Federated learning with MedMNIST

Mohammed Raeesul Irfan Riaz Ahmed
Rochester Institute of Technology
mr6248@rit.edu

Teja Kiran Reddy Sirivella
Rochester Institute of Technology
ts7244@rit.edu

Brady Lax
Rochester Institute of Technology
brl7498@rit.edu

I. Experiments

A. Experimental Setup

We conducted the experiments on the Granger system using an NVIDIA Tesla P4 GPU (architecture: sm_61). The training environment was managed using a Conda environment with PyTorch, CUDA, and the Flower framework. Dependencies were installed via Conda to leverage GPU acceleration efficiently.

The PathMNIST dataset used for the experiment contains:

- 89,996 training samples
- 10,004 validation samples
- 7,180 test samples

In the previous phase, we utilized the TinyViT-5M-224.dist_in22k model, which has approximately 5 million parameters. Although this model provided a good balance between efficiency and performance, it required over 100 rounds to converge, resulting in an impractically long training time. To address this issue, we transitioned to a Convolutional Neural Network (CNN) model, which offers faster convergence. Additionally, leveraging CUDA on the Tesla P4 GPU allowed us to accelerate training further, significantly reducing the overall runtime for our experiments.

B. Experiment Design

Our goal was to evaluate the convergence process with two scenarios:

- **With Attack:** 50% of the labels are flipped for 3 out of 10 clients to simulate a malicious data poisoning attack.
- **With Defense:** With Defense: In this scenario, we implemented a Trust and Reputation mechanism to defend against malicious attacks. The defense system aims to identify and exclude the three malicious clients (whose labels are flipped for 50% of their data) from the aggregation process. By excluding these clients, the system ensures that only trustworthy data contributes to the global model, thereby mitigating the impact of data poisoning attacks and enhancing the overall convergence rate.

The federated setup involved:

- 10 clients and 1 server using the Flower framework.
- **Non-IID partitioning:** Each client received a unique subset of data to simulate realistic, non-uniform data distributions.
- **Server-Client Communication:** The server initialized training by distributing baseline model parameters (from a saved .pth file) to all clients.
- **Federated Training:** Each client trained locally for 10 epochs, and the server aggregated client parameters using FedAvg before sending updates back to the clients.

We hypothesized that in the second scenario (with attack), flipping labels for 3 clients would delay the model's convergence. This would manifest as increased loss over rounds and a decrease in accuracy as corrupted model weights negatively impact the global model's learning process.

C. Procedure

1) Scenario 2: With Attack:

- For 3 out of the 10 clients, 50% of the labels were flipped to simulate a data poisoning attack.
- These clients trained with corrupted data, sending distorted weights to the server after each round.
- We expected that the loss would increase over rounds and accuracy would degrade due to corrupted model weights from malicious clients.
- The convergence delay would be analyzed by comparing the results with the non-attack scenario.

2) Scenario 2: With Defense:

- A Trust and Reputation mechanism was implemented to identify and exclude 3 malicious clients with 50% of their labels flipped, preventing their distorted weights from impacting the global model.
- Only reliable client contributions were aggregated, mitigating the effect of data poisoning and ensuring model integrity.
- Faster convergence with reduced loss and improved accuracy was observed, compared to the attack scenario without defense.

II. Experimental results

We conducted experiments on a Federated Learning (FL) setup with 10 clients, each participating in 10 rounds of training. During each round, every client performed 5 local epochs before sending their model updates to the server. The results are analyzed for two scenarios: conventional FL without any defense mechanism and FL enhanced with a Trust and Reputation mechanism.

