Federated Trustworthy Updates for Secure Training: Byzantine Robust Federated Learning via Client Trustworthiness Scores

Rami Pellumbi*
Department of Statistics & Data Science
Yale University
rami.pellumbi@yale.edu

Abstract

Federated learning is a machine learning technique that enables training a model across multiple decentralized edge devices or servers without exchanging local data samples. This approach is vulnerable to byzantine attacks, where malicious participants (byzantine clients) upload arbitrary or carefully modified weights to the central server, degrading its performance. Byzantine robust federated learning aims to mitigate the influence of byzantine clients during the training process. In this study, we propose FEDerated TRustworthy Updates for Secure Training (FED-TRUST): a method to establish a client trustworthiness score based on the difference between their weights and the median weights of all clients after each training round. The impact of a client's weights on the global model is directly proportional to their trustworthiness score. We demonstrate that, in the case of independent and identically distributed (i.i.d) data amongst clients and an appropriate number of adversarial clients, our method effectively identifies adversaries and significantly improves the results of federated learning in the presence of adversaries. In the non i.i.d case, the training approach works under suitable conditions.

1 Introduction

Federated learning has emerged as a promising approach to train machine learning models across multiple decentralized edge devices or servers without the need for exchanging local data samples. This technique offers several advantages, such as preserving data privacy and reducing communication overhead.[5] However, one significant challenge it faces is its vulnerability to byzantine attacks, where malicious participants, known as byzantine clients, can upload arbitrary or carefully modified weights to the central server. These adversarial actions can degrade the performance of the aggregated global model, posing a severe threat to the system's reliability and effectiveness. [1] [3]

The vulnerability of federated learning systems to such attacks necessitates robust mechanisms that can ensure the integrity and trustworthiness of the training process. These systems are crucial in environments where data privacy is paramount and where data cannot be centralized, such as in healthcare, finance, and mobile services. Therefore, there is a critical need for strategies that can secure federated learning against these types of threats to maintain the quality and fairness of the machine learning models produced.

To address this, we propose FED-TRUST: a method that enhances the robustness of federated learning systems by establishing a client trustworthiness score for each client. This score is based on the deviation of a client's weights from the median weights of all clients after each training round and the impact of a client's weights on the global model is directly proportional to their trustworthiness score. By assigning higher importance to the contributions of trustworthy clients and reducing the

*Source code: https://github.com/ramipellumbi/Federated-Learning-with-an-Adversary

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

influence of potentially malicious clients, our approach seeks to safeguard the federated learning process against byzantine attacks, thereby enhancing both the security and the performance of these distributed systems.

2 Related Work

Byzantine robust federated learning has attracted significant attention in recent years due to the increasing importance of data privacy and security in distributed machine learning. Several attack methods and robustness approaches have been proposed to mitigate the impact of byzantine attacks on federated learning systems. In trust-based schemes, it is assumed that some of the clients or datasets are adversarial in the distributed setting. Authors in [2] optimize a regularized loss function that accounts for the quality of a source by defining a discrepancy function between two distributions with respect to a hypothesis class. This discrepancy is large if a predictor that performs well on one distribution and badly on the other and allows them to modify their loss function to incorporate and penalize discrepancy. The work in [7] introduces `SignGuard`, which enhances the robustness of federated learning against byzantine clients using statistic derived from the sign of the gradient vector. Authors in [4] propose privacy-preserving byzantine robust federating learning, which integrates distributed Paillier encryption and zero-knowledge proofs with existing byzantine robust approaches to filter out anomalous parameters from malicious nodes. More recent work in [8] applies recent progress in high dimensional robust statistics to propose a new byzantine robust protocol which achieves a near optimal statistical rate for strongly convex losses. Similar to our proposed approach, [6] applies median methods in the iterative federated clustering framework to estimate cluster membership and optimize their models. Their work does not establish a trust score for a client but uses statistics from the median to draw conclusions on whether a client is adversarial.

