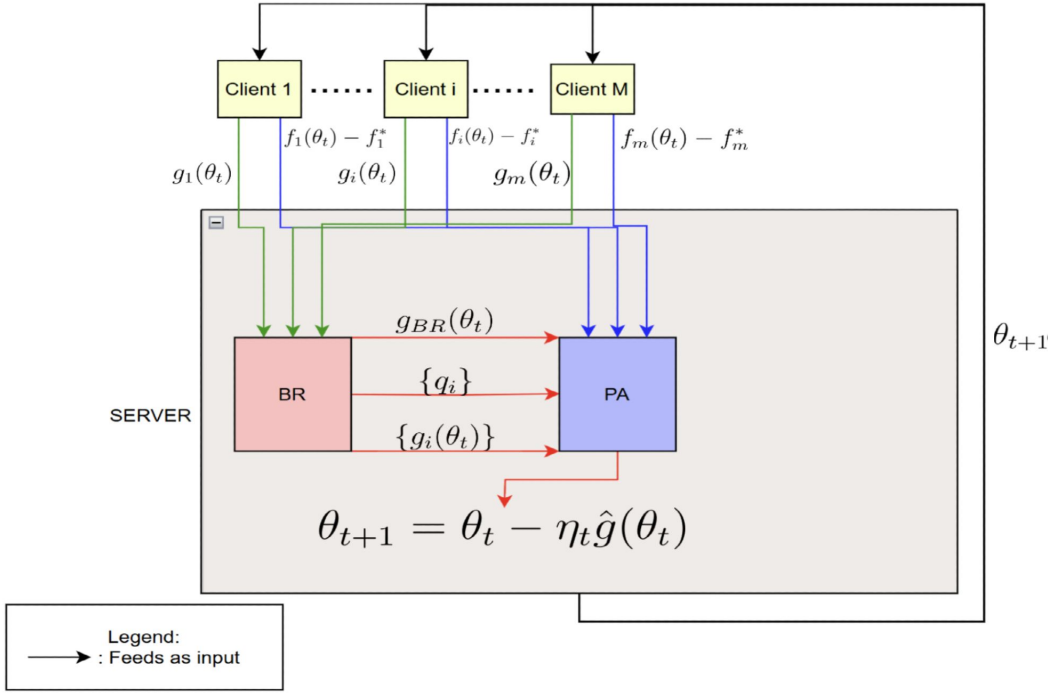


Image from Lin, Xi, et al. "Smooth Tchebycheff Scalarization for Multi-Objective Optimization." arXiv preprint arXiv:2402.19078(2024).

