FedAegis: Edge-Based Byzantine-Robust Federated Learning for Heterogeneous Data

Fangtong Zhou, Zhouyu Li, Xiaojian Wang, Huayue Gu, Ruozhou Yu

***Abstract*—This paper studies how an edge-based federated learning algorithm called FedAegis can be designed to be robust under both heterogeneous data distributions and Byzantine adversaries. The divergence of local data distributions leads to suboptimal results for the training process of federated learning, and the Byzantine adversaries aim to prevent the training process from converging in a distributed learning system. In this paper, we show that an edge-based hierarchical federated learning architecture can help tackle this dilemma by utilizing edge nodes geographically close to clusters of local devices. By combining a distributionally robust global loss function with a local Byzantine-robust aggregation rule, FedAegis can defend against remote Byzantine adversaries who cannot manipulate local devices' connections to edge nodes, meanwhile accounting for global data heterogeneity across benign local devices. Experiments with the MNIST, FMNIST and CIFAR-10 datasets show that our proposed algorithm can achieve convergence and high accuracy under heterogeneous data and various attack scenarios, while state-of-the-art defenses and robustness mechanisms are non-converging or have reduced average and/or worst-case accuracy.**

***Keywords*—Federated learning, edge computing, data heterogeneity, Byzantine robustness, distributionally robust optimization**

I. Introduction

The increasing usage of mobile devices has resulted in unprecedented massive amount of data being generated, and provides machine learning applications with the perfect opportunity to train jointly in a distributed way. As a result, *federated learning (FL)* [1] has emerged to take advantage of training with the decentralized data on mobile devices instead of gathering all data and training in the cloud. The federation of local devices protects the privacy and security of the raw data owned by mobile device users. More efficient use of the network bandwidth and lower latency can also be achieved by only communicating training updates instead of raw data [2]. Given the advantages, FL has seen application in the field of Internet of Things (IoT) [3], health AI diagnosis [4], etc.

One challenge for FL is the heterogeneous data distributions among local devices. Since the wildly adopted optimization method, stochastic gradient descent (SGD) [5], relies on independent and identically distributed (IID) data sampling, many research works assume IID data across training local devices for simplicity [1]. Nevertheless, this assumption hardly holds in reality. The training data on a specific mobile device is determined by its user. Thus, the local data can be highly heterogeneous (non-IID) across local devices. In addition to non-IID data, another challenge is the possibility of Byzantine attacks. For instance, an attacker can poison the learning process remotely by compromising a subset of the local devices and modifying their raw data or model updates before or during training. This can lead to the training process not being able to converge, or degrade the accuracy of the trained model on specific or all classes of data. Severe damage could be incurred in use cases like autonomous driving or smart healthcare if such attacks cannot be controlled.

Both non-IID data on benign devices and Byzantine attacks from compromised devices are non-trivial problems as shown in existing works [3], [6]–[8]. Yet, their combined effects can be even harder to address, since a central learner has no way of distinguishing between a benign device with non-IID data and a compromised device launching dedicated attacks. Solutions tackling one challenge alone may lead to poor performance when facing the other. For instance, robust algorithms for non-IID data may unintentionally include malicious data or model updates in training, and hence cannot converge or have low accuracy; Byzantine-robust learning algorithms may have to trim off too many non-IID data points or model updates, such that they have poor performance on rare but benign data. To our best knowledge, there is no existing work addressing the combined challenges of non-IID data and Byzantine attacks, which are both realistic risks to FL in real-world scenarios.

The goal of this paper is to propose a solution that can achieve both superior accuracy and robustness under Byzantine attacks and non-IID data. We present FedAegis, an edge-based hierarchical Byzantine-robust FL algorithm for non-IID data. Compared to traditional FL algorithms, FedAegis combines Byzantine-robust aggregation in FL with a distributionally robust loss function to address both challenges at the same time. The two techniques are further novelly combined using an edge-based 3-layer architecture, which utilizes edge computing to mitigate the impact of trimming in Byzantine-robust aggregation on the model robustness for benign but heterogeneous data. Our contributions are summarized as follows:

1) We design FedAegis, an edge-based hierarchical Byzantine-robust FL algorithm for non-IID data to tackle the combined challenges of heterogeneous data and Byzantine adversaries. Our solution novelly combines Byzantine-robust aggregation and a distributionally robust loss function under an edge-based 3-layer FL architecture.
2) We show experiment results with various datasets, data distributions and attack settings to validate the superior accuracy and robustness of FedAegis compared to state-of-the-art non-robust and robust FL algorithms.

The rest of this paper is organized as follows. Sec. II talks about background and related work. Sec. III describes the FL model, our edge system model and the threat model. Sec. IV explains details of our solution. Sec. V presents the evaluation results. Sec. VI concludes the whole paper.