3 Architecture

3.1 Intuition

The FED-TRUST algorithm is crafted to counter byzantine attacks in federated learning systems, enhancing the security and robustness of these systems. Central to this algorithm is the idea that benign clients generally submit similar model weights, whereas malicious clients often introduce significantly divergent weights due to adversarial objectives. To address this, FED-TRUST evaluates the trustworthiness of each client by measuring the deviation of their model weights from the median across all clients. The algorithm quantifies this deviation using the L1 distance between a client's weights and the median weights for each layer of the model. It then normalizes these scores to a range between 0 and 1, where higher deviations result in scores closer to 0 and lower deviations closer to 1. The normalization is achieved by subtracting the normalized deviation from 1. To accommodate changes over time, the trustworthiness scores are updated every communication round through a weighted average of the previous and the new scores, ensuring a gradual adaptation of trust. During the model aggregation phase, the server adjusts the influence of each client's contribution based on their trustworthiness scores and the quantity of data they contribute. Clients that are deemed more trustworthy and those that provide more data have a more substantial impact on the updated global model. This strategy diminishes the potential damage from malicious clients by reducing their influence on the model's evolution, thereby promoting convergence towards the true underlying data distribution.

3.2 Implementation

The setup of the FED-TRUST algorithm starts as the typical federated learning setup, i.e., involves a server and multiple clients participating in the federated learning process.[5] There are N clients and each client has its own local dataset $\mathcal{D}_n$ of size $|\mathcal{D}_n| = S_n$, which is used to train the model locally. The server maintains the global model and coordinates the communication between clients. The algorithm operates in communication rounds, where in each round the server sends the current global model weights to the clients, who then perform local training on their respective datasets. After training, the clients send their updated model weights back to the server for aggregation. Instead of weighting the clients weights by the fraction of data the client processed, the server first computes a 'trustworthy' score for each client based on the deviation of their weights from the median weights

across all clients. The weight updates from the clients are performed as (Algorithm 1):

$$w_{r+1} \leftarrow \sum_n T^n_{r+1} w^n_{r+1}, \tag{1}$$

where w^n_{r+1} are the weight updates of client n in this round and T^n_{r+1} is the total weight given to client n, which is a function of the volume of data they processed and their trustworthy score, computed as:

$$\widetilde{T}^n_{r+1} = \tau^n_{r+1} \frac{S_n}{S}, \quad n = 1, ..., N, \tag{2}$$

where τ^n_{r+1} is as calculated in Algorithms 2 and 3, and $S = \sum_n S_n$ is the number of total samples processed. Of course, we normalize the above to ensure the weight update is a convex combination of the client weights:

$$T^n_{r+1} = \frac{\widetilde{T}^n_{r+1}}{\sum_n \widetilde{T}^n_{r+1}} \tag{3}$$

In the case when thresholding is used, only clients with score greater than the threshold, T, are included in the aggregation, i.e., the update becomes:

$$w_{r+1} \leftarrow \sum_{n \in \{n: T^n_{r+1} > T\}} T^n_{r+1} w^n_{r+1}, \tag{4}$$

where the set $\{T^n_{r+1} : T^n_{r+1} > T\}$ is appropriately normalized after thresholding.

The primary Algorithm is listed in Algorithm 1. We relegate Algorithms 2 and 3, which clarify the details of updating a client's trustworthy score, to the Appendix.

Algorithm 1 TRUST Federated Learning. The N clients are indexed by n; B is the minibatch size, and η is the learning rate. Each client is given a proper subset $\mathcal{D}_n$ of $\mathcal{D}$. The variable τ^n_r is the client's 'trustworthiness' score in round r. The variable T^n_r is the weight that client is given as a function of the number of samples they have as well as their trustworthy score. The variable T represents the threshold a client's score needs to be over to be included in the aggregation.