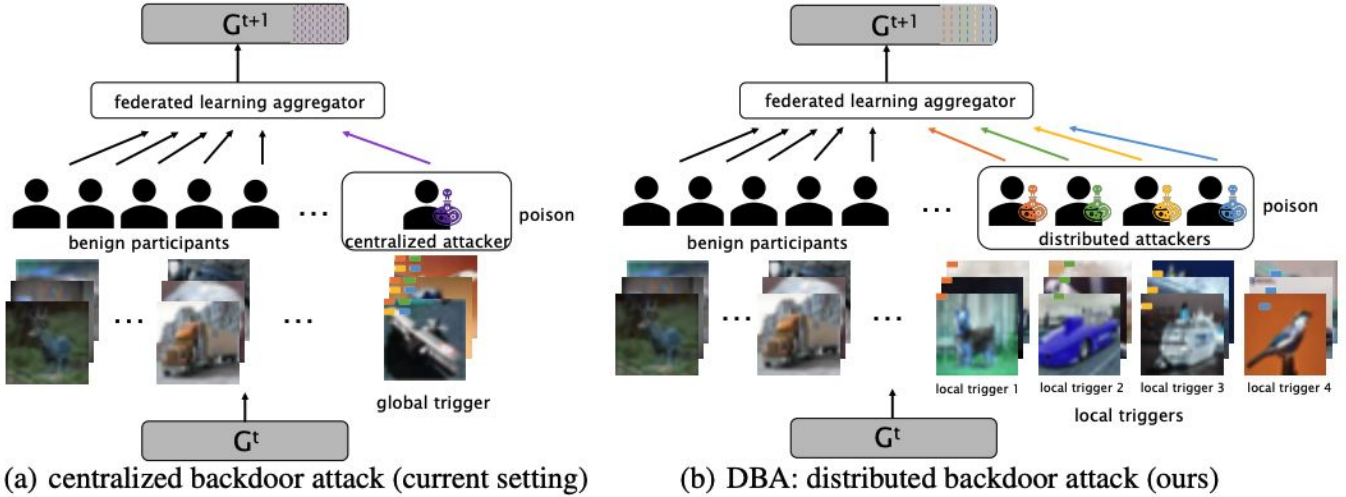
Proposed Framework

- PAwBRoFL:
  - Combines **performance awareness** with **Byzantine robustness**



Technical foundations

- Model poisoning and attack:
  - **Distributed Backdoor Attack** (DBA) Xie et al. (2019)



Technical foundations (Cont.)

- Robust Aggregation
  - **No-Regret Algorithm** Hopkins et al. (2020)
  - **Filtering Algorithm** Diakonikolas et al. (2017)

Algorithm 1 No-Regret $(D_m, \epsilon, \sigma^2, \xi, \eta)$

Input: $D_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$: The dataset of m samples in d-dimensional space;
ϵ: The fraction of samples that may be corrupted;
σ^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.
Set $d_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ for all $i, j \in [m]$ ▷ Compute pairwise distances
Remove all $\mathbf{x}_i$ with
$\left|\{j \in [m] : d_{i,j} > \Omega(\sqrt{d \log(m)})\}\right| > 2\epsilon m$ ▷ Remove outliers
Let $\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_{m'}$ be the remaining samples
Initialize: $q_i^{(0)} = \frac{1}{m'}, i \in [m']$ ▷ Initialize weights
for $k = 0, 1, \ldots$ **do**
  if $\|\mathrm{Cov}_{q^{(k)}}(\mathbf{x})\|_2 \leq \xi$ **then**
    Break ▷ Check spectral norm
  else
    for $i \in [m']$ **do**
      Compute $g_i^{(k)} = g(q^{(k)}; \mathbf{x}_i)$ ▷ Compute generalized quasi-gradients
      $\tilde{q}_i^{(k+1)} = q_i^{(k)} \cdot \left(1 - \frac{\eta\epsilon}{2\sigma^2 d} \cdot g_i^{(k)}\right)$ ▷ Update weights $\tilde{q}$
    end for
    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
    where $\Delta_{m',\epsilon} = \{q \mid \sum q_i = 1, q_i \leq \frac{1}{(1-\epsilon)m'}\}$
  end if
end for
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

From Zhu et al. (2023)

Technical foundations (Cont.)

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

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

A differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is L-smooth if the gradient of g satisfies the condition

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

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.

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

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

where $\mu > 0$ is the smoothing parameter that controls the smoothness of g_μ.

Methodology

Assume the model to be optimized is given by θ, local client objectives are given by $f_i(\theta)$ for $i \in \{1, \ldots, m\}$. Consider

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

where $f_i^* = \operatorname{argmin}_\theta f_i(\theta)$, λ_i are preference coefficients. The gradient of the objective f_{PA} can be given by

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

where $y_i = \frac{\lambda_i}{\mu} \left(f_i(\theta) - f_i^* \right)$. Existing Byzantine-robust aggregators calculate a weighted combination of the local gradients of the clients:

$$g_{\mathrm{BR}}(\theta) = \sum_{i=1}^{m} q_i g_i(\theta)$$

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.

Methodology (Cont.)

Integrate the performance-aware gradient, to obtain the performance-aware Byzantine-robust gradient, given by

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

where $z_i = \frac{q_i}{\mu}\left(f_i(\theta) - f_i^*\right)$. 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. Thus, we obtain the final PAwBRoFL gradient, adjusting for the magnitude of the gradient as follows:

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

Methodology (Cont.)