II. Background and Related Work

Byzantine Adversaries and Defenses in FL. There exist two kinds of attacks against general machine learning models: data

Zhou, Li, Wang, Gu, Yu ({fzhou, zli85, xwang244, hgu5, ryu5}@ncsu.edu) are with North Carolina State University, Raleigh, NC 27606, USA. This research was supported in part by NSF grants 2007391 and 2045539. The information reported here does not reflect the position or the policy of the funding agency.

poisoning attacks and model poisoning attacks. Many works like [9], [10] aim to poison training data. In FL, however, data poisoning attacks could hardly succeed as described in [6] and validated in our results in Sec. V. Hence local model poisoning attacks are wildly adopted [6], [7], [11], [12]. As a few examples, Gaussian attacks [12] use tensors drawn from a Gaussian distribution in place of the original updates; omniscient attacks [12] use negative tensors of the original ones; inner product manipulation attacks [7] use negative tensors of the expectation of all benign workers' updates.

Byzantine-robust aggregation has been proposed to defend against general Byzantine adversaries in FL [12]–[14]. Defenses are primarily based on trimming a certain portion of model updates considered as outliers and hence likely malicious, or computing an aggregated model based on a majority of the received updates. Krum [12] for instance finds a local model closest to all benign models as the global model. Trimmed mean defense [13] trims a certain part of the smallest and largest values in the received model updates for each model parameter, and then takes the average of the rest as value of the parameter in the aggregated model. Median defense [13] chooses the median value of each parameter among all updates as the global model parameter value. These defenses, however, do not consider non-IID data, and hence may lead to low accuracy on non-IID data as shown in our evaluation.

Non-IID Data. Existing works [3], [8] have pointed out how non-IID data can degrade the performance of traditional FL algorithms such as FedAvg and FedSGD [1] compared to when data are IID across local devices. Reference [8] identified weight divergence as the root cause, which is due to the difference between the data distribution on each local device and the population distribution among all. Existing works have addressed this problem using a shared dataset among local devices [8], which is not practical in privacy-sensitive situations, as well as utilizing robust optimization techniques in the learning process [15], which nevertheless is vulnerable to Byzantine adversaries as we show in our evaluation.

III. System and Threat Models

A. Federated Learning Model

The traditional FL model consists of a central server and local devices that form a 2-layer architecture. In FL, the central server sends the global model parameters to the local devices as their initial model parameters in each communication round. After getting the initial model parameters, the local devices train the models with local data and then send the updated model parameters back to the server. The server then aggregates all model updates received from local devices to derive the new global model parameters for training in the next round. In the classic FedAvg algorithm, for instance, the central server takes the weighted average of the received local model updates as the global model parameters for the next round.

Formally, let W denote the global parameters kept by the central server, and let $\mathbf{\Psi} = \{\Psi^j \,|\, j = 1, \ldots, n\}$ be the set of model parameters of local devices, where n is the number of local devices. Denote the current communication round as t, and let W_t and $\mathbf{\Psi}_t = \{\Psi_t^j\}$ be global model parameters before update and the local model parameters of devices after update in round t respectively. The aggregation rule used by the central server is denoted as agg. Then we have:

$$W_{t+1} = agg(\mathbf{\Psi}_t). \tag{1}$$

The 2-layer FL architecture is depicted in Fig. 1.

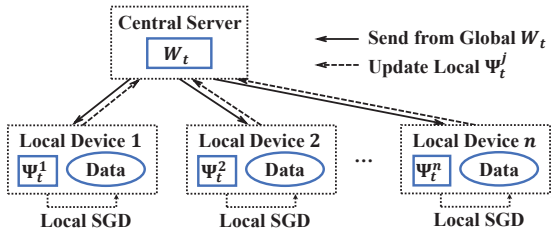


Fig. 1: Traditional 2-layer FL architecture

B. Edge System Model

Edge computing has been introduced in FL to reduce communication overhead in [16]. We assume local devices connect to edge nodes based on their geographical locations. Devices under each edge node may have IID or non-IID data distributions. Edge nodes perform intermediate aggregation between local devices and the central server. Let m be the number of edge nodes, and denote the set of aggregated model parameters of edge nodes as $\mathbf{\Theta} = \{\Theta^i \,|\, i = 1, \ldots, m\}$. The system model of an edge-based 3-layer FL architecture with attacks which will be explained in the next subsection is depicted in Fig. 2.

