**INTRODUCTION**

How do we train a model without violating privacy or regulatory constraints?

**Federated Learning**
- A group of parties collaboratively trains a machine learning model without sharing/revealing training data.
- Only model updates, such as model weights or gradients, are shared.
- More data better models (Google coined the term in 2016)

**Existing Challenges**
- Data heterogeneity: each party may have a dissimilar data distribution (NON-IID)
- Byzantine threats: parties crash/stall when sending updates, computation errors
- Byzantine attacks: the existence of malicious parties

How to deal with Byzantine threats in federated learning without compromising model performance?

**RELATED WORK**

- Robust statistics:
  - Coordinate median [1] - use coordinate-wise geometric median as the aggregated gradient
  - Geometric median of means, trimmed mean, repeated median
- Pruning updates from malicious parties:
  - Krum [2] - chooses one party's gradients having the smallest $\ell_2$ norms with all other parties' gradients, computation complexity $O(n^2)$
  - Based on Krum: Multi-Krum and Bulyan
- Others:
  - Distributed momentum [3] - uses momentum at the party side to strengthen existing robust aggregation algorithms (i.e. Krum, median)
  - Residual-based reweighting [4] - reweights updates by party based on gradient residuals from a repeated median regression line

**Drawbacks**: Rely on assumptions of bounded honest gradients, which DOES NOT hold in NON-IID case.

**PILOT STUDY**

Existing robust aggregation methods failed in NON-IID case

**OUR SOLUTION - LEGATO**

Algorithm 4: Federated Learning with LEGATO.

1. **Aggregator** Maximum global round $K$, and a learning rate policy $(\eta_t)$.
2. Initialize $w_0$.
3. for round $k = 1, \ldots, K$ do
   - Query party $q_k \in P$ with the current global model weights $w_k$, for its current gradient $g_{q_k}$ and add to its $g^2$.
   - // Aggregated gradient $g^2 = \text{LEGATO}(g^2)$ // (Algorithm 5)
4. return $w_K$.

- **Party**
  - Each party $p \in P$ owns its local dataset, $D_p$, training batch size $B$ and the current model weights $w_k$.
  - Initialize the local model with $w_0 = w_1$.
  - $g_0 = \nabla \ell(w_0, B)$ // $\ell$ denotes the loss function.
  - return $g_0$.

**Threat Model**
- The aggregator is honest and wants to detect malicious or erroneous gradients received during the training process.
- Parties may be dishonest and may collude with each other to evade detection.

**Proposition 1**. LEGATO has time complexity $O(dh + d)$.

**Proposition 2**. LEGATO has space complexity $O(dmn)$.

$n$: number of parties; $d$: layer dimension; $m$: size of the gradient log.

**PRELIMINARY EXPERIMENTS**

**Gaussian attack**, IID, 4 Byzantine parties

**Non-attack**, NON-IID data, WITHOUT Byzantine parties

**LEGATO catches up after 200 rounds**

**Gaussian attack, IIDs, 4 Byzantine parties**

**LEGATO consistently performs better than baselines for NON-IID case**

**Impact of Byzantine attacks**

**Fooling attacks**

- Gradient averaging
- Coordinate median
- Re scaling

**LEGATO consistently performs better than baselines for NON-IID case**

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>LEGATO Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Connected</td>
<td>8.4%</td>
</tr>
<tr>
<td>Convolutional</td>
<td>1.0%</td>
</tr>
</tbody>
</table>

**Resilience of LEGATO**

- Maintains performance in NON-IID data with Byzantine attacks.