MITIGATING DATA INJECTION ATTACKS ON FEDERATED LEARNING

Or Shalom[1], *Amir Leshem*[2], *Waheed U. Bajwa*[3]

[1]shalomo7@biu.ac.il; [2]amir.leshem@biu.ac.il; [3]waheed.bajwa@rutgers.edu
[1,2]Faculty of Engineering, Bar-Ilan University, Ramat Gan 5290002, Israel
[3]Dept. of Electrical & Computer Engineering, Rutgers University–New Brunswick, NJ 08854 USA

ABSTRACT

Federated learning is a technique that allows multiple entities to collaboratively train models using their data without compromising data privacy. However, despite its advantages, federated learning can be susceptible to false data injection attacks. In these scenarios, a malicious entity with control over specific agents in the network can manipulate the learning process, leading to a suboptimal model. Consequently, addressing these data injection attacks presents a significant research challenge in federated learning systems. In this paper, we propose a novel approach to detect and mitigate data injection attacks on federated learning systems. Our mitigation strategy is a local scheme, performed during a single instance of training by the coordinating node, allowing for mitigation during the convergence of the algorithm. Whenever an agent is suspected of being an attacker, its data will be ignored for a certain period; this decision will often be re-evaluated. We prove that with probability one, after a finite time, all attackers will be ignored while the probability of ignoring a trustful agent becomes zero, provided that there is a majority of truthful agents. Simulations show that when the coordinating node detects and isolates all the attackers, the model recovers and converges to the truthful model.

Index Terms— Attack Detection, Data Injection Attacks, Federated Learning, Provable Security

1. INTRODUCTION

Big-data processing capabilities have increased significantly over the past years due to the increasing volume and variety of data that is being generated and collected. Today, data has become an asset, and processing that data is required in many, if not all, the industries affecting our lives, such as healthcare, finance, transportation, manufacturing, and many more. With the increased need for data, a new need for the privacy and security of the data has risen [1, 2].

Federated learning is a popular approach for collaboratively training machine learning models while preserving data privacy [3, 4]. In this approach, instead of training a centralized model using a combined dataset as traditionally done, multiple independent agents train local models using their private datasets, which are assumed to be statistically independent. The agents exchange the local model parameters (i.e., weights and biases) with a centralized node coordinating the learning process, which in turn returns a new model (or model updates) to all the agents (cf. Figure 1). Although the agents benefit from the training performed on other agents' data, the datasets are not shared and remain private.

Even though the federated learning model preserves privacy, it is vulnerable to various security threats, including data injection and poisoning attacks [5–8], backdoor access [9, 10], gradient attacks [11] and many more [12, 13]. In data injection attacks, malicious participants (agents) inject false data into the training process to manipulate the global model. Detecting data injection attacks in federated learning is challenging due to the distributed nature of the data across multiple devices. Although a coordinating node oversees the training process, it does not have access to the complete dataset, limiting its ability to comprehensively monitor for these attacks. This has led to the development of various techniques for detecting and preventing data injection attacks, including model-based methods [14, 15], anomaly detection techniques [16], Byzantine resilient methods [17–20], and federated outlier detection [21, 22].

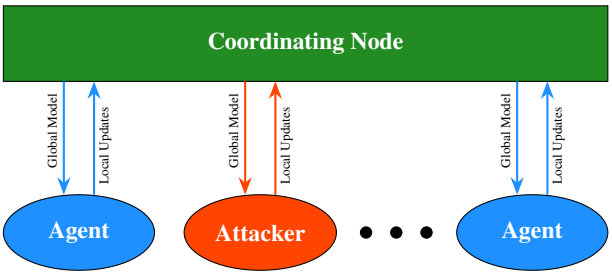


Fig. 1: A simple federated learning system. Each agent performs training on its private dataset; the local updates are then transmitted to a coordinating node, which returns a global model. Some of the agents may be malicious, meaning they might send unreliable updates to the coordinating node.

Previous works on detecting data injection attacks in federated learning have made notable contributions. For instance, Tolpegin et al. [7] proposed a PCA-based detector. Based on the difference in agents' parameter updates, they were able to separate the parameter updates into two clusters, one comprising malicious agents and the other consisting of trustworthy agents. Yar et al. [10] suggested a modified dual "gradient clipping defense". Standard clipping defense is a scheme specifically designed for data poisoning attacks, in which agents that send updates with norms too high are clipped. Yar et al. showed that in a dynamic network, a dual threshold clipping defense with one smaller threshold for neighboring agents and another larger threshold for the global model achieves better results.

In this paper, we present a novel detection method for data injection attacks in federated learning. The method is performed along the model training process and is based on evaluating the gradient of the updates of the participating agents, comparing this gradient to the coordinatewise median over the agents, and ignoring updates from suspicious agents. Considering the history of detections using a majority voting among the coordinating node's decisions, we

This work is partially supported by ISF grant 2197/22.

can overcome false alarms and missed detections. We prove convergence to a truthful model with probability one, provided that data is independent and identically distributed (i.i.d.) among agents. It could be argued that the assumption of i.i.d. data might impose limitations. However, many theoretical papers use i.i.d. data for the theoretical analysis (see [7, 23]. This has also been the case with signal processing adaptive algorithms where i.i.d. data is widely assumed. The approach taken generalizes the decentralized optimization for M-estimators in [16] to the federated learning context. We also demonstrate by simulations that the proposed technique can overcome constant output and label-flipping attacks [5–13], even when these attacks are hidden with partially truthful responses.

2. PROBLEM FORMULATION

2.1. Federated Learning

The federated learning problem involves learning a model using agents' private data. In this setting, each agent, using its private data, refines its model parameters. These local updates are then transmitted to the coordinating node, denoted as agent 0, which synthesizes them into a global model. This process ensures that only the model's parameters, not the data itself, are shared among agents.

