Byzantine-Robust Dynamic Weighted Aggregation Framework for Optimal Attack Mitigation in Federated Learning

Supplementary Material

Anonymous ICCV submission

Paper ID 9981

We organize the supplementary material as follows:

1. Section 1 elaborates the black-box data poisoning attack algorithmic details.
2. Section 2 provides more details on hypothesis testing.
3. Section 3 provides the FLOT algorithm and detailed proof of its Byzantine resilience.
4. Section 4 is an extension to experimental settings related to datasets, CNN architecture, and analysis on loss graph.

1. Black-box Attack Method

In this paper, we consider the M-SimBA [5] data poisoning attack as it is more recent and powerful than that of other gradient-based black-box attack methods. In order to generate the adversarial image, a random gradient perturbation is initially added to the original image. It is calculated as

$$I_{adv} = I_x + \epsilon * G_p, \tag{1}$$

where I_{adv} is the adversarial image, I_x is the original image, and G_p is the randomized gradient perturbation. The step size (ϵ) controls the intensity of perturbation. The local black-box model uses I_{adv} to calculate the most confused class score, $[y']$ as

$$y' =_{\hat{Y} \neq Y} \{P(\hat{Y}|I_x)\}, \tag{2}$$

where Y, $\hat{Y}$ are original and predicted class, respectively. The process is repeated until the algorithm generates the final adversarial image as per Eq. 1. For the initial iteration, the gradient is updated in the positive direction. The gradient is added in the negative direction for the next iterations and is subsequently changed randomly.

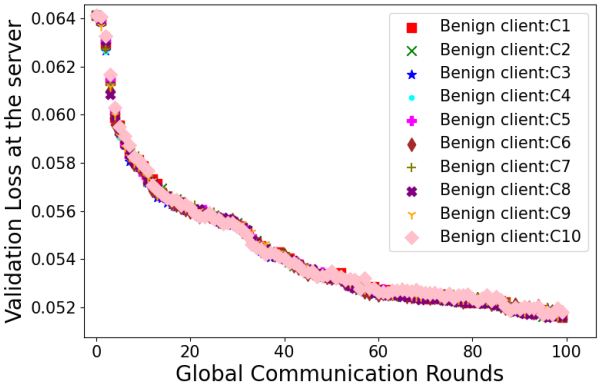
The iterative method creates an adversarial image that will eventually get misclassified. In addition, it converges on the $L2$ norm such that $||I_{adv} - I_x||_2 < \theta$. The threshold parameter (θ) controls the deviation of adversarial image *w.r.t.* original image *without* making it *perceivable to the human eye*. In the final step, converged gradient perturbation (G_p) is added to the input image according to Eq. 1. This step returns the final adversarial image to the local dataset of malicious clients for training. The attack pipeline is shown in Algorithm 1.

Algorithm 1 M-SimBA Data Poisoning Attack [5]

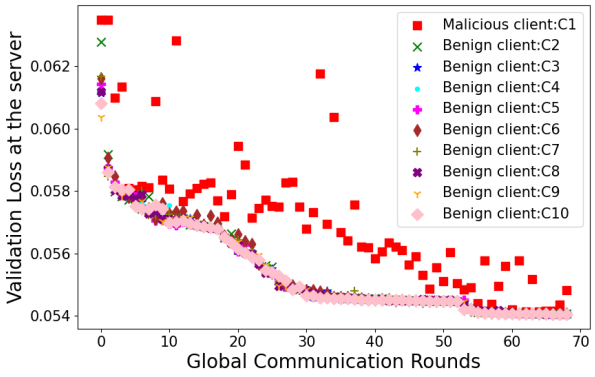
Input: G_t, Global model; $\mathbb{D}_l$: original train data;
Output: $\mathbb{D}_{poison}$: Poisoned train data