procedure SERVER EXECUTION
  initialize w_0
  initialize $\tau^n_0 = \frac{1}{N}$ for all n
  for each communication round $r = 1, 2, ...$ **do**
    for each client $n = 1, 2, ..., N$ **do**
      $w^n_{r+1} \leftarrow \mathsf{ClientUpdate}(n, w_r)$
    end for
    `ComputeTrustScores`$(\{w^n_{r+1}\}_{n=1}^N)$
    Compute scores T^n_{r+1} for all n as in Equations (2) & (3)
    if $T > 0$ **then**
      Threshold the scores T^n_{r+1} and normalize the remaining scores before weighting
    end if
    $w_{r+1} \leftarrow \sum_{n \in \{n: T^n_{r+1} > T\}} T^n_{r+1} w^n_{r+1}$
    Set τ^n_{r+1} to T^n_{r+1} for all n

  end for
end procedure

procedure $\mathbf{ClientUpdate}(n, w)$ // run on client n
  $\mathcal{B} \leftarrow (\mathcal{D}_n$ split into batch of size $)B$
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
  Send w to server
end procedure

4 Empirical Results

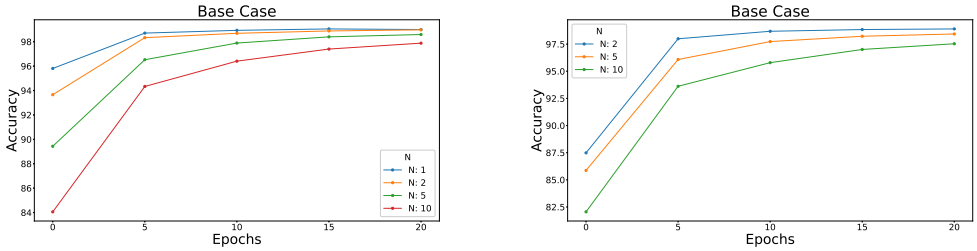
An adversarial client is deemed one that injects noise sampled from a Gaussian with mean $\mathcal{N}(0, \sigma^2 I)$ into its parameters, where $\sigma > 0$ is a hyperparameter. The number of adversaries in the federated training setup with N clients is denoted $N_{\text{adv}} < N$.

To assess the effectiveness of the FED-TRUST algorithm, comparisons were made against a baseline federated learning setup without any robustness mechanisms against byzantine attacks. This baseline represents a standard federated averaging approach (FedAvg), where the server computes the global model by averaging the updates received from all clients without any adjustment for trust. Experiments are run where the server employs both the standard FedAvg technique vs. FED-TRUST for varying values of $N, N_{\text{adv}}, \sigma$. Each client runs 20 rounds of training, and each local client update processes all mini batches (of size 64) of the clients data per round. We run all our assessments on the canonical MNIST dataset, and explore clients in the system having an i.i.d distribution vs. non-i.i.d distribution of the dataset. We use the accuracy of the server model on the test images after all rounds of training are complete as the comparison method across configurations.

We show our results for both the I.I.D case at the clients as well as the non I.I.D case. The accuracy in all figures represents the accuracy on the test set at that checkpoint.

4.1 I.I.D Results

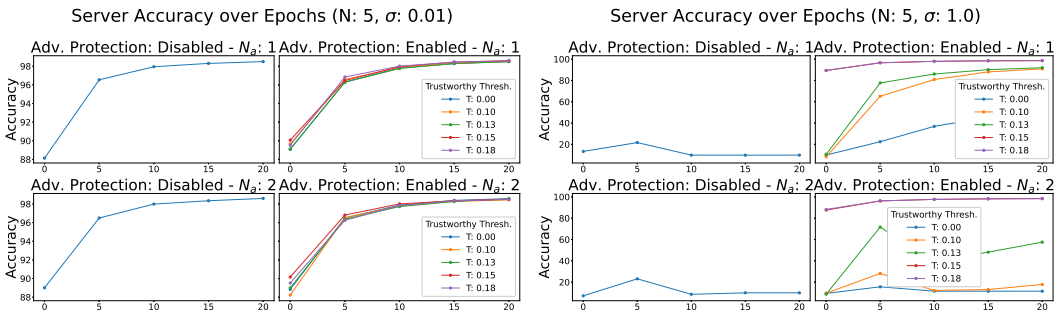
We begin with our baseline results in Figure 1a and then juxtapose that to adversarial clients in both the FedAvg and FED-TRUST regimes. We see that as N increases the noise introduced from the aggregation introduces performance drops that are rectified through the rounds of training. In Figures 2 and 3, we observe that with small weight perturbation, using both the standard aggregation and FED-TRUST induce similar performance as the noise is small enough to be averaged out. In the case where $\sigma = 1$, there is enough variance on the weights that even one byzantine client completely corrupts the global models performance.