Consider a dataset $\mathcal{D}$ labeled with labels from a set C. Within this framework, agents $1, \ldots, N$—referred to as the edge agents—iteratively refine their model parameters during the learning phase using this labeled dataset. The objective typically involves minimizing a function using a gradient descent approach:

$$\min_W F(W), \text{ where } F(W) := \sum_{k=1}^{N} p_k F_k(W), \tag{1}$$

where N represents the number of participating agents, $\sum_k p_k = 1$, and F_k is the local empirical risk function for the k-th agent. Although $p_k = \frac{1}{N}$ is common, varying these values can prioritize the risk of certain agents. At each time step, after completing their learning phase, the edge agents broadcast their model parameters to the coordinating node, which then computes and returns the averaged model parameters to all agents.

2.2. Data Injection Attacks

The federated learning approach was previously shown to achieve excellent results [24, 25], particularly when all collaborating agents share the same goal. However, consider a scenario where some agents participating in federated learning are malicious. We use the following notation: Denote the set of attackers $A \subset 1, \ldots, N$ and let $n_a = |A|$. We assume that $0 \leq n_a < N/2$, and $n_t = N - n_a$ is the number of trustworthy agents.

In this context, since attacking agents must coordinate to achieve a false model, we can assume, without loss of generality, a single malicious agent, say agent a, influencing the joint model training. Agent a's goal is to manipulate the joint model training, thus preventing it from performing successfully by transmitting false parameters, voiding the convergence of the model around an optimal point, and steering it towards a false model with predetermined performance. This agent is capable of executing various destructive attack schemes, such as "label flipping attack", "constant output attack", "randomized attack" and more (see [5–8]).

In a randomized attack, the attacker broadcasts a random set of model parameters back to the other agents, a tactic that rapidly degrades the learned model's performance. However, this type of attack is relatively easy to detect, as the malicious agents' responses stand out significantly from those of the trustworthy ones. The constant output attack aims to steer the system towards a model that consistently classifies a single class c irrespective of the data. Lastly, the more challenging to detect label flipping attack involves choosing a specific permutation of the labels. The attacker responds as if this permutation has been applied to the training data's output or by sending a model trained to recognize the permuted class values. For example, in an attack on MNIST, this could involve using a model that classifies class c as class $(c+2)$ mod 10 rather than its true value c for any given training data point.

A stronger type of attack would try to hide the previous attacks by combining a false model and a true one. This reduces the statistical discrepancy between the malicious agents' response and the other agents' response, by adding a bias to the reported model. While the attacker's main goal is to manipulate the joint model parameters and prevent convergence to a steady optimal point, it also has a secondary goal of remaining hidden and disguising the attack. In this strategy, let $W_{a,r}(t)$ represent the model parameters that the attacker, posing as a regular agent, would have updated by reliably updating the model $W(t-1)$ provided by the coordinating agent with correct data at time t, while $W_{a,f}$ is a pre-trained false model, classifying labels according to the attacker's desired attack scheme.[1]

Building on this concept, the attack can be formally described as follows: A malicious agent a responds at time t by sending

$$W_a(t) := g(t)W_{a,r}(t) + (1 - g(t))W_{a,f}, \tag{2}$$

where $W_a(t)$ denotes the set of parameters transmitted by agent a at time t, and $g(t)$ is a time-varying mixing weight satisfying the following conditions:

- For all $t \geq 0$, $0 \leq g(t) \leq 1$,
- $g(0) \equiv 1$, $\lim_{t\to\infty} g(t) = 0$, and
- $g(t)$ decreases over time, i.e., $g(t+1) \leq g(t)$.

For instance, delaying the start of the attack to time T_a can be achieved by setting $g(t) = 1, 0 \leq t \leq T_a$. While the monotonicity of $g(t)$ can be relaxed, it is essential for ensuring convergence to the attacker's desired model.

Note that this attack scheme is realistic as the attacker has no access to other agents' datasets and it can't manipulate the model learning process or the parameters aggregation done by the coordinating agent. The attacker is only capable of pre-learning a manipulated model and it doesn't rely on a specific neural network configuration, optimization function, or loss function.

3. ATTACKER DETECTION AND AVOIDANCE

In our proposed method, the coordinating agent compares the updates received from edge agents over time. The private datasets are assumed to be identically distributed and therefore if an agent is malicious and its model parameters update differently, it will stand out and be considered malicious.

To localize the attacker, we propose a low-complexity metric, computed over time by the coordinating agent once every ΔT updates. When the coordinating agent suspects an edge agent to be an attacker, it ignores its parameter updates for the next ΔT updates. Let $I_k = [(k-1)\Delta T + 1, k\Delta T]$. Define the two hypotheses tested over the interval I_k:

$\mathcal{H}^0_{j,k}$ – agent j is trustworthy.

$\mathcal{H}^1_{j,k}$ – agent j is malicious.

[1] For simplicity of notation we assume $W_{a,f}$ is time independent, but it can also be dependent.

The proposed detection metric for a given interval I_k, computed over time for agent j's model parameters, is given by

$$\Delta U_{j,k} := \frac{1}{\Delta T}\sum_{t\in I_k} U_j(t) \underset{\mathcal{H}_j^1}{\overset{\mathcal{H}_j^0}{\lessgtr}} \delta_u\sqrt{N}, \tag{3}$$

where

$$U_j(t) := \|\Delta W_j(t) - \text{median}\{\Delta W_\ell(t) : \ell \in \{1,\dots,N\}\backslash\{j\}\}\|_\infty. \tag{4}$$

Here, the median is a coordinatewise operation, $\Delta W_j(t) := W_j(t) - W_j(t-1)$, and δ_u is a predefined threshold. The following lemma characterizes the probability of accurately identifying malicious agents using the metric ΔU_j under certain assumptions.

