AFLGuard: Byzantine-robust Asynchronous Federated Learning

Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S. Bentley
Conventional Paradigm: Centralized Learning

Google, Facebook, Apple

Machine learning model

Training data

Clients
Smartphone, IoT devices, self-driving cars
Challenges of Centralized Learning

- Data leakage

- High communication cost
  - Intolerable for resource-constrained clients
Federated Learning

• Training data stay locally on clients
• Clients train models locally
• Clients send model updates to server
• Real-world deployment
Federated Learning Background
Synchronous Federated Learning

- Clients use the same global model to update local models
- Server has to wait until receiving local model updates from all clients
- Training process is slow, due to straggling clients
  - Heterogenous computing capabilities

Clients send local model updates $g$ to the server
Synchronous Federated Learning

• Clients use the **same** global model to update local models
• Server has to **wait** until receiving local model updates from all clients
• Training process is **slow**, due to **straggling clients**
  - Heterogenous computing capabilities

![Diagram](attachment:image.png)

Server sends global model $\theta$ to clients
Asynchronous Federated Learning

- Clients use different global models to update local models
- Server updates the global model immediately upon receiving local model update from any client

Clients send local model updates $g$ to the server
Asynchronous Federated Learning

- Clients use different global models to update local models
- Server updates the global model immediately upon receiving local model update from any client

\[ \theta^1 \]

Server sends global model \( \theta \) to clients
Poisoning Attacks to Federated Learning

Diagram:
- Global model
- Local model
- Malicious client
- Benign client
- Data poisoning attack
- Local model poisoning attack
Challenges

• Only one local model update is received, nothing to compare against

• Difficult to distinguish between malicious local model updates and delayed benign local model updates
Our AFLGuard

• Server collects a small trusted training dataset

• Server maintains a server model
  • Like how a client maintains a local model

• Use server model update to filter out malicious information
Our AFLGuard

A client local model update $g_i$ is considered malicious if

- **Direction** of $g_i$ deviates substantially from that of $g_s$ (server model update) or
- **Magnitude** of $g_i$ deviates substantially from that of $g_s$

$$\|g_i - g_s\| \leq \lambda \|g_s\|$$
Experimental Results

MNIST
100 clients, 20 malicious
Server’s trusted training dataset: 100 examples sampled from MNIST
Maximum client delay and server delay are set to 10

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>AsyncSGD</th>
<th>Kardam</th>
<th>BASGD</th>
<th>Zeno++</th>
<th>AFLGuard</th>
</tr>
</thead>
<tbody>
<tr>
<td>No attack</td>
<td>0.05</td>
<td>0.12</td>
<td>0.19</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td>LF attack</td>
<td>0.09</td>
<td>0.15</td>
<td>0.26</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>Gauss attack</td>
<td>0.91</td>
<td>0.39</td>
<td>0.27</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>GD attack</td>
<td>0.90</td>
<td>0.90</td>
<td>0.89</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>Adapt attack</td>
<td>0.91</td>
<td>0.91</td>
<td>0.90</td>
<td>0.10</td>
<td>0.07</td>
</tr>
</tbody>
</table>

The testing error rates of the global model.
Client Delay

![Client Delay Graph](image_url)

- **Test error rate**
- **Client delay**
- **LF attack**

Legend:
- Red: AsyncSGD w/o attacks
- Purple: AsyncSGD
- Orange: Kardam
- Black: BASGD
- Green: Zeno++
- Blue: AFLGuard
Server Delay

Test error rate

Server delay

LF attack

AsyncSGD w/o attacks

AFLGuard
Conclusion

• We propose a new method called AFLGuard to defend against poisoning attacks in asynchronous federated learning

• We theoretically and empirically show the robustness of AFLGuard
Thank You!