(a) Performance under FedAVG with No Adversaries Under I.I.D Training Data.

(b) Performance under FedAVG with No Adversaries Under Non I.I.D Training Data.

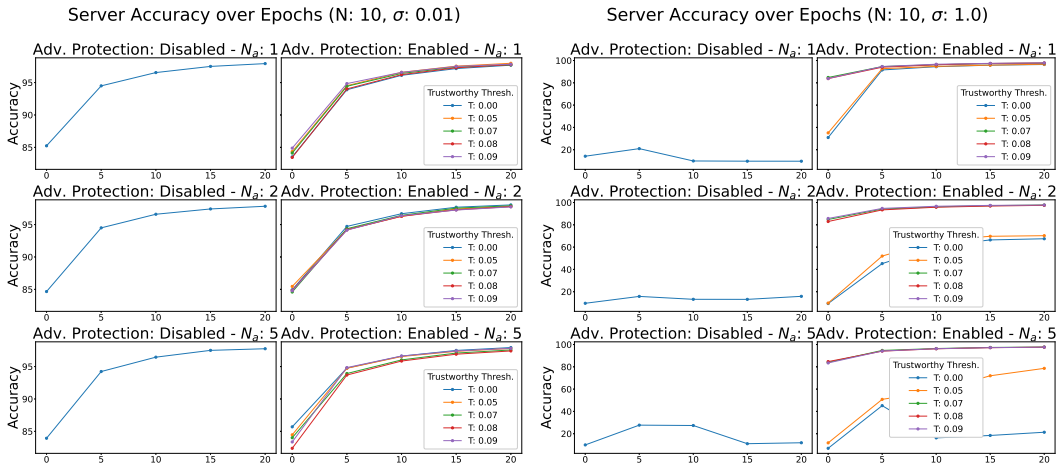
Figure 1: Base Case Performance.



(a) Accuracy of the model with low noise variance

(b) Accuracy of the model with high noise variance

Figure 2: Comparison of server model accuracy under different noise variances for $N = 5$ clients.



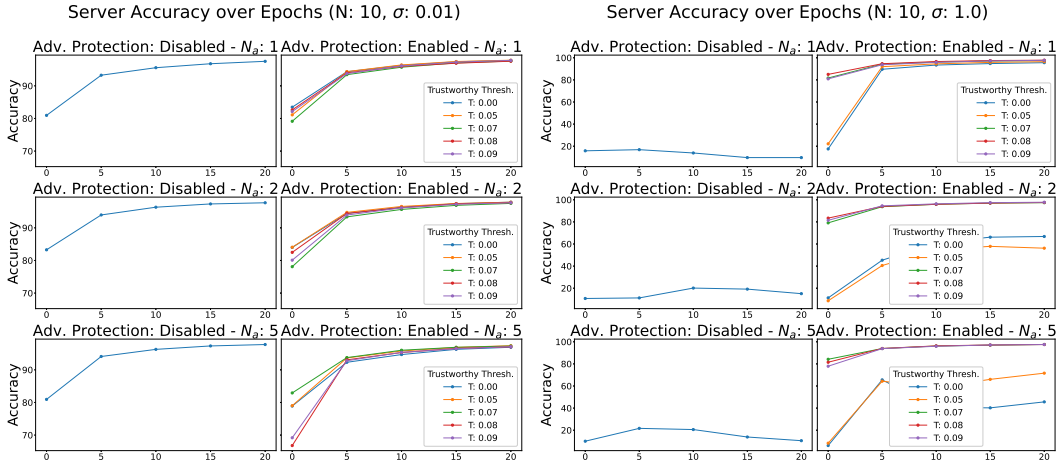
(a) Accuracy of the model with low noise variance (b) Accuracy of the model with high noise variance

Figure 3: Comparison of server model accuracy under different noise variances for $N = 10$ clients.