Lemma 1 *When attacker(s) are present in the network, the probability of False-Alarm and the probability of attacker Detection can be bounded as*

$$\begin{aligned} P_{FA}(k) &= \mathbb{P}\left(\Delta U_{j,k} > \delta_u\sqrt{N}|\mathcal{H}_j^0\right) \leq \\ &\leq 2\exp\left(-\frac{\delta_u^2 N}{2r_a(t_k^*)}\right) \xrightarrow[k\to\infty]{} 0, \end{aligned} \tag{5a}$$

$$\begin{aligned} P_D(k) &= \mathbb{P}\left(\Delta U_{j,k} > \delta_u\sqrt{N}|\mathcal{H}_j^1\right) \geq \\ &\geq 1 - \exp\left(-\frac{\max\{0, -\delta_u\sqrt{N} + [\mu_r(t_k^\dagger)]_{s^*}\}^2}{2[r_r(t_k^\dagger) + r_a(t_k^\dagger)]}\right) \xrightarrow[k\to\infty]{} 1, \end{aligned} \tag{5b}$$

where the time indexes $t_k^* = \arg\max_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty$, $t^\dagger = \arg\min_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty$ are time indexes with the highest/lowest difference between the parameters-updates in interval I_k respectively, $s^* = \arg\max_s |[\Delta W_j(t^\dagger) - \Delta W_l(t^\dagger)]_s|$ and μ_r, r_r, r_a are sub-Gaussian parameters defined in the proof of the Lemma.

Proof (Lemma 1) *As previously stated, the attackers' broadcast a pre-chosen model together with a random noise imitating the trsutworthy agents' parameters convergence rate i.e. $W_f(t)$ is a sub-Gaussian random variable with mean W_f and parameter $r_r(t)$. This allows us to define $\Delta W_f(t) = W_f(t) - W_f(t-1)$ as a zero mean sub-Gaussian random variable with parameter $2r_r(t)$.*

The trustworthy agents on the other hand are updating according to a gradient descent algorithm with a learning factor α, i.e.

$$W_r(t+1) = W_j(t) - \alpha\nabla F(W_j(t)). \tag{6}$$

Without loss of generality, we assume that the trustworthy agents are trying to converge to an optimal model with parameters W_r, wether there are attackers present in the network or not.

Define the trustworthy agents' model parameters update (where there are n_a attackers present in the network) as

$$\Delta W_r(t+1) = \frac{n_a}{N}\left[W_f(t) - W_r(t)\right] - \alpha\nabla F(W_j(t)). \tag{7}$$

From this definition, we see that the trustworthy agents' parameters update can be described as a sub-Gaussian random variable with mean $\mu_r(t)$ and some parameter $r_a(t)$, where

$$\mu_r(t) = \frac{n_a}{N}[W_f - W_r] - \alpha\mathbb{E}[\nabla F(W_j(t))] \neq 0. \tag{8}$$

Define the probability of False-Alarm at the I_k interval as the probability for a trustworthy agent to be marked as an attacker,

$$P_{FA}(k) = \mathbb{P}\left(\Delta U_{j,k} > \delta_u\sqrt{N}|\mathcal{H}_j^0\right). \tag{9}$$

Assuming that the majority of agents are trustworthy, the median of the parameters-update ΔW_l, depicting the parameters-update median given in Equation (4) is distributed similarily to a trustworthy agent parameters-updates. This assumption allows us to bound the probability of False-Alarm as

$$\begin{aligned} P_{FA}(k) &= \mathbb{P}\left(\Delta U_{j,k} > \delta_u\sqrt{N}|\mathcal{H}_j^0\right) = \\ &= \mathbb{P}\left(\frac{1}{\Delta T}\sum_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty > \delta_u\sqrt{N}\Big|\mathcal{H}_j^0\right) \leq \\ &\leq \mathbb{P}\left(\|\Delta W_j(t_k^*) - \Delta W_l(t_k^*)\|_\infty > \delta_u\sqrt{N}|\mathcal{H}_j^0\right) \leq \\ &\leq 2\exp\left(-\frac{\delta_u^2 N}{2r_a(t_k^*)}\right) \xrightarrow[k\to\infty]{} 0, \end{aligned} \tag{10}$$

where $t_k^* = \arg\max_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty$ is a time index with the highest difference between the parameters-updates in interval I_k, the last transition is using Hoeffding's bound to bound the probability as $[\Delta W_j(t_k^*) - \Delta W_l(t_k^*)]$ is a zero-mean sub-Gaussian random variable with parameter $2r_a(t_k^*)$. Note that as we eliminate the attackers, the trustworthy agents converges to the optimal parameter and $r_a(t) \xrightarrow[t\to\infty]{} 0$.

Similarily for the probability of False-Alarm, we define the probability of Detection at the I_k interval as the probability to detect an attacker,

$$P_D(k) = \mathbb{P}\left(\Delta U_{j,k} > \delta_u\sqrt{N}|\mathcal{H}_j^1\right). \tag{11}$$

In order to bound the probability of Detection, we look at the probability of Miss-Detection,

$$P_{MD}(k) = 1 - P_D(k) = \mathbb{P}\left(\Delta U_{j,k} < \delta_u\sqrt{N}|\mathcal{H}_j^1\right). \tag{12}$$

Under the previous assumption, for a trustworthy agent ΔW_l we get

$$\begin{aligned} P_{MD}(k) &= \mathbb{P}\left(\Delta U_{j,k} < \delta_u\sqrt{N}|\mathcal{H}_j^1\right) = \\ &= \mathbb{P}\left(\frac{1}{\Delta T}\sum_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty < \delta_u\sqrt{N}\Big|\mathcal{H}_j^1\right) \leq \\ &\leq \mathbb{P}\left(\left\|\Delta W_j(t_k^\dagger) - \Delta W_l(t_k^\dagger)\right\|_\infty < \delta_u\sqrt{N}|\mathcal{H}_j^1\right), \end{aligned} \tag{13}$$

where $t^\dagger = \arg\min_{t\in I_k}\|\Delta W_j(t) - \Delta W_l(t)\|_\infty$ is a time index with the lowest difference between the parameters-updates in interval I_k. We mark as $\rho = |[\Delta W_j(t^\dagger) - \Delta W_l(t^\dagger)]_{s^*}|$ where $s^* = \arg\max_s |[\Delta W_j(t^\dagger) - \Delta W_l(t^\dagger)]_s|$. Those definitions allows us to bound Equation (13) using Chernoff's bound as

$$\begin{aligned} P_{MD}(k) &\leq \mathbb{P}(|\rho| < \delta_u\sqrt{N}) \leq \\ &\leq \mathbb{P}(\rho - \mu_\rho > -\delta_u\sqrt{N} + |\mu_\rho|) \leq \\ &\leq \frac{\mathbb{E}[\exp(s(\rho - \mu_\rho))]}{\exp(s(-\delta_u\sqrt{N} + |\mu_\rho|))}. \end{aligned} \tag{14}$$