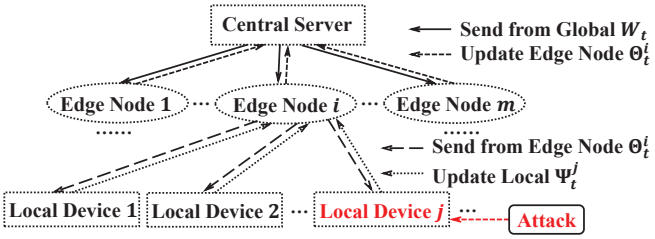


Fig. 2: Edge system model with attacks

Specifically, in each communication round, the central server broadcasts the global model parameters to local devices via the edge nodes. Local devices perform local training and then send the updated parameters to their connected edge nodes. Each edge node performs model aggregation for all connected local devices before sending the aggregated model parameters to the central server. Finally the central server aggregates parameters received from all edge nodes to update the global model which finishes one round of training. Note that the edge nodes and the central server can apply different aggregation rules. For edge node i and the central server, we define their aggregation rules as:

$$\Theta_t^i = agg_e^i(\mathbf{\Psi}_t); \quad W_{t+1} = agg_c(\mathbf{\Theta}_t), \tag{2}$$

where agg_e^i and agg_c are the aggregation rules used by edge node i and the central server, respectively.

C. Threat Model

We assume a small fraction of local devices are compromised by a Byzantine adversary, while the central server and the edge nodes are benign. The attacker can launch data poisoning attacks or local model poisoning attacks as described in Fig. 3. In a data poisoning attack, the attacker replaces benign data of each compromised local device with malicious data *before the training process starts*. In a local model poisoning attack, the attacker actively manipulates the model updates that each compromised device sends to the central server *during the training process*. The attacker's goal is to prevent the training process from converging or lower accuracy of the trained

model towards the entire target data distribution or specific classes of data. In this paper, we consider remote Byzantine adversaries who are unable to disturb the connections between local devices and edge nodes but can manipulate training data and the model update process of the local devices. These attackers widely exist in real world. For instance, an attacker can take control of a number of mobile devices remotely by exploiting hardware or software vulnerabilities on the devices. The attacker, however, cannot change the physical locations of the devices as well as which edge nodes they connect to.

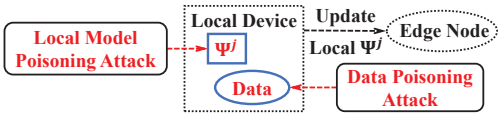


Fig. 3: Byzantine attacks on local devices

IV. Distributionally and Byzantine Robust Optimization

A. Overview

We propose an edge-based FL algorithm along with DRO, FedAegis, to deal with the aforementioned challenges. To tackle the Byzantine adversaries, we apply Byzantine-robust aggregation on the edge node layer to mitigate the manipulated updates. To handle non-IID data across benign devices, DRO is applied to the overall learning model to improve both average-case and worst-case accuracy. In our algorithm, the edge nodes not only reduce the communication overhead, but more importantly serve as the first line of defense against Byzantine adversaries who would attempt to destabilize learning or lower accuracy, while leaving as much benign information as possible for the central server to tackle the benign data heterogeneity with DRO. Under the assumption of a remote Byzantine adversary who cannot manipulate devices' geographical locations and connections to the edge nodes, this defense is expected to work well in most cases where not all compromised devices are connected to one or a few edge nodes. Next we introduce the detailed techniques used in building this algorithm.

B. Byzantine-Robust Aggregation Rules

FedAvg is not robust under Byzantine adversaries since it aggregates the malicious updates and fails to converge in the training process as shown later in our results. To defend against Byzantine adversaries, Byzantine-robust aggregation rules have been proposed in the literature as defenses:

Trimmed Mean [13]: The intuition is to trim out the potentially malicious local model updates and take the mean of the rest local model parameters as the global model parameters. It assumes the compromised models are likely the ones that are distinct from most of the other local models. For the ith parameter of the model, the central server first sorts all local models' ith parameters. The largest and smallest $P\%$ of the ith local model parameters are then removed and for the rest, the central server takes the average value as the ith parameter for the global model. The time complexity is $O(n \log n \cdot d)$ given n total devices and d model parameters.

Median Defense [13]: As its name, the median defense takes the median value of each parameter among all local models as the central server's parameter. Taking the ith parameter of the model as an example, the central server takes the median value among all local models' ith parameters as the ith parameter of the global model. The time complexity is $O(n \cdot d)$.

Other Defenses: Defenses such as Krum [12] and Bulyan [14] have high complexity and are unsuitable for large-scale FL. For instance, Krum picks a local model closest to potentially benign models *w.r.t.* Euclidean distance between model parameters as the global model. The time complexity is $O(n^2 \cdot d)$.

We apply a Byzantine-robust aggregation rule denoted as agg_{br} at each edge node to mitigate malicious updates early in the stage. Then, at the central server, we apply the FedAvg, which takes the average of the intermediate aggregated models uploaded by all edge nodes, weighted by the number of data points under each edge node. Denote d_i as the number of data points of all local devices under edge node i, and D as the total number of data points of all local devices. Then the aggregation rules in Eq. (2) can be rewritten as:

$$\Theta_t^i = agg_{br}(\mathbf{\Psi}_t); \quad W_{t+1} = \sum\nolimits_{i \in m} \frac{d_i}{D} \Theta_t^i. \tag{3}$$

C. Distributionally Robust Optimization

DRO is a technique that creates an uncertainty set based on the empirical distribution of data. It can increase the robustness of machine learning models by improving their performance under the worst-case data distribution within the uncertainty set. Inspired by [17], we design a DRO-based global loss function for our FL model.