The introduction of FED-TRUST with no threshold is able to significantly improve the performance over baseline up until $N_{\text{adv}} \approx {}^{N}/_{2}$. This result makes sense, since once roughly half the clients are adversaries, median outlier detection will fail to detect them. Introducing threshold scores of ${}^{1}/_{1.1N}$, ${}^{1}/_{1.3N}$, ${}^{1}/_{1.5N}$, and ${}^{1}/_{2N}$, we see that thresholds near the uniform distribution, i.e., ${}^{1}/_{1.1N}$, ${}^{1}/_{1.3N}$, recover performance near completely – as though the adversaries do not exist.

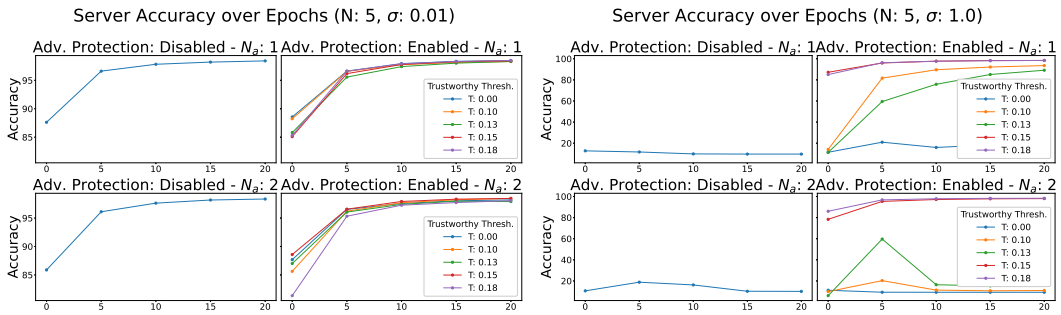
4.2 Non I.I.D Results

Similar to above, we begin with our baseline results in Figure 1b and then juxtapose that to adversarial clients in both the FedAvg and FED-TRUST regimes. In the non i.i.d case we consider $N > 1$ and ensure clients have dissimilar non uniform distributions on the data. Similar to the i.i.d case, we see that as N increases the noise introduced from the aggregation introduces performance drops that are rectified through the rounds of training. In Figures 5 and 4, we observe that with small weight perturbation, using both the standard aggregation and FED-TRUST induce similar performance as the noise is small enough to be averaged out. In the case where $\sigma = 1$, there is enough variance on the weights that even one byzantine client completely corrupts the global models performance. The introduction of FED-TRUST with no threshold is able to significantly improve the performance over baseline up until $N_{\text{adv}} \approx {}^{N}/_{2}$.



(a) Accuracy of the model with low noise variance (b) Accuracy of the model with high noise variance

Figure 4: Comparison of server model accuracy under different noise variances for $N = 10$ clients with non i.i.d training data.



(a) Accuracy of the model with low noise variance

(b) Accuracy of the model with high noise variance

Figure 5: Comparison of server model accuracy under different noise variances for $N = 5$ clients with non i.i.d training data.

We draw the same conclusions as in the i.i.d case. While we are able to see the FED-TRUST regime resolve adversaries in this setting, we remark that MNIST itself is a simple problem and should not be indicative of the potential for this to perform on non i.i.d client distributions.

5 Conclusion

This paper introduced the FED-TRUST algorithm, a new approach to enhance the robustness of federated learning systems against Byzantine attacks. By assigning trustworthiness scores to clients based on the deviation of their model weights from the median, and integrating these scores into the model aggregation process, FED-TRUST effectively mitigates the influence of malicious actors. Our experiments on the MNIST dataset demonstrate that FED-TRUST substantially improves the performance and security of federated learning in scenarios with independent and identically distributed (i.i.d.) data, especially when the proportion of adversarial clients is kept below the critical threshold of $N/2$. Moreover, we find that the optimal threshold is slightly below uniform, with $T = 1/N*1.1$ consistently filtering out adversaries under the attack specified.