From the definition of ρ we see that it is a sub-Gaussian random variable with mean $\mu_\rho = -[\mu_r(t_k^\dagger)]_{s^*}$ and parameter $r_\rho = r_r(t_k^\dagger) + r_a(t_k^\dagger)$, hence we get

$$\mathbb{E}[\exp(s(\rho - \mu_\rho))] \leq \exp\left(s^2[r_r(t_k^\dagger) + r_a(t_k^\dagger)]/2\right), \tag{15}$$

which leads us to

$$P_{MD}(k) \leq \frac{\exp\left(s^2[r_r(t_k^\dagger) + r_a(t_k^\dagger)]/2\right)}{\exp(s(-\delta_u\sqrt{N} + |\mu_\rho|))}. \tag{16}$$

Finding the maximal point at $s' = \frac{-\delta_u\sqrt{N}+|\mu_\rho|}{r_r(t_k^\dagger)+r_a(t_k^\dagger)}$, $s' > 0$, gives

$$P_{MD}(k) \leq \exp\left(-\frac{\max\{0, -\delta_u\sqrt{N} + [\mu_r(t_k^\dagger)]_{s^*}\}^2}{2[r_r(t_k^\dagger) + r_a(t_k^\dagger)]}\right), \tag{17}$$

which result in a probability of Detection converging to 1 as k grows

$$P_D(k) \geq 1 - \exp\left(-\frac{\max\{0, -\delta_u\sqrt{N} + [\mu_r(t_k^\dagger)]_{s^*}\}^2}{2[r_r(t_k^\dagger) + r_a(t_k^\dagger)]}\right) \xrightarrow[k\to\infty]{} 1. \tag{18}$$

Lemma 2 *Assume that the majority of agents are trustworthy. Furthermore, assume that data is sub-Gaussian and i.i.d. between agents and classes. There are values δ_u and ΔT for which $P_{FA}(I_k) < \frac{1}{2} < P_D(I_k)$ when detection is based on consecutive ΔT model updates, where P_{FA} denotes the probability of a false alarm and P_D denotes the probability of detection.*

The proposed detector facilitates continuous operation, unaffected by the convergence time of the joint model, drawing on the insights from the lemma. Let d_k denote the outcome of applying (3) on interval I_k. This results in a sequence of decisions $d_1, d_2, \ldots$, where $d_i = 1$ if the examined agent crosses the threshold, and $d_i = 0$ otherwise. At the conclusion of $K\Delta T$ updates, the coordinating agent assesses each edge agent's behavior. If an edge agent's average decision score over these K intervals, calculated as

$$\frac{1}{K}\sum_{k=1}^{K} d_k, \tag{19}$$

exceeds $1/2$, the agent's input is excluded for the next segment I_{K+1}. Nonetheless, the coordinating agent continues to compute the statistics (3) during this period. Furthermore, if

$$\frac{1}{K}\sum_{k=1}^{K} d_k < \frac{1}{2}, \tag{20}$$

the agent is added back to the list of trustworthy agents. The validity of this approach is encapsulated in the following lemma:

Lemma 3 *Assume we set δ_u, ΔT such that for each k, $P_{FA}(I_k) < 1/2 < P_D(I_k)$. Then, with probability 1, there exists a sufficiently large k_0 such that the presented scheme (cf. (3)–(20)) ignores all the malicious agents after time $k_0\Delta T$, while ensuring that updates from all trustworthy agents are incorporated beyond this time.*

The proof of Lemma 3 is based on a sequence of decisions where for each interval I_k of length ΔT the detector is applied to obtain a decision d_k. Then a majority among all prior decisions is used to decide whether to disconnect the agent. By the assumption $P_{FA}(I_k) < \frac{1}{2} < P_D(I_k)$ and the Borel Cantelli lemma the proof follows.

Proof (Lemma 3) *Let $d_1, d_2, \ldots$ be a sequence of decisions regarding a specific agent, using the detector in (3) based on intervals of length ΔT where ΔT is selected such that $P_{FA}(I_k) < \frac{1}{2} < P_D(I_k)$.*

Let $D_K = \sum_{k=1}^{K} d_k$. We block an agent whenever $D_k > \frac{K}{2}$. Note that by the selection of ΔT and the fact that the intervals are mutually disjoint, we have that for all K $P(D_K < K/2) \leq P(X < K/2)$, where X is a binomial random variable with K trials and probability $p < \frac{1}{2}$ if the agent is malicious. Similarly, if the agent is trustworthy there is such a binomial random variable Y with $p < \frac{1}{2}$ such that $P(D_K > K/2) \leq P(Y > K/2)$.

Using Chernoff's inequality for X, Y, we can bound from below the probability that a trustworthy agent was blocked from the network, and the probability that an attacker remained inside the network during the interval I_k.

Since $X \sim B(K, p)$,

$$Pr(X = k) = \binom{K}{k} p^k q^{K-k}, \quad q = 1 - p, \tag{21}$$

Using Chernoff's inequality we obtain the following:

$$Pr(X < K/2) \leq [4p(1-p)]^{K/2} \tag{22}$$

$$Pr(Y > K/2) \leq [4p(1-p)]^{K/2} \tag{23}$$

Note that for $p \neq 1/2$, this probability is smaller then 1 and decreases exponentially with K.

Let A_K be the event

$$A_K = \left\{\frac{1}{K}\sum_{k=1}^{K} d_k > \frac{1}{2}\middle| \textit{the agent is trustworthy}\right\}. \tag{24}$$

By (23) $\sum_{K=1}^{\infty} Pr(A_K) < \infty$. Hence by the Borel-Cantelli Lemma, the probability that only finitely many A_K occur has probability 1, i.e,

$$Pr\left(\limsup_{K\to\infty} A_K\right) = 0. \tag{25}$$

This implies that each trustworthy agent is removed only finitely many times. A similar argument yields that with probability 1, each malicious agent is included in the computation only finitely many times

$$B_K = \left\{\frac{1}{K}\sum_{k=1}^{K} d_k < \frac{1}{2}\middle| \textit{the agent is malicious}\right\}. \tag{26}$$

Since we have a finite network we get that the result holds for all the agents. The total number of errors is finite with probability 1. This proves the lemma.