Denote a machine learning model as $h(\omega)$ with parameters $\omega \in \Omega \subseteq \mathbb{R}^d$ where d is the parameter dimension and Ω is the parameter space; $\mathbb{R}$ is the real number set. Let the training loss function associated with learning model h and data point (x, y) be $L(h(x, \omega), y)$. The loss function denoted as F is then defined as the expectation of the loss for training data points $B = \{(x_k, y_k) \mid k = 1, \ldots, s\}$ of all local devices:

$$F(B, \omega) = \mathbb{E}_{(x_k, y_k) \in B}[L(h(x_k, \omega), y_k)]. \tag{4}$$

Let the empirical distribution of all training data be $\mathbb{P} = \{p_k = 1/s \mid k = 1, \ldots, s\}$. Define $\mathcal{Q} = \{\mathbb{Q} \mid \sum_{k=1}^{s} q_k = 1\}$ as the uncertainty set, where $\mathbb{Q} = \{q_k \in [0, 1] \mid k = 1, \ldots, s\}$ is an arbitrary probability distribution over the training data. The difference between the empirical distribution and an arbitrary $\mathbb{Q} \in \mathcal{Q}$ is defined via the Kullback–Leibler (KL) divergence. For discrete probability distributions M and N defined on the same probability space, $\mathcal{X}$, the KL divergence from N to M is defined as:

$$D_{\mathrm{KL}}(M \parallel N) = \sum\nolimits_{x \in \mathcal{X}} M(x) \log(M(x)/N(x)). \tag{5}$$

Given an arbitrary distribution $\mathbb{Q} \in \mathcal{Q}$, we apply the KL divergence between $\mathbb{Q}$ and $\mathbb{P}$ as a regularization term to the empirical loss function $F(B, \omega)$ over $\mathbb{Q}$, and define the DRO loss function as:

$$D_{\mathbb{Q}}(\omega) = \mathbb{E}_{(x_k, y_k) \sim \mathbb{Q}}[L(h(x_k, \omega), y_k)] - \frac{1}{\beta} D_{\mathrm{KL}}(\mathbb{Q} \parallel \mathbb{P}). \tag{6}$$

The hyper-parameter β along with the KL divergence controls the distance between the empirical training distribution and an arbitrary distribution in the created uncertainty set. The closer a distribution $\mathbb{Q}$ is to the empirical distribution, the smaller the value of the regularization term is.

The goal is to optimize the DRO loss function $D_{\mathbb{Q}}(\omega)$ over any distribution $\mathbb{Q}$ in the uncertainty set instead of just for the empirical distribution to achieve robust performance. In other words, we would like to find parameters ω of the model $h(\omega)$, which minimize the maximum DRO loss over all distributions

in the uncertainty set:

$$\arg\min_{\omega\in\Omega}\{\max_{\mathbb{Q}\in\mathcal{Q}}[D_{\mathbb{Q}}(\omega)]\}. \tag{7}$$

To solve this optimization problem, we need to find the optimal solution for the inner maximization problem and then minimize the maximum inner objective by solving for the optimal model parameters ω. Thus, we first derive the optimal solution $\mathbb{Q}^*$ for the inner maximization problem as:

$$\mathbb{Q}^* = \{q_k^* = softmax(\beta L(h(x_k,\omega), y_k)) \mid k = 1,\dots,s\}. \tag{8}$$

Derivation of Eq. (8): Given any ω, the inner problem can be rewritten as a constrained optimization problem:

$$\max_{\mathbb{Q}\in\mathcal{Q}} \sum\nolimits_{k=1}^{s} \{q_k[L(h(x_k,\omega),y_k)] - \frac{1}{\beta} q_k \log(sq_k)\}. \tag{9}$$

There exists a Lagrangian multiplier $\mu \in \mathbb{R}$ so that Eq. (10) below holds. To simplify, denote the loss $L(h(x_k,\omega),y_k)$ for data point (x_k,y_k) as $L_k(\omega)$. As Eq. (9) is strictly convex over $\mathbb{Q}$, based on the Karush–Kuhn–Tucker conditions, we have:

$$\begin{cases} L_k(\omega) - \frac{1}{\beta}(\log(sq_k)+1) + \mu = 0, \forall k = 1,\dots,s; \\ \sum\nolimits_{k=1}^{s} q_k = 1. \end{cases} \tag{10}$$

By solving Eq. (10), we can derive $\mathbb{Q}^*$ as in Eq. (8).