Algorithm 3 PAwBRoFL

Input: Initialize parameter $\theta_0 \in \Theta$, step size η_t, local model update interval H, and total iteration T .
Server: send θ_0 to all clients.
for $t = 1, 2 \ldots, T$ **do**
  for $i \in [m]$ **do**
    Client i: Update local parameter $\theta_t^{(i)} = \theta_{t+1}^{(i)} - \eta \nabla f_i(\theta_t^{(i)})$
    if $t \mod H = 0$ **then**
      send local model update $g_i(\theta_t) = \theta_t^{(i)} - \theta_{t-H}^{(i)}$ to server.
    end if
  end for
  if $t \mod H$ **then**
    Server:
    $g_{\text{BR}}(\theta_t), \{q_i\}, \{g_i(\theta_t)\} \leftarrow$ BR-Aggregator$(\{g_i(\theta_t)\})$ ▷ e.g., No-Regret, Filtering
    Compute $\hat{g}(\theta_t)$ given in (9)
    Update $\theta_{t+1} = \theta_t - \eta_t \hat{g}(\theta_t)$
    Send θ_{t+1} clients, setting $\theta_{t+1}^{(i)} = \theta_{t+1}$
  end if
end for
Output: θ_T

Experimental Setup

Utilizated MNist & Fashion MNist Datasets evenly spread across 100 Clients

Attack Type: Distributed Backdoor Attack

- varied in both number of malicious clients, as well as aggressiveness of attacks

Evaluation Metrics:

- Test Accuracy
- Attack Success Rates
- Client Optimality Gap
  - The difference between the theoretically best achievable value had there been no attacks, and current model's performance

3 scenarios: No attack, Attack w/ No Regret, Attack w/ Filtering

Results (MNIST)

vanilla Fed Learning (solid blue),

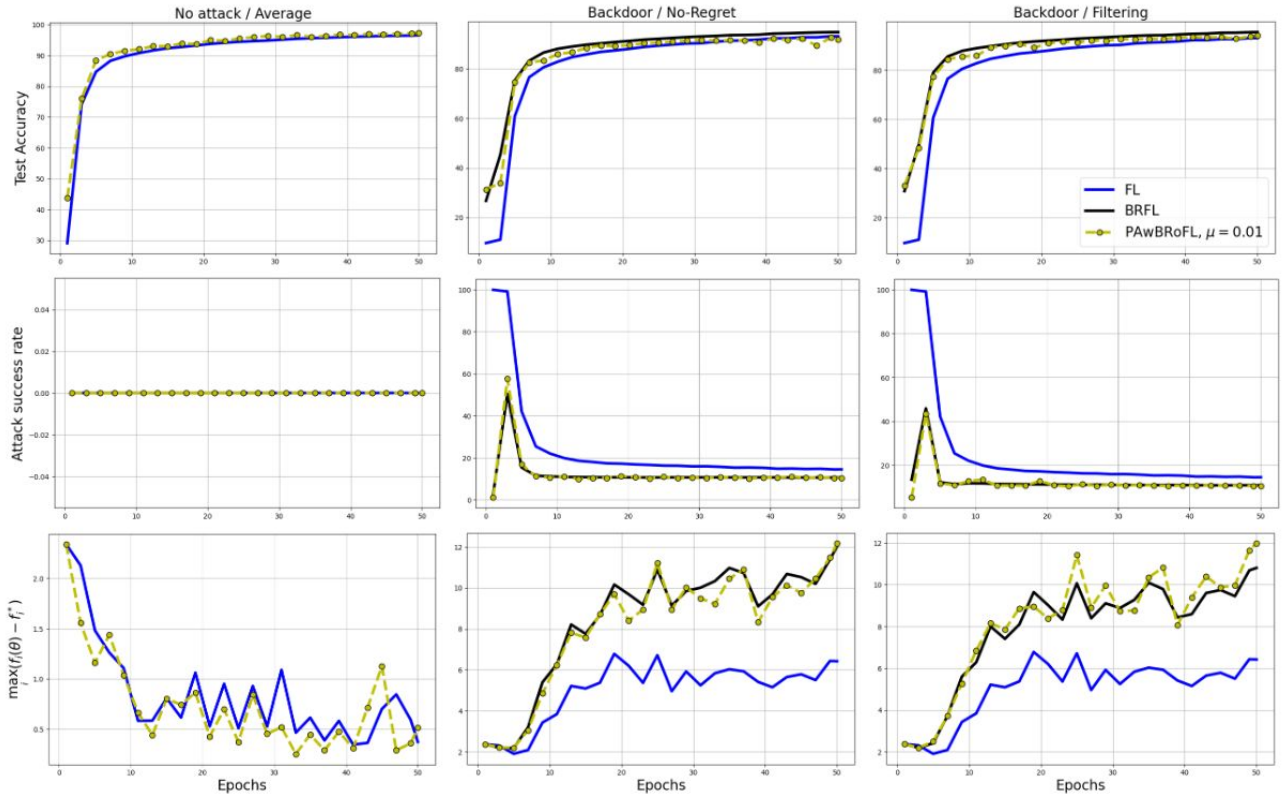
vanilla BR Fed Learning (solid black),

PAwBRoFL method (dashed yellow with circle marker).