4. NUMERICAL SIMULATIONS

In this section, we evaluate the effectiveness of our detection algorithm through simulated attacks on federated learning systems, utilizing the MNIST dataset, a cropped version of the 'LISA Traffic

Light' dataset [26–28] and the MIT-BIH Arrhythmia dataset for this purpose. The MNIST dataset comprises images of handwritten digits $0, \ldots, 9$, with the objective being to accurately classify each digit. The MNIST dataset includes $60,000$ pictures used for training and $10,000$ pictures used for model validation. The LISA dataset was collected in San Diego, CA, USA. It provides 17 daytime and 7 nighttime recordings. We are using a cropped version of this dataset, where an image of each traffic light was extracted from the video and labeled. There are in total $36,534$ traffic light pictures, with the following labels: 'go', 'stop', 'warning', 'stopLeft', 'goLeft', 'goForward', and 'warningLeft'. The dataset is divided into a training set of $32,797$ images and a validation set of $3,737$ images. As the most common labels in the dataset were 'go', 'stop' and 'warning' (comprising almost 90% of the dataset), we have reduced the number of labels from 7 to 3, marking 'goLeft' and 'goForward' as 'go', 'stopLeft' as 'stop', and 'warningLeft' as 'warning'. Examples of the cropped traffic lights pictures can be seen in Figure 2. The MIT-BIH Arrhythmia dataset [29, 30] is a sample set of ECG strips, derived from over 4000 long-term Holter recordings (48 subjects aged 23 to 89) that were obtained by the Beth Israel Hospital Arrhythmia Laboratory between 1975 and 1979. The MIT-BIH includes 17 labels including 'Supraventricular tachyarrhythmia' (SVTA), 'Idioventricular rhythm' (IVT), and many more. An example taken from the MIT-BIH dataset can be seen in Figure 3.

To gauge performance, we focus on the classification error of the learned model as our primary metric of interest. Consequently, we evaluate the algorithm's efficacy based on the average error rate of the learned model, rather than quantifying detection probabilities for individual agents.

In each example, we performed 100 random cross-validation experiments. In the MNIST examples, each experiment involved 60,000 images to train the agents and 10,000 images for testing their performance, where each agent received a random set of $60,000/N$ different images for training. In the LISA Traffic Light dataset examples, each experiment involved 32,797 images to train the agents and 3,737 images for testing their performance, where each agent received a random set of $32,797/N$ different images for training. In the 'MIT-BIH' dataset examples, each experiment involved 800 records to train the agents and 200 records for testing their performance, where each agent received a random set of $800/N$ different records for training.

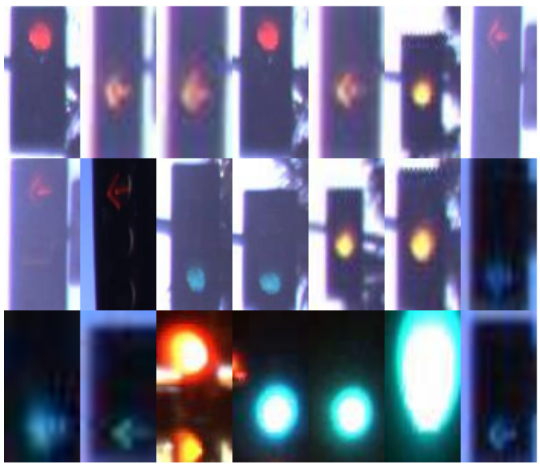
The division of training data between agents ensures that each agent's data is distinct from the data of the other agents in the training phase. In the MNIST example, the dataset is divided between the agents in a non-iid manner (with overlap) such that no agent receives images of all the available digits. In the MIT-BIH ECG dataset, the dataset is also divided in a non-iid manner where each agent received data from different patients. Note that although the data was collected from different patients the labels (classification) are overlapping as different patients had the same classification result.

In the MNIST dataset example, the number of participating trustworthy agents included $N = 3, 5, 10$ with an additional agent participating as an attacker. In the LISA Traffic Light and the MIT-BIH datasets examples, the number of participating agents is $N = 5$, with one designated as an attacker. The attacker's mixing weight was $g(t) = 1/\sqrt{t+1}$ and the integration time was $\Delta T = 5$.

For the MNIST and LISA Traffic Light datasets examples we implemented a 7-layer convolutional neural network (CNN):

(a) An example of a 'go' traffic light taken from the video.

(b) An example of a 'stop' traffic light taken from the video.



(c) Examples of cropped traffic lights taken from the dataset.

Fig. 2: Examples of traffic lights taken from the LISA Traffic Lights dataset video and the cropped dataset [26].

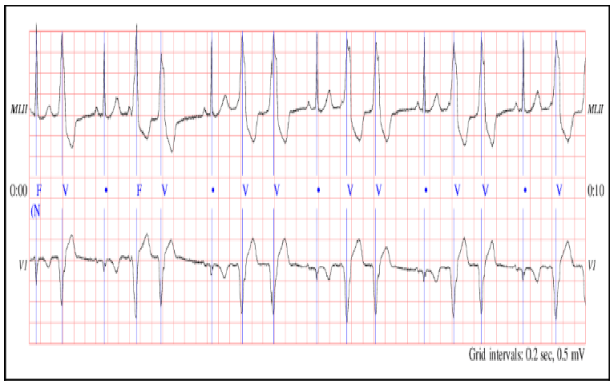


Fig. 3: Example (10s) of annotations in MIT-BIH database [29].

Layer	Size	Input	Output
Conv+ReLU	Cx32x3x1	Cx28x28	32x26x26
Conv+ReLU	32x64x3x1	32x26x26	64x24x24
MaxPool2D	2x2	64x24x24	64x12x12
Dropout + Flatten	$p = 0.25$	64x12x12	1x9216
FC + ReLU	9216x128	1x9216	1x128
Dropout	$p = 0.5$	1x128	1x128
FC	128x10	1x128	1x10

where a 'Conv' layer size is $(CH_{in}, CH_{out}, kernel, stride)$, and C is the amount of color channels in each dataset, $C = 1$ for the MNIST dataset and $C = 3$ for the LISA Traffic Light dataset.