Given Eq. (8) as the optimal solution to the inner problem, we can then solve Eq. (7) as a general machine learning problem. The gradient of the DRO loss function is as follows:

$$\begin{aligned} \nabla_\omega D_{\mathbb{Q}^*}(\omega) &= \sum\nolimits_{k=1}^{s} \frac{\partial D_{\mathbb{Q}^*}(\omega)}{\partial L_k(\omega)} \nabla_\omega L_k(\omega) \\ &= \sum\nolimits_{k=1}^{s} q_k^* \nabla_\omega (L_k(\omega)) = \mathbb{E}_{(x_k,y_k)\sim\mathbb{Q}^*}[\nabla_\omega(L_k(\omega))]. \end{aligned} \tag{11}$$

As we can see from Eq. (11), with data sampled based on $\mathbb{Q}^*$, $\nabla_\omega L_k(\omega)$ is an unbiased estimation of the gradient of DRO loss. Therefore, the optimization problem in Eq. (7) can be solved using SGD via sampling data points based on the distribution $\mathbb{Q}^*$. However, since $\mathbb{Q}^*$ will not available until one forward propagation over the entire training dataset B, we use historical data to approximate $\mathbb{Q}^*$ for the current round [17]. Let t_k be the last round in which data point (x_k,y_k) is sampled. In round t, we approximate $\mathbb{Q}^*$ with $\hat{\mathbb{Q}}^* = \{\hat{q}_k^*\}$, where:

$$\hat{q}_k^* = \exp(\beta L_k(\omega_t)) / \sum\nolimits_{\kappa} \exp(\beta L_\kappa(\omega_{t_\kappa})). \tag{12}$$

In Eq. (12), the denominator of the softmax in Eq. (8) is replaced by the sum of stale exponentiated loss $L_\kappa(\omega_{t_\kappa})$ of each data point (x_κ,y_κ) when it was last sampled, while the numerator is the current loss $L_k(\omega_t)$ of the sampled data point (x_k,y_k). Ideally, we would like to sample from $\hat{\mathbb{Q}}^*$ during training based on the above derivation. However, this is difficult to implement in FL since each local device independently decides which data points to sample. Instead, in Algorithm 1, we apply uniform sampling for each local device, and applies $\hat{q}_k^*$ as a multiplicative weight to the loss gradient of data point (x_k,y_k). This, however, requires each local device to obtain the historical global information $EL = \sum_\kappa \exp(\beta L_\kappa(\omega_{t_\kappa}))$. To realize this, each local device needs to send its own exponentiated loss sum, $EL_{i,j} = \sum_{\kappa\in B_j} \exp(\beta L_\kappa(\omega_{t_\kappa}))$, where B_j is denoted as the dataset of local device j, to the central server via edge node i after every round. The central server then aggregates the exponentiated losses, and then sends the aggregated exponentiated loss EL back to each local device in the next round. Directly applying the weights would result in a very small value for the loss gradient (corresponding to a significantly reduced learning rate). Thus, we normalize $\hat{q}_k^*$ to $\bar{q}_k^* = s\hat{q}_k^* = \exp(\beta L_k(\omega_t))/\overline{EL}$ by the total number of data points s of all devices, where $\overline{EL} = EL/s$, to keep the same scale of learning rate as sampling from $\hat{\mathbb{Q}}^*$. This way, each local device obtains the global information to compute $\bar{q}_k^*$ of each sampled local data point, which is then multiplied to the loss gradient of each data point before the local updates.

Algorithm 1: FedAegis

Input: training dataset $B = \bigcup_{j=1}^{n} B_j$ with a total number s; batch size b; learning rate η; DRO hyper-parameter $\beta > 0$; weight clipping range $[q_{min}, q_{max}]$; max communication round T.

1 Initialize $\overline{EL} \leftarrow 1$;
2 **for** *communication round* $t = 1,2,\dots,T$ **do**
3 central server broadcasts $(W_t, \overline{EL})$ to edge nodes;
4 **for** $\forall$*edge node* i *in parallel* **do**
5 edge node broadcasts $(W_t, \overline{EL})$ to local devices;
6 **for** $\forall$*device* j *under edge node* i *in parallel* **do**
7 uniformly sample a batch of size b in B_j;
8 **for** *data point* (x_k,y_k) *in the batch* **do**
9 $\bar{q}_k^* \leftarrow \exp(\beta L_k(W_t))/\overline{EL}$;
10 $\bar{q}_k^* \leftarrow$ w_clip $(\bar{q}_k^*, [q_{min}, q_{max}])$;
11 $EL_{i,j} \leftarrow \sum_k \exp(\beta L_k(W_t))$;
12 $\Psi_t^j \leftarrow W_t - \eta \cdot \frac{1}{b}\sum_k \bar{q}_k^* \nabla L_k(W_t))$;
13 device j sends (Ψ_t^j, EL_j) to edge node i;
14 $\Theta_t^i \leftarrow agg_{br}(\mathbf{\Psi}_t)$, $EL_i \leftarrow \sum_j EL_{i,j}$;
15 edge node i sends (Θ_t^i, EL_i) to central server;
16 $W_{t+1} \leftarrow \sum_{i\in m} \frac{d_i}{D}\Theta_t^i$, $\overline{EL} \leftarrow \frac{1}{s}\sum_i EL_i$, ;