While FED-TRUST shows promising results in enhancing the robustness of federated learning, it has limitations. The effectiveness of the algorithm diminishes when the number of adversarial client approaches half of the total clients, i.e., $N_{\text{adv}} \to N/2$. In such scenarios, the median weight used to calculate trustworthiness scores may itself be influenced by adversaries, potentially leading to incorrect trust assessments and compromised model integrity.

Future research could focus on extending the FED-TRUST framework to better handle scenarios where adversarial clients form a majority. This might involve developing more sophisticated metrics for trustworthiness or mechanisms that can use other aspects of client data or behavior to assess trust. Additionally, exploring the application of FED-TRUST in non-i.i.d. environments for more complex datasets and its integration with other privacy-preserving techniques like differential privacy could provide deeper insights and broader applicability in the practical deployment of federated learning systems. Lastly, implementing more sophisticated attack mechanisms and assessing FED-TRUST against them to measure robustness against the current gold standards in attack mechanisms would allow for a more concrete conclusion on the algorithms effectiveness.

References

[1] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[2] Nikola Konstantinov and Christoph Lampert. Robust learning from untrusted sources. *CoRR*, abs/1901.10310, 2019.

[3] Shenghui Li, Edith C.-H. Ngai, and Thiemo Voigt. An experimental study of byzantine-robust aggregation schemes in federated learning. *IEEE Transactions on Big Data*, page 1–13, 2024.

[4] Xu Ma, Yuqing Zhou, Laihua Wang, and Meixia Miao. Privacy-preserving byzantine-robust federated learning. *Computer Standards & Interfaces*, 80:103561, 2022.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[6] Zhixu Tao, Kun Yang, and Sanjeev R. Kulkarni. Byzantine-robust clustered federated learning, 2023.

[7] Jian Xu, Shao-Lun Huang, Linqi Song, and Tian Lan. Byzantine-robust federated learning through collaborative malicious gradient filtering. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1223–1235, 2022.

[8] 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, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 3151–3178. PMLR, 25–27 Apr 2023.

Appendix

The below algorithms are provided for completeness in the description of FED-TRUST.

Algorithm 2 Compute Trust Scores of all clients

procedure COMPUTETRUSTSCORES($\{w^n\}_{n=1}^N$))
  // model weights where every layer has the median values
  $M \leftarrow$ CalculateMedianWeights($\{w^n\}_{n=1}^N$)))
  $\kappa \leftarrow [\,]$
  for each w^n, $n = 1, ..., N$ **do**
    $\Delta \leftarrow 0$
    for each layer weights l_{w^n} **do**
      $M_l \leftarrow$ `median weights of this layer`
      $\delta \leftarrow \|l_{w^n} - M_l\|_1$
      $\Delta \leftarrow \Delta + \delta$
    end for
    Append Δ to κ
  end for
  `MAX` $\leftarrow \max_n \kappa^n$
  $\tilde{\tau} \leftarrow [\,]$
  for each $\Delta \in \kappa$ **do**
    `score` $\leftarrow 1 - \Delta/$`MAX`
    Append `score` to $\tilde{\tau}$
  end for
  `UpdateTrustScores`($\{\tilde{\tau}^n\}_{n=1}^N$)
end procedure

Algorithm 3 Update Trust Scores of all clients in a communication round

procedure UPDATETRUSTSCORES($\{\tilde{\tau}^n\}_{n=1}^N$)
  for each client $n = 1, ..., N$ **do**
    // τ_r^n is the trust score last round and $\tilde{\tau}^n$ is the new median deviation this round
    $\tau_{r+1}^n \leftarrow 0.9 \times \tau_r^n + 0.1 \times \tilde{\tau}^n$
  end for
  $\tau \leftarrow \sum_{n=1}^N \tau_{r+1}^n$
  for each client $n = 1, ..., N$ **do**
    $\tau_{r+1}^n \leftarrow \tau_{r+1}^n / \tau$
  end for
end procedure