For the MIT-BIH dataset example, we are using a 17-layer CNN suggested in [31]:

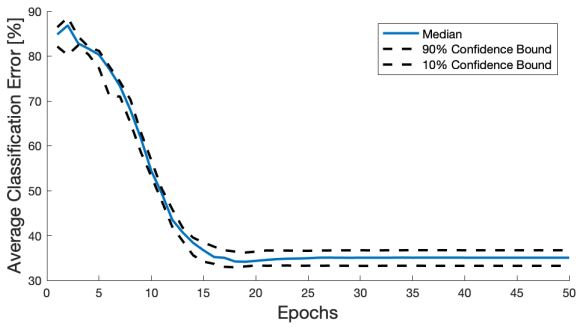
Layer	Size	Input	Output
Conv+ReLU	1x8x16x2 (Pad 7)	1x3600x1	8x1800x1
MaxPool1D	8 (Stride 4)	8x1800x1	8x449x1
Conv+ReLU	8x12x12x2 (Pad 5)	8x449x1	12x224x1
MaxPool1D	4 (Stride 2)	12x224x1	12x111x1
Conv+ReLU	12x32x9x1 (Pad 4)	12x111x1	32x111x1
MaxPool1D	5 (Stride 2)	32x111x1	32x54x1
Conv+ReLU	32x64x7x1 (Pad 3)	32x54x1	64x54x1
MaxPool1D	4 (Stride 2)	64x54x1	64x26x1
Conv+ReLU	64x64x5x1 (Pad 2)	64x26x1	64x26x1
MaxPool1D	2 (Stride 2)	64x26x1	64x13x1
Conv+ReLU	64x64x3x1 (Pad 1)	64x13x1	64x13x1
MaxPool1D	2 (Stride 2)	64x13x1	64x6x1
Conv+ReLU	64x72x3x1 (Pad 1)	64x6x1	72x6x1
MaxPool1D	2 (Stride 2)	72x6x1	72x3x1
Flatten	72x3	72x3x1	1x216
FC + ReLU	216x64	1x216	1x64
Dropout	$p = 0.1$	1x64	1x64
FC	64x17	1x64	1x17

4.1. Example I—MNIST Non-IID Constant-Output Attack

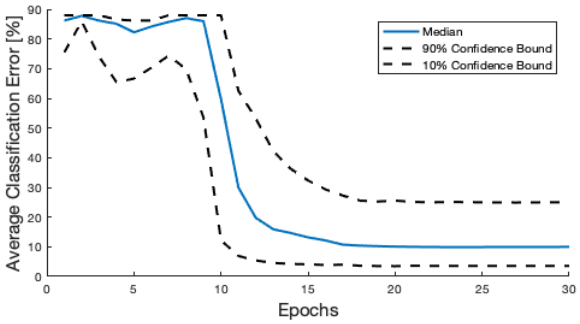
In this example, the agents are attempting to learn a classifier for the MNIST dataset, while the attacker (w.l.o.g marked as agent 0) is aiming to inject a model that consistently outputs the digit 9. In this example we have 5 trustworthy agents in the network, the dataset is divided between the agents in a non-iid manner (with overlap) such that each agent $1, 2, 3, 4, 5$ receives pictures of the following digits respectively $\{0-3\}, \{2-5\}, \{4-7\}, \{6-9\}, \{8, 9, 0, 1\}$.

Figure 4 displays the accumulated statistics over 100 experiments, including the 10% and 90% confidence bounds. Figure 4(a) shows the statistics of 100 trustworthy experiments with no attackers present in the network, and Figure 4(b) illustrates the statistics of 100 experiments of an attacked network with detection.

Figure 5 depicts the mitigation scheme ROC for different numbers of trustworthy agents, $N = 3, 5, 10$, in the network. We can see that the higher portion the attacker holds in the network, it becomes harder to detect it.



(a) Trustworthy network with no attackers (100 Experiments).



(b) Network Resilience With Detection (100 Experiments).

Fig. 4: Comparative Analysis of Constant-Output Attack on the Non-IID MNIST dataset: Showcasing 100 experiments on a $N = 5$ agent network, this figure highlights the model's classification without attackers present (a) and with the detection scheme where attackers are present (b). Included are 10% and 90% confidence bounds, underscoring the attack's effect and the detection's efficacy.

4.2. Example II—Traffic Lights-Constant-Output Attack

In this example, the agents are attempting to learn a classifier for the cropped LISA dataset, while the attacker is aiming to inject a model that consistently outputs a 'stop'. Figure 6 displays the accumulated statistics over 100 experiments, including the 10% and 90% confidence bounds. Figure 6(a) shows the statistics of 100 experiments of an attacked network without detection, and Figure 6(b) illustrates the statistics of 100 experiments of an attacked network with detection.

Figure 9 depicts an illustration of the Confusion-Matrix without the detection scheme (a) and with the detection scheme (b). It is very clear that the Constant-Output attack is very powerful yet the mitigation scheme is able to detect it.