for b_p = 1 to batches in $\mathbb{D}_l$ **do** ▷ *Loop through batches in train data*
  for i = 1 to b_s **do** ▷ *Loop through batch size of images*
    $CCS \leftarrow$ most confused class score given by G_t
    $tempCCS \leftarrow 0$ ▷ *Temporary variable to store CCS*
    $ifGradChecked \leftarrow 0$
    Create initial adversarial image (I_{adv}) according to:
    $I_{adv} = I_x + \epsilon * G_p$
    while ($G_t(I_{adv}) ==$ Y) **do**
      if $CCS < tempCCS$ and $ifGradChecked = 0$ **then**
        Update $G_p \leftarrow -(G_p)$
        $ifGradChecked \leftarrow 1$
      else
        Randomize G_p
        $ifGradChecked \leftarrow 0$
      if $||(I_{adv} + \epsilon * G_p) - I_x||_2 < \theta$ **then**
        Update adversarial image (I_{adv}) according to:
        $I_{adv} = I_x + \epsilon * G_p$
      $tempCCS \leftarrow CCS$
      Pass I_{adv} to the Black-box local model C_t for inference
      Update $CCS \leftarrow$ probability of most confused class
    $\mathbb{D}_{poison} \leftarrow I_{adv}$
return $\mathbb{D}_{poison}$

2. More Results on Hypothesis Testing



(a) No Attack



(b) Single-client Attack

Figure 1: Validation losses of individual client model at the server *w.r.t.* global communication rounds under no attack and single-client attack settings for KBTS dataset. Here, the global model is updated with the remaining good-performing client updates. For the next iteration, the clients train their local models using this new global model.

Our defense is based on the hypothesis that the updates from a malicious client doing data poisoning will differ from benign client updates in terms of loss of validation data at the server. Figure. 1 shows the validation loss of 10 clients under no attack and single-client attack settings. We observe under no attack settings, the validation loss of all the updates is clustered together (shows similar behavior) and reduces with the increase in global communication rounds. On the contrary, there are a clear dispersion in the malicious client (C1-squared entries) loss values compared to other benign clients' losses. Hence, our FLOT used loss function-based model rejection to suppress updates from malicious clients.

3. Byzantine Resilience of FLOT

Algorithm 2 Proposed FLOT Method at the Server

Input: w_t^n, n client updates for t^{th} round; $\mathbb{D}_v$: Validation data
Output: G_{t+1}: Updated global model

$\alpha = \{\}$ ▷ *LFR based weight multiplier vector*
for $i = 1$ to n **do** ▷ *Loop through n models*
  $\alpha \leftarrow \mathcal{L}(\mathbb{D}_v, w_t^i)$ ▷ *Validation loss*
$\alpha \leftarrow |\alpha - \max(\alpha)|$
$\alpha \leftarrow \text{normalize}(\alpha)$ ▷ *s.t.* $\alpha_i \in [0,1], \forall i \in n$
$\mathcal{M} \leftarrow$ FLOT cost matrix
$G_{t+1} \leftarrow ot.lp.barycenter(w, \mathcal{M}, \alpha)$ ▷ *FLOT aggregator*
return G_{t+1}

Algorithm 2 presents detailed steps of our proposed FLOT method at the server. Further, we prove the Byzantine resilience of FLOT starting with Corollary 3.0.1, followed by Proposition 3.1 and its proof.

Corollary 3.0.1 *An aggregation rule A of the form $A(w_1, w_2, \ldots, w_n) = \sum_{i=1}^{n} \lambda_i w_i$ (federated averaging [7]), where $\lambda_i > 0$ and $\sum_{i=1}^{n} \lambda_i = 1$, is not byzantine robust as a single malicious client k can prevent convergence by proposing $w_k = \frac{1}{\lambda_k} w'_k - \sum_{i=1}^{n-1} \frac{\lambda_i}{\lambda_n} w_i$, then $A(w_1, w_2, \ldots, w_n) = w'_k$, where w'_k is the malicious update from the single byzantine client [2].*

The below Proposition 3.1 signifies that if there are ρ malicious clients, χ client updates that are trained on highly non-*i.i.d.* data, and the combined validation loss excluding these $\rho + \chi$ model updates is less than ω, then our FLOT function is $(\omega, \rho\chi)$ - Byzantine Resilient, where $\omega \geq 0$.

Proposition 3.1 *Let $[\mathcal{N}] = \{w_1, \ldots, w_n\}$ be any non-i.i.d. local clients models in $\mathbb{R}^d$. Let $[\mathcal{R}] = \{w'_1, \ldots, w'_\rho\}$ be any non-i.i.d. Byzantine local clients models. Let $[\mathcal{X}] = \{w''_1, \ldots, w''_\chi\}$ be any highly non-i.i.d. benign local clients models. FLOT is said to be $(\omega, \rho\chi)$-Byzantine Resilient) where $\omega \geq 0$, if for any $(w_1, \ldots, w'_1, \ldots, w'_\rho, \ldots, w''_1, \ldots, w''_\chi, \ldots, w_n)$ model updates, satisfies the following*