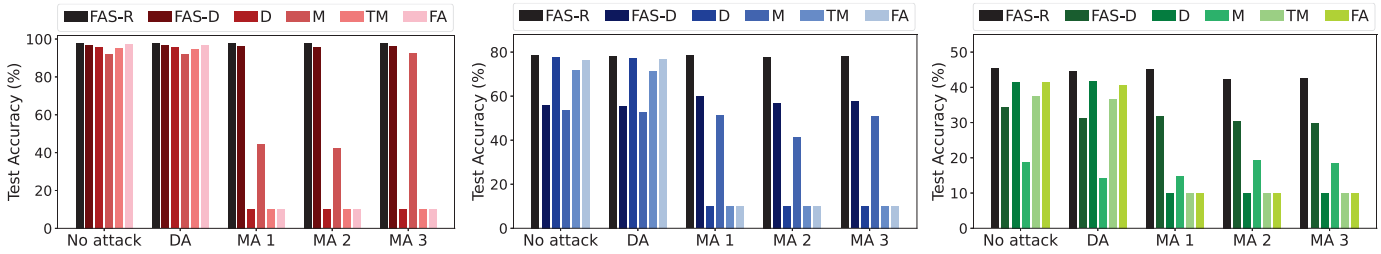
D. FedAegis: Edge-based Byzantine-robust FL with DRO

Combining Byzantine-robust aggregation with DRO, our edge-based FL algorithm, FedAegis, is shown in Algorithm 1. In FedAegis, a communication round starts with the central server broadcasting the global model parameters W_t and the $\overline{EL}$ in Lines 3 and 5. The local training on each device is in Lines 6-12. Line 9 calculates the weight of each sampled data point for its loss gradient. To tackle instability that these weights bring, we use weight clipping [17] to clip the weight values that are beyond a range of $[q_{min}, q_{max}]$ in Line 10. Line 13 updates the local model parameters and $EL_{i,j}$ under edge node i. Edge node i then aggregates the model parameters using median defense and calculates $EL_i = \sum_j EL_{i,j}$ in Line 14. In Line 15, the aggregated parameters and EL_i for edge node i are sent to the central server. Next, the central server aggregates parameters from all edge nodes via FedAvg in Line 16 and obtains the updated global model parameters W_{t+1}, meanwhile calculating $\overline{EL} = \frac{1}{s}\sum_i EL_i$ for the next round.

V. Performance Evaluation

A. Experiment Setup

1) Models and datasets: We used convolutional neural networks (CNNs) to train and test on three datasets with their default training-testing data splitting: MNIST [18], FMNIST [19] and CIFAR-10 [20]. These are benchmarks for supervised image classification. Specifically, CNNs with 2 convolutional layers and 2 or 3 fully connected layers were



(a) MNIST dataset (b) FMNIST dataset (c) CIFAR-10 dataset

Fig. 4: Average test accuracy for the three datasets under non-IID data setting and different attack/no-attack scenarios

used for MNIST, FMNIST and CIFAR-10, with 21840, 12128 and 62006 total parameters respectively. More complex models could be used for higher accuracy. In MNIST and FMNIST, there were 600 local devices each with 100 data points, and 20 edge nodes each connected to 30 devices. In CIFAR-10, there were 500 devices each with 100 data points and 10 edge nodes each connected to 50 devices. To simulate non-IID data for local devices, we divided each dataset into 5 sub-datasets, each containing data points of only 2 classes. In all algorithms, 50 devices were randomly picked in one round. In our 3-layer architecture, 5 edge nodes with 10 devices under each were randomly picked in one round. Default hyper-parameters are: $\eta = 0.01$; $\beta = 10$; $q_{min} = 0.6$; $q_{max} = 6$; $T = 500$.