Top Row: test accuracy performance (HIGH)

Middle row: Attack success rate (LOW)

Bottom row: maximum local optimality gap (LOW



Results (Fashion-MNIST)

vanilla Fed Learning (solid blue),

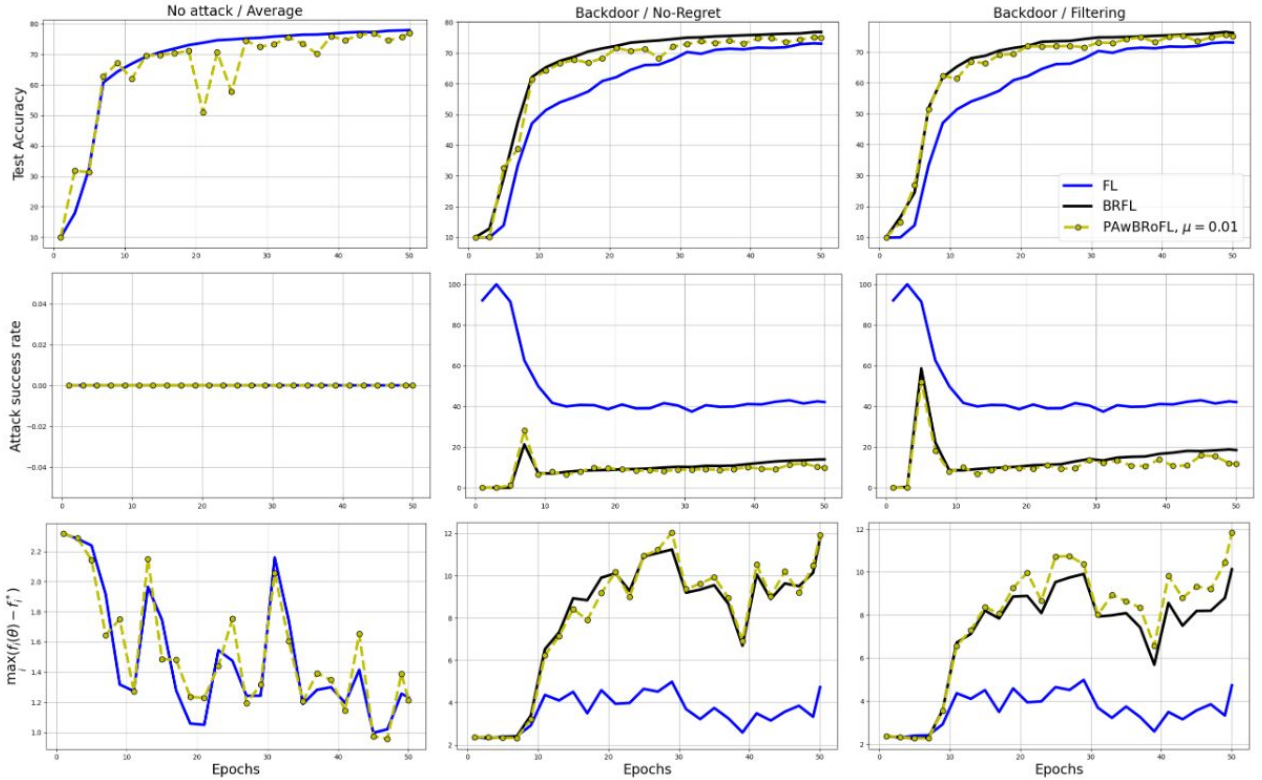
vanilla BR Fed Learning (solid black),

PAwBRoFL method (dashed yellow with circle marker).