$$\sum_{w_k \in ([\mathcal{N}] - [\mathcal{R}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \quad (3)$$

$$\sum_{w_k \in [\mathcal{N}] - [\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \quad (4)$$

$$\left\| \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{N}']} \mathcal{L}(\mathbb{D}_v, w_k) \right\| \geq \omega, \quad (5)$$

for some $\omega \geq 0$. Here, $\mathcal{N} = [\mathcal{N}'] - ([\mathcal{R}] + [\mathcal{X}])$, $\mathcal{L}(\mathbb{D}_v, w_k)$ denote the loss of w_i model on validation data $\mathbb{D}_v$.

Proof. Without loss of generality, we assume (a) the Byzantine client updates are indexed after benign client vectors, (b) the highly non-*i.i.d.* updates are indexed after the Byzantine updates, i.e.,

$$FLOT(w_1, \ldots, w'_1, \ldots, w'_\rho, \ldots, w''_1, \ldots, w''_\chi, \ldots, w_n). \tag{6}$$

First, we focus on proving the condition *(i)* of Proposition 3.1. Consider first case where $w_k|k \in ([\mathcal{N}] - [\mathcal{R}])$, (benign model updates without any malicious updates). Based on the **Theorem 2.** of Jagielski *et al.* [3] given by

$$\mathcal{L}(\mathbb{D}', \hat{w}) \leq \mathcal{L}(\mathbb{D}_{tr}, w^*), \tag{7}$$

where $\mathbb{D}'$ represents the malicious training data samples, $\mathbb{D}_{tr}$ is total training data including malicious samples. $\mathcal{L}(.,.)$ is the training loss on poisoned $\hat{w}$ and main w^* models, respectively. However, Jagielski *et al.* [3] proved it in terms of data poisoning attacks in centralized machine learning settings with a number of malicious samples under attack. We extend it to federated learning settings in terms of multiple malicious client models that are trained on poisoned and different amounts of non-*i.i.d.* data. Using the set of malicious updates $[\mathcal{R}]$, set of benign updates $([\mathcal{N}] - [\mathcal{R}]) = \{w_1, w_2, \ldots, w_{n-\rho}\}$, validation data at the server $\mathbb{D}_v$, and Eq. 7, we provide the below formulation to prove condition *(i)* of Proposition 3.1 as

$$\begin{aligned} \mathcal{L}(\mathbb{D}_v, w_1) &< \mathcal{L}(\mathbb{D}_v, w'_1), \\ \mathcal{L}(\mathbb{D}_v, w_2) &< \mathcal{L}(\mathbb{D}_v, w'_1), \\ &\ldots \\ \mathcal{L}(\mathbb{D}_v, w_\rho) &< \mathcal{L}(\mathbb{D}_v, w'_\rho), \end{aligned} \tag{8}$$

summing up elements on both hand sides and further adding remaining $n - \rho$ elements on both sides, and rearranging terms, we get

$$\sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w'_k), \tag{9}$$

$$\begin{aligned} &\sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{k=\rho+1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \\ &\sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w'_k) + \sum_{k=\rho+1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k), \end{aligned} \tag{10}$$

$$\sum_{k=1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w'_k) + \sum_{k=\rho+1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k). \tag{11}$$

Adding an additional $\sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w_k)$ term to the right hand side of Eq. 11 still holds the equation.

$$\sum_{k=1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w'_k) + \sum_{k=\rho+1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w_k),$$

$$\sum_{k=1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w'_k) + \sum_{k=1}^{\rho} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{k=\rho+1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k), \tag{12}$$

$$\sum_{k=1}^{n-\rho} \mathcal{L}(\mathbb{D}_v, w_k) < \sum_{k=1}^{n} \mathcal{L}(\mathbb{D}_v, w_k), \tag{13}$$

$$\sum_{w_k \in [\mathcal{N}]-[\mathcal{R}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k). \tag{14}$$

Here $=$ holds true when $\rho = 0$. Finally, Eq. 14 proves the condition *(i)* of Proposition 3.1.