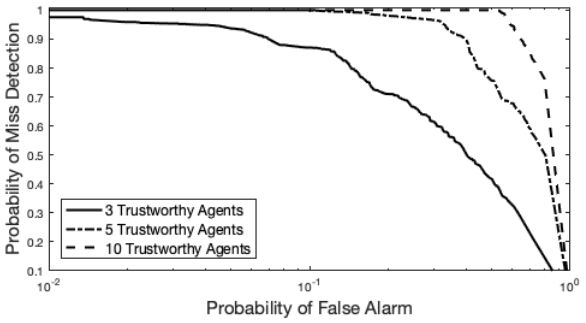
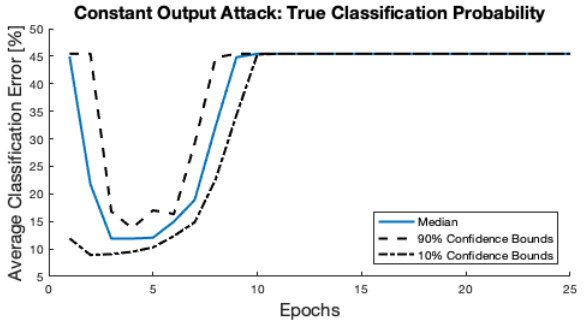
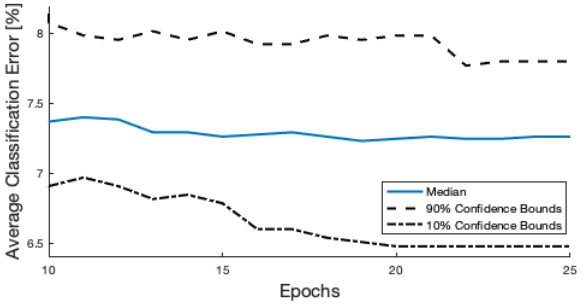


Fig. 5: ROCs temporal difference detection performance of the attacker applying Constant-Output Attack on the Non-IID MNIST dataset, for different network sizes, varying between $N = 3, 5, 10$.

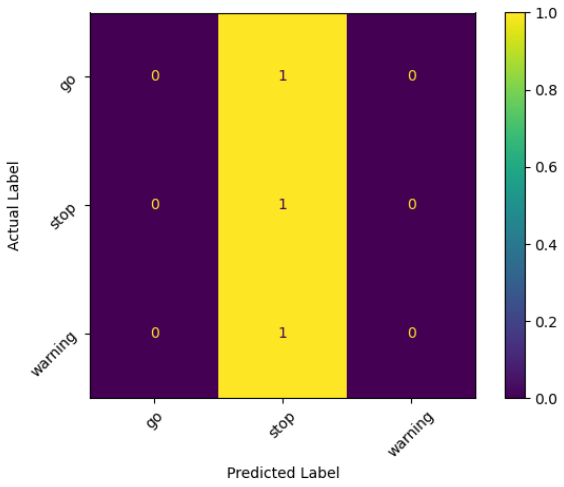


(a) Attack Impact Without Detection (100 Experiments).

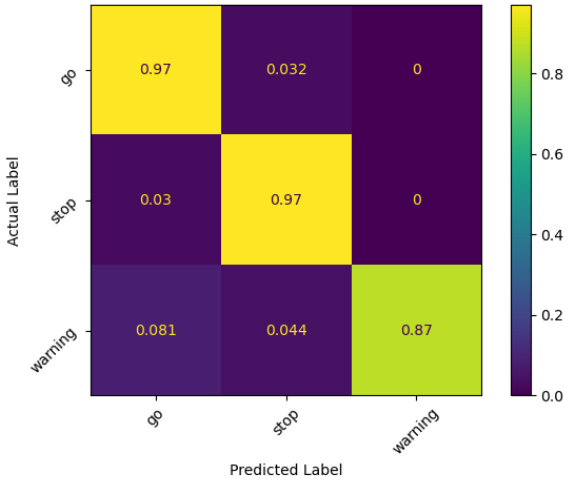


(b) Network Resilience With Detection (100 Experiments).

Fig. 6: Comparative Analysis of Constant-Output Attack on the Traffic-Lights dataset: Showcasing 100 experiments on a $N = 5$ agent network, this figure highlights the model's classification error with and without the detection scheme [(a) and (b)]. Included are 10% and 90% confidence bounds, underscoring the attack's effect and the detection's efficacy. Note that approximately 58% of the Traffic-Lights dataset consists of 'stop' picture, i.e. a successfull Constant 'stop' attack will result in $\sim 42\%$ error rate.

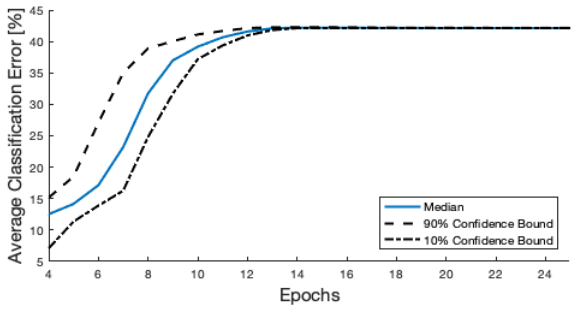


(a) Attack Impact Without Detection.

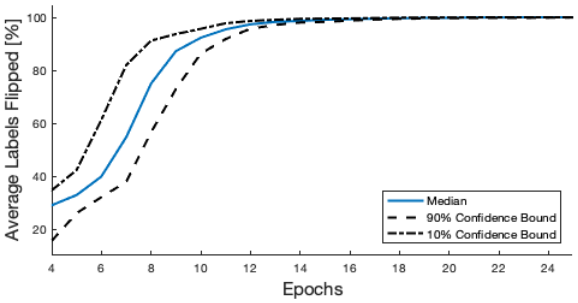


(b) Network Resilience With Detection.

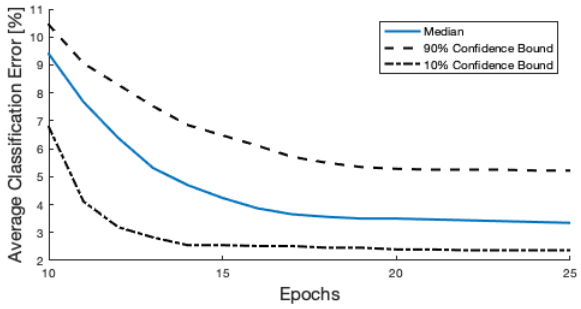
Fig. 7: Confusion Matrix for the Constant-Output Attack on the Traffic-Lights dataset on a $N = 5$ agent network. This figure examples the model's classification performance with and without the detection scheme [(a) and (b)].



(a) Classification Error Without Detection (100 Experiments).