Top Row: test accuracy performance (HIGH)

Middle row: Attack success rate (LOW)

Bottom row: maximum local optimality gap (LOW)



Results (Fashion-MNIST)

vanilla Fed Learning (solid blue),

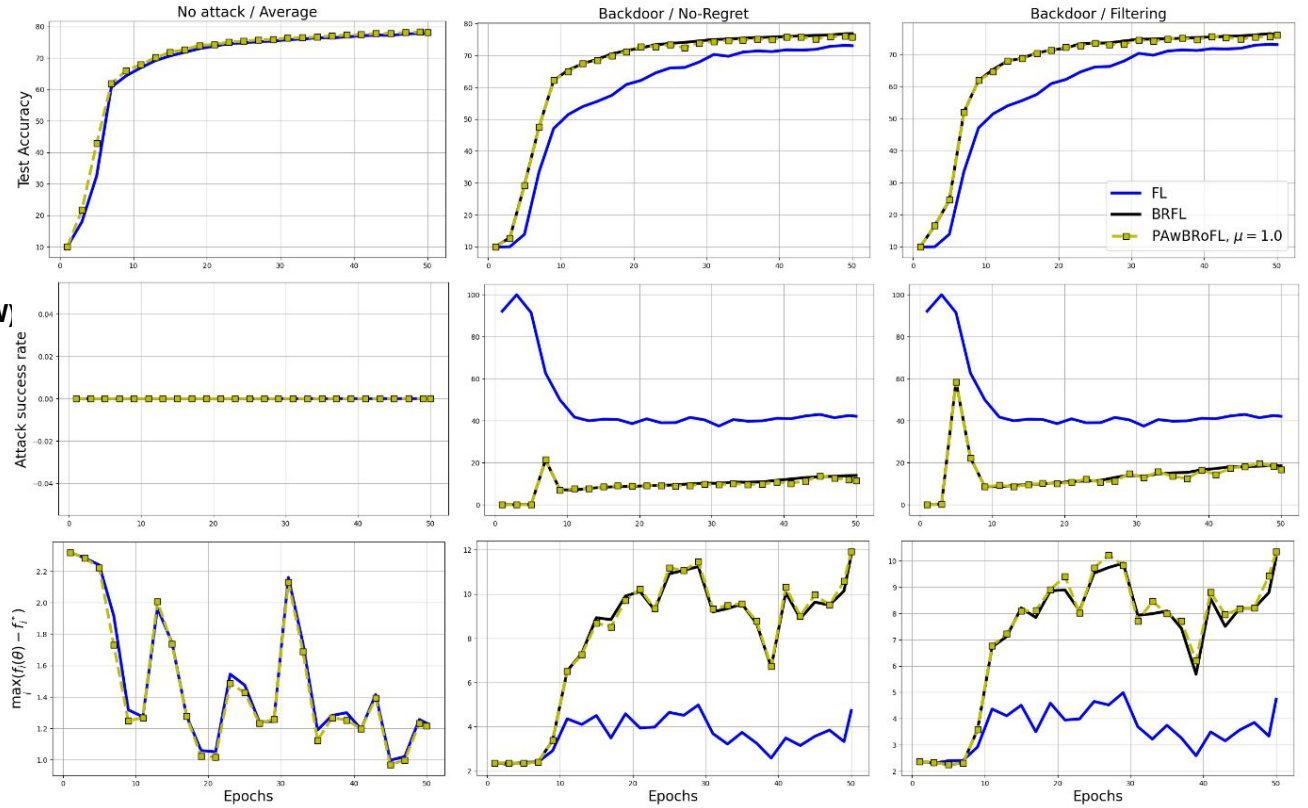
vanilla BR Fed Learning (solid black),

PAwBRoFL method (dashed yellow with circle marker).

Top Row: test accuracy performance (HIGH)

Middle row: Attack success rate (LOW)

Bottom row: maximum local optimality gap (LOW)



Discussion

PAwBRoFL gives a basic foundation for integrating performance awareness with Byzantine robustness in FL setting

PAwBRoFL's performance varies between datasets

- Hyper-parameter tuning?
- Dynamic temperature parameterization?

Exploring other metrics of performance (EX: Validation loss gap, gradient norm) would be interesting future work

References

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.

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

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`.

See report for complete list of reference.