Next, we prove the condition *(ii)* of Proposition 3.1 based on Balakrishnan *et al.* [1]. In this work, the authors propose an optimization method to select a subset of client updates that carry representative gradient information of the entire client set. Further, they transmit only the selected subset of client updates to the server for aggregation. The aim is to find an approximation of full clients (n) aggregation gradient via a subset $\mathcal{S}$ of client updates. The authors formulate the problem to provide the upper bound for the aggregated gradient approximation derived from the subset $\mathcal{S}$ of clients as

$$\begin{aligned} &\left\| \sum_{k \in n} \nabla F_k(v^k) - \sum_{k \in \mathcal{S}} \gamma_k \nabla F_i(v^i) \right\| \leq \\ &\sum_{k \in n} \min_{i \in \mathcal{S}} \left\| \nabla F_k(v^k) - \nabla_i F_i(v^i) \right\|, \end{aligned} \tag{15}$$

where given a subset $\mathcal{S}$, they define a mapping $\sigma : \mathcal{V} \to \mathcal{S}$, such that the gradient information $\nabla F_k(v^k)$ from a client k is approximated by the gradient information from a selected client $\sigma(k) \in \mathcal{S}$. Further, they provide the gradient approximation error as

$$\left\| \frac{1}{n} \sum_{k \in \mathcal{S}^t} \gamma_k \nabla F_k(v_t^k) - \frac{1}{n} \sum_{k \in n} \nabla F_k(v_t^k) \right\| \leq \delta, \\ \left\| \sum_{k \in \mathcal{S}^t} \gamma_k \nabla F_k(v_t^k) - \sum_{k \in n} \nabla F_k(v_t^k) \right\| \leq n\delta, \tag{16}$$

where t is the communication round, $\{\gamma\}_{k \in \mathcal{S}_t}$ are the weights assigned to gradients, and δ is the error rate that is used as a measure to characterize the goodness of gradient approximation. The above equation states that the gradient approximation from subset $\mathcal{S}$ of clients at communication round t is less than $n\delta$ times full gradient aggregation from all clients. Further, we use this observation and extend it to validation loss that there exists a subset of client updates ([N]-[X]) whose sum of validation losses is less than that of the sum of total clients. It is given as

$$\left\| \sum_{k \in \mathcal{S}^t} \mathcal{L}(\mathbb{D}_v, v_t^k) - \sum_{k \in n} \mathcal{L}(\mathbb{D}_v, v_t^k) \right\| \leq n\delta, \\ \left\| \sum_{w_k \in [\mathcal{N}]-[\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) \right\| \leq n\delta. \tag{17}$$

Here, $\mathcal{N}] - [\mathcal{X}]$ denote the subset of clients obtained after discarding $[\mathcal{X}]$ non-*i.i.d.* clients whose validation loss is higher than that of remaining clients. Finally, the below equation proves the condition *(ii)* of Proposition 3.1.

$$\sum_{w_k \in [\mathcal{N}]-[\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) \leq n\delta \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \\ \sum_{w_k \in [\mathcal{N}]-[\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k). \tag{18}$$

Combining Eq. 14 and Eq. 18 we get

$$\sum_{w_k \in [\mathcal{N}]-[\mathcal{R}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{19}$$

$$\sum_{w_k \in [\mathcal{N}]-[\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{20}$$

$$\sum_{w_k \in [\mathcal{N}]-[\mathcal{R}])} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{w_k \in [\mathcal{N}]-[\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) \leq \\ \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{21}$$

$$\sum_{w_k \in [\mathcal{R}]} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{w_k \in [\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) + \\ 2 \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq 2 \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{22}$$

$$2 \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq 2 \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) - \\ \sum_{w_k \in [\mathcal{R}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{23}$$

$$2 \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) + \\ \sum_{w_k \in [\mathcal{R}]} \mathcal{L}(\mathbb{D}_v, w_k) + \sum_{w_k \in [\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k) + \\ \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) - \\ \sum_{w_k \in [\mathcal{R}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{X}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{24}$$

$$2 \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) + \\ \sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k), \tag{25}$$

$$\sum_{w_k \in ([\mathcal{N}]-([\mathcal{R}]+\mathcal{X}])} \mathcal{L}(\mathbb{D}_v, w_k) \leq \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k), \tag{26}$$

$$\left\| \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{N}']} \mathcal{L}(\mathbb{D}_v, w_k) \right\| \geq 0, \tag{27}$$

generalizing,

$$\left\| \sum_{w_k \in [\mathcal{N}]} \mathcal{L}(\mathbb{D}_v, w_k) - \sum_{w_k \in [\mathcal{N}']} \mathcal{L}(\mathbb{D}_v, w_k) \right\| \geq \omega, \tag{28}$$

where $\omega \geq 0$, $[\mathcal{N}'] = [\mathcal{N}] - ([\mathcal{R}] + \mathcal{X}])$. Finally, Eq. 28 proves the condition *(iii)* of Proposition 3.1.