2) Byzantine Adversaries: We implemented one type of data attack and three types of local model attacks. Denote the set of compromised local devices before attack as A, and benign local devices as Γ with $|\Gamma| = P \cdot n$. The default percentage of compromised devices for each dataset was: $P = 10\%$ for MNIST, $P = 6\%$ for FMNIST and $P = 1\%$ for CIFAR-10. Compromised devices were distributed evenly under each edge node. The details of the attacks are as follows.
Gaussian Attack: The attacker replaced the weight updates of compromised devices with tensors drawn from a zero-mean Gaussian distribution. The corrupted local weight update for device $\alpha \in A$ is $\hat{\Psi}^\alpha = \mathcal{N}(0, \sigma^2)$, where σ was the attack strength with a default value of 200.
Omniscient Attack: The weight updates were replaced by the negative of it multiplied by a scalar S denoting the attack strength with a default value of 10. The corrupted local weight update for device $\alpha \in A$ is $\hat{\Psi}^\alpha = -S \cdot \Psi^\alpha$.
Inner Product Manipulation (IPM) Attack: The attacker replaced weight updates with negative tensors of average of all benign weight updates. IPM attack was based on the strong assumption that the attacker can eavesdrop on the updates of all devices. A scalar S was used to adjust the attack strength with a default value of 20. The corrupted local weight update for device $\alpha \in A$ is $\hat{\Psi}^\alpha = -\frac{S}{n(1-P)} \sum_{\gamma \in \Gamma} \Psi^\gamma$.
Data Attack: We launched a *label flipping attack* where the labels of data points on each compromised local device were flipped from one class to another chosen by the attacker.

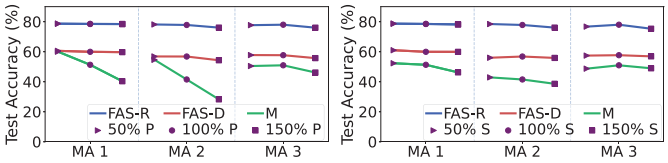
In our results, we denote Gaussian attack, Omniscient attack, IPM attack and data attack as **MA 1**, **MA 2**, **MA 3** and **DA**, respectively. We compared our algorithm with several state-of-the-art non-robust and robust FL algorithms. FedAvg with a cross-entropy loss function, is denoted as **FA**. When DRO is added to the original loss function, we denote it as **D**. When trimmed mean defense is used but without DRO, we denote it as **TM**; similarly when median defense is used, we denote it as **M**. Our algorithm is denoted as **FAS**. Noting that our algorithm could be sensitive to the distribution of local devices under the edge nodes, we considered two typical distributions of the local devices: a *random* distribution where local devices under an edge node possessed data from randomly selected sub-datasets, and a *disparate* distribution where each edge node only served devices with data from the same sub-dataset. We denote our algorithm applied in the former case as **FAS-R** and the latter as **FAS-D**.

3) Metrics: We evaluated the performance of different learning algorithms using accuracy and F_1 score under the test datasets. Specifically, to evaluate the robustness of learning algorithms against heterogeneous data, we evaluated the F_1 score of the worst-performing classes in testing. The balanced F-measure is widely used in the literature, which is the harmonic mean of precision and recall: $F_1 = 2/(recall^{-1} + precision^{-1})$. For our ten-class datasets, instead of the balanced F-measure which aims two-class classification problem, the unbalanced F-measure was used instead. Consider a subset of classes C, and let the probability of a data point belonging to any class in C be ρ. Let r be the probability that our prediction is correct. For instance, assume all classes have the same number of data points, then for a single class $C = \{c\}$, we have $\rho = 0.1$. This leaves us with: True Positive $= n\rho r$; False Negative $= n\rho(1-r)$; False Positive $= n(1-\rho)r$; True Negative $= n(1-\rho)(1-r)$. Therefore, we have the unbalanced F-measure: $F_1' = 2r\rho/(r+\rho)$. From this equation, F_1' is monotonously increasing with $r \in (0, 1)$. The maximum value of the unbalanced F-measure is: $F_{1\max}' = 0.1819$ with $r = 1, \rho = 0.1$. For our unbalanced F-measure, every empirical F_1 value was divided by $F_{1\max}'$ to make a normalization.

B. Evaluation Results

1) Average test accuracy: The average test accuracy, which is the average accuracy of all test devices, for MNIST, FMNIST and CIFAR-10 under default percentage of compromised devices and attack strength is in Fig. 4. When facing non-IID data without attack, **FA** performed better than **TM** and **M** for all three datasets. This shows that trimming off local model updates as in **TM** and **M** leads to lower accuracy. Among all attacks, the data attack was relatively weak, and could be defended by all algorithms even including **FA**. This is consistent with the observations in [6]. Under local model attacks and non-IID data, **FA** and **D** with no defense were not able to converge. The **TM** defence was also non-converging when facing the combined effects of both non-IID data and the attacks. Noting that a test accuracy of 10% means a random guess and indicates the non-convergence of **FA**, **D** and **TM** under attacks. **M** alleviated the non-convergence problem but

was not able to maintain a high accuracy on the test datasets. Under both random and disparate device distributions across edge nodes, our algorithm (**FAS-R** and **FAS-D**) managed to tackle the two challenges at the same time and remained Byzantine-robust as well as achieved high accuracy at all times.