In the first scenario, we introduced an attack by poisoning the data of 3 out of the 10 clients. This was done by flipping

Fig. 1. Evaluation metrics an attack

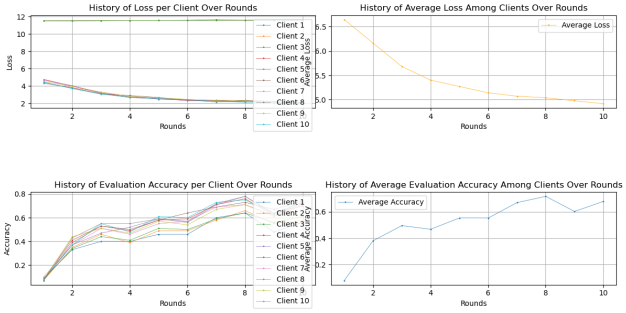

Fig. 2. Evaluation metrics with defense

the labels in their datasets, causing these clients to generate and send malicious weights to the server. These poisoned weights negatively affected the global model, dragging down the performance of the entire FL system. Specifically, the malicious clients introduced instability to the training process, resulting in increased losses and slower convergence for the global model.

Our experimental results indicated a significant discrepancy in loss values among clients. During the first round, the poisoned clients exhibited the highest losses, approximately 11.52, compared to losses of less than 5 for the remaining non-malicious clients. These losses were calculated using cross-entropy loss, which served as a metric to evaluate the performance of individual client models. Due to the aggregation of the malicious weights with the non-malicious weights, the overall loss for the global model remained elevated and converged slowly. Even after 10 rounds of training, the loss decreased only modestly and stabilized around 2.2.

In terms of accuracy, the attack had a pronounced negative effect. While the normal FL setup without any attacks achieved a final accuracy of approximately 80%, the accuracy in the poisoned scenario started at nearly 0% and only improved to around 55-60% by the end of the 10th round. This resulted in a difference of roughly 20% in accuracy between the two setups. This disparity underscores the effectiveness of the attack in disrupting the training process, slowing down the convergence, and reducing the overall efficiency of the FL system.

These results demonstrate the vulnerability of conventional FL setups to data poisoning attacks, as the malicious clients significantly impacted the global model's ability to converge and achieve high accuracy. The findings highlight the importance of implementing robust defense mechanisms, such as Trust and Reputation, to mitigate the effects of such attacks and maintain the integrity and efficiency of the FL system.

In the second scenario, we implemented a defense mechanism based on trust and reputation to mitigate the impact of malicious clients. The mechanism assigns a reputation score to each client, and if a client's reputation falls below a predefined threshold (beta), their weights are excluded from the aggregation process. This approach aims to enhance the resilience of the Federated Learning (FL) system against data poisoning attacks.

Our experiment revealed that the defense mechanism required a few rounds to compute sufficient trust and reputation scores for each client. During the first three rounds, the defense mechanism did not activate, as it was still calculating the trustworthiness of the clients. By the 4th round, the mechanism identified that the three malicious clients had trust scores below the beta threshold. Consequently, their weights were excluded from the aggregation process from the 4th round onwards.

This exclusion of malicious weights had a noticeable impact on the model's performance. After the defense mechanism activated, the global model began to slowly recover and converge. By the end of 10 rounds, the accuracy of the system improved to approximately 70%. This performance was significantly better than the attack scenario without defense, where the accuracy reached only 55-60%. However, it was still lower than the 80% accuracy achieved under normal training conditions without any attacks.

These results demonstrate that the Trust and Reputation mechanism effectively mitigated the negative effects of the attack, allowing the FL system to partially recover and improve its accuracy. While the defense mechanism could not fully match the performance of the normal setup, it provided a significant improvement over the attack scenario, highlighting its potential as a viable strategy for protecting FL systems against malicious clients.

REFERENCES

[1] Differential privacy
https://privacytools.seas.harvard.edu/differential-privacy
[2] Communication-Efficient Learning of Deep Networks from Decentralized Data https://arxiv.org/abs/1602.05629
[3] Federated Learning with Differential Privacy: Algorithms and Performance Analysis
https://arxiv.org/abs/1911.00222
[4] Vision Transformer (ViT)
https://huggingface.co/docs/transformers/en/model_doc/vit
[5] Levy distribution
https://en.wikipedia.org/wiki/L%C3%A9vy_distribution
[6] Comparative assessment of federated and centralized machine learning
https://ar5iv.labs.arxiv.org/html/2202.01529