4. Experiments

4.1. Datasets

GTSRB [8] is a well-known benchmark dataset for traffic sign classification. It consists of 43 traffic sign classes. Most (70%) of the training data (27,446 samples) is divided using the Dirichlet distribution with $\alpha = 1$. Further, 10% (3920 samples) is used as validation data, and the remaining 20% of the data (7842 samples) is used for testing. **KUL Belgium traffic sign (KBTS)** dataset [6] is another benchmark dataset for traffic sign classification. It consists of 62 traffic sign classes. A majority (70%) of the training data (4884 samples) is divided using the Dirichlet distribution with $\alpha = 1$. Further, 10% (697 samples) is used as validation data, and the remaining 20% of the data (1397 samples) is used for testing. **CIFAR10** [4] is a well-known benchmark dataset for classification that contains 60,000 samples with ten different classes. Most (70%) of the training data (42,000 samples) is divided using the Dirichlet distribution with $\alpha = 1$. Further, 10% (6000 samples) is used as validation data, and the remaining 20% of the data (12000 samples) is used for testing.

4.2. Base Classifier Architecture

We build a custom 4-layer CNN architecture followed by two fully connected layers and treat this as a global model, as shown in Table 1. The model is trained with images of size 150×150 using categorical cross-entropy as loss function optimized using Adam optimizer. During the training of the global classifier for 200 epochs through FL protocol, each client trains for $E = 5$ local epochs on the local data with a batch size $b_s = 64$ and with a learning rate of $l_r = 0.01$. All the clients are trained individually and sequentially at each global epoch. We used Python 3.6+, Pytorch, and python OT (especially *ot.lp.barycenter* function with solver='interior-point') and implemented the entire setup on Nvidia Tesla M60 GPU & 8GB RAM.

Table 1: CNN configuration

Black-box CNN (4 Conv layers)
input (150×150 RGB images)
conv2d_64; kernel 5; stride 1
conv2d_128; kernel 3; stride 1
conv2d_256; kernel 1; stride 1
conv2d_256; kernel 1; stride 1
Fully connected layer 1
Fully connected layer 2
Softmax classifier

Test loss analysis: Figure 2 shows the performance of FLOT compared to other Byzantine server rules. For brevity, we showed the hard case of a multiclient attack (33% Byzantine) for the KBTS dataset with ten clients. We observed that our FLOT showed a lower loss, followed by Krum. Further, we observe that FLOT configuration is better in this case as RS+FLOT randomly selects some clients and applies FLOT on top of it. As there are less number of clients for the KBTS dataset, sampling clients randomly and using FLOT leads to losing the benign client updates and lower performance. Trimmed mean, with its ability to trim client updates from beginning to end, leads to discarding benign updates and including malicious updates. Hence, it performs worst compared to other methods.

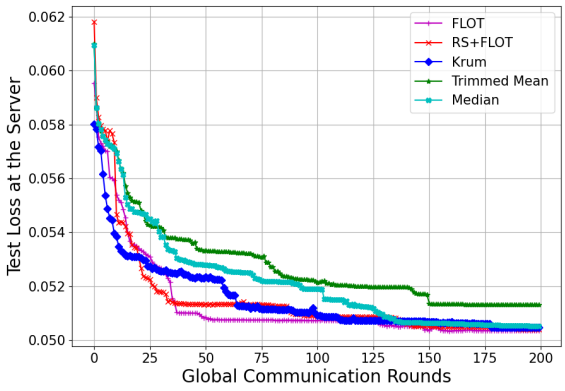


Figure 2: Comparison of test losses of FLOT with different Byzantine aggregation techniques at the server for 200 global communication rounds under 33% multi-attack settings for KBTS dataset.

References

[1] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2021. 3

[2] 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. 2

[3] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018. 3

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[5] K Naveen Kumar, C Vishnu, Reshmi Mitra, and C Krishna Mohan. Black-box adversarial attacks in autonomous vehicle technology. In *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7. IEEE, 2020. 1

[6] Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool. Traffic sign recognition — how far are we from the solution? In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013. 5

[7] 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*, pages 1273–1282. PMLR, 2017. 2

[8] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011. 5