(a) Percentage of attack devices P (b) Attack strength S

Fig. 5: FMNIST under different attack settings

We further used FMNIST to test different attack settings to show effectiveness of our algorithm, including with 50% or 150% of the default percentage of attack devices (P) or the attack strength (S), as shown in Fig. 5. We only show the results of **FAS-R**, **FAS-D** and **M**, since **D**, **TM** and **FA** are not robust under attacks from Fig. 4. The accuracy of **M** decreased as P or S went up. **FAS-R** and **FAS-D** remained the same level of test accuracy under different settings, further showing their robustness under different attack power with non-IID data.

2) Worst-class performance: In Table I, we show the average worst one, two and three classes normalized F_1 scores for the test datasets. Only algorithms **FAS-R**, **FAS-D** and **M** are shown since the others are not robust under attacks. In almost all cases, our algorithm (**FAS-R** and **FAS-D**) reached the best average worst-class normalized F_1 scores. M in rare cases may slightly outperform for $k=3$ worst classes, by sacrificing performance of the worst 1 or 2 classes. This indicates that our algorithm improved the worst-case performance for heterogeneous data under attacks, further demonstrating the strong robustness advantage of our algorithm.

Setting \ Metric		Average Worst k-class Normalized F_1 Score								
		$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$
Attack	Method	MNIST			FMNIST			CIFAR-10		
MA 1	**FAS-R**	**5.335**	**5.366**	**5.392**	**1.551**	**2.375**	**2.799**	**0.965**	**1.056**	**1.312**
	FAS-D	**5.105**	**5.168**	**5.177**	**0.147**	**0.565**	**0.866**	**0.096**	**0.116**	**0.385**
	M	1.778	1.862	1.912	0.083	0.535	0.777	0.003	0.008	0.014
MA 2	**FAS-R**	**5.320**	**5.352**	**5.380**	**1.209**	**2.194**	**2.663**	**0.620**	**0.763**	**1.055**
	FAS-D	**5.205**	**5.230**	**5.254**	**0.103**	**0.488**	0.580	**0.076**	**0.090**	**0.355**
	M	1.691	1.783	1.834	0.043	0.391	**0.767**	0.011	0.024	0.047
MA 3	**FAS-R**	**5.339**	**5.382**	**5.409**	**1.649**	**2.465**	**2.882**	**0.553**	**0.672**	**0.965**
	FAS-D	**5.309**	**5.316**	**5.330**	**0.197**	**0.538**	0.781	**0.056**	**0.067**	**0.340**
	M	4.619	4.745	4.832	0.172	0.496	**0.999**	0.004	0.010	0.029

TABLE I: Average worst k-class normalized F1 score for the three datasets under different attacks and non-IID data

VI. Conclusions

In this paper, we proposed an edge-based hierarchical federated learning algorithm, FedAegis, to tackle the problems of Byzantine adversaries and heterogeneous data distributions in FL. FedAegis applies Byzantine-robust aggregation at the edge nodes to defend against Byzantine adversaries early in the communication round. This serves to stop the attack as early as possible meanwhile leaves sufficient information for the central server to further tackle data heterogeneity across benign devices. To address the heterogeneous data distributions, we applied distributionally robust optimization to customize the loss function of the training model, which defines an uncertainty set around the empirical data distribution to capture the different devices' data distributions. We evaluated FedAegis and compared it to state-of-the-art non-robust and robust FL algorithms on three datasets: MNIST, FMNIST and CIFAR-10. The results showed that FedAegis could not only improve the test accuracy and the worst-case performance for non-IID data, but also remained robust under Byzantine attacks, outperforming state-of-the-art algorithms which either did not converge or had low average and/or worst-case performance when facing both non-IID data and Byzantine attacks.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *PMLR AISTATS*, 2017, pp. 1273–1282.

[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 2020.

[3] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 31, no. 9, pp. 3400–3413, 2019.

[4] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Elsevier Int. J. Med. Inform.*, vol. 112, pp. 59–67, 2018.

[5] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Springer Proceedings of COMPSTAT*, 2010, pp. 177–186.

[6] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *USENIX Security*, 2020, pp. 1605–1622.

[7] C. Xie, O. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," in *PMLR UAI*, 2020, pp. 261–270.

[8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[9] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *IEEE S&P*, 2018, pp. 19–35.

[10] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[11] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[12] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *NeurIPS*, vol. 30, 2017.

[13] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *PMLR ICML*, 2018, pp. 5650–5659.

[14] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *PMLR ICML*, 2018, pp. 3521–3530.

[15] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review," *arXiv preprint arXiv:1908.05659*, 2019.

[16] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, "Edgefed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209 191–209 198, 2020.

[17] L. Fidon, S. Ourselin, and T. Vercauteren, "Distributionally robust deep learning using hardness weighted sampling," *arXiv preprint arXiv:2001.02658*, 2020.

[18] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[19] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[20] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, "Cifar-10: Knn-based ensemble of classifiers," in *IEEE CSCI*, 2016, pp. 1192–1195.