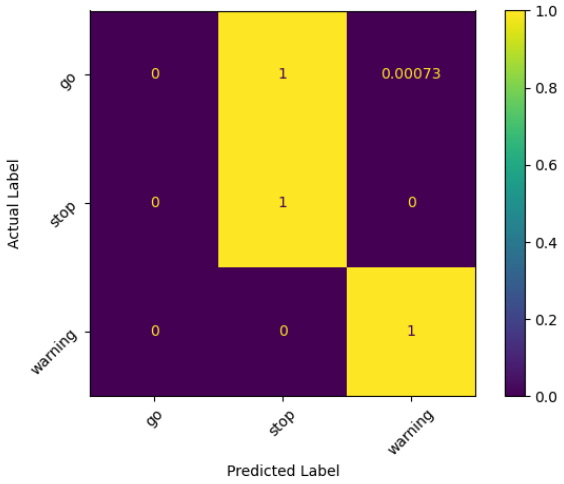


(b) Label-Flip Success Rate Without Detection (100 Experiments).

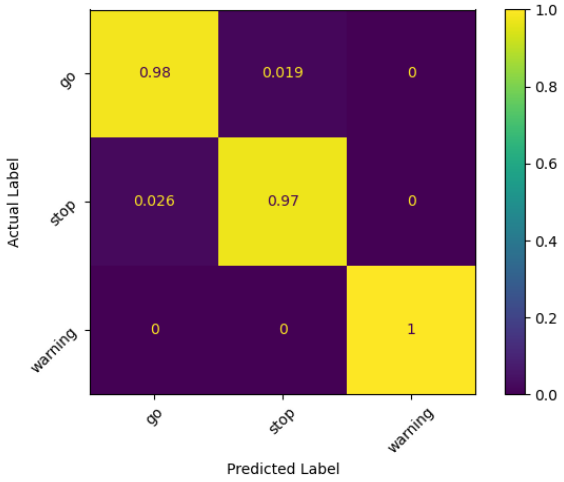


(c) Classification Error with Detection (100 Experiments).

Fig. 8: Evaluating Label-Flipping Attack Impact on the Traffic-Lights dataset: Plots (a) and (b) present the outcomes of 100 experiments without detection, showing classification error and label-flip success rate, respectively. (c) Classification error with detection implemented, offering a comparative perspective. All plots include 10% and 90% confidence bounds.



(a) Attack Impact Without Detection.



(b) Network Resilience With Detection.

Fig. 9: Confusion Matrix for the Label-Flip Attack (Flip 'go' to 'stop') on the Traffic-Lights dataset on a $N = 5$ agent network. This figure shows the model's classification performance with and without the detection scheme [(a) and (b)].

4.3. Example III—Traffic Lights-Label-Flipping Attack

In this example, the agents are attempting to learn a classifier for the cropped LISA dataset, while the attacker aims to inject a model that flips the 'go' label to 'stop'. Consider a label-flipping function $h(c) : C \to \widetilde{C}$, where C is the set of possible labels and $\widetilde{C} \subseteq C$. For any sample d_i with label c_i, the attacker's model will return $h(c_i)$ instead. The label-flipping function used in this example is described in the following table:

c_i	'go'	'stop'	'warning'
$h(c_i)$	'stop'	'stop'	'warning'

Note that multiple label flipping is also supported.

Figure 8 presents the acquired statistics over 100 experiments, complete with the 10% and 90% confidence bounds. Figure 8(a) displays the average classification error from 100 experiments of an attacked network without detection. Figure 8(b) illustrates the average number of successful label-flippings according to the label-flipping function $h(c_i)$, and Figure 8(c) depicts the average classification error from 100 experiments of an attacked network with detection.

Figure 9 depicts an illustration of the Confusion-Matrix without the detection scheme (a) and with the detection scheme (b). It is very clear that only the 'go' label is affected as stated by the attack description, fully classified as 'stop' without the detection scheme applied and truthfully classified as 'go' with the detection scheme. Note that the 'warning' label isn't affected by that targeted attack at all.

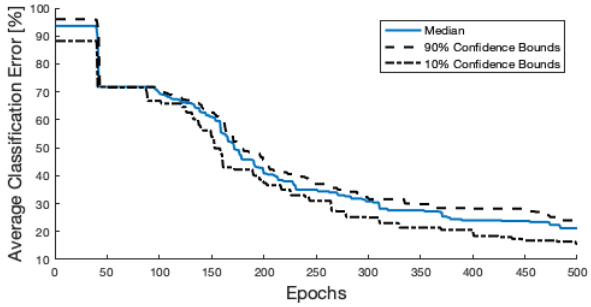
4.4. Example IV—ECG Non-IID Constant-Output Attack

In this example, the agents are attempting to learn a classifier for the ECG MIT-BIH dataset, while the attacker (w.l.o.g marked as agent 0) is aiming to inject a model that consistently outputs 'Supraventricular tachyarrhythmia' (SVTA). In this example we have 4 trustworthy agents in the network, the dataset is divided between the agents in a non-iid manner such that each agent $1, 2, 3, 4$ receives records from 12 different subjects, with no overlap. Note that each record is individually labeled thus each agent may be exposed to all 17 labels.

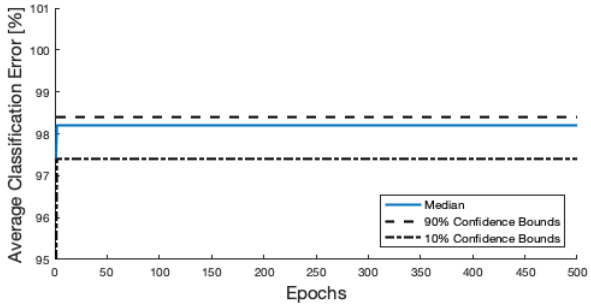
Figure 10 displays the accumulated statistics over 100 experiments, including the 10% and 90% confidence bounds. Figure 10(a) shows the statistics of 100 experiments of an attacked network without detection, and Figure 10(b) illustrates the statistics of 100 experiments of an attacked network with detection. Figure 10(c) illustrates the statistics of 100 experiments of a trustworthy network with no attack, offering a comparative perspective. We can see that the mitigation scheme detects the attack long before it becomes effective, allowing the network to maintain convergence as if it was not attacked at all.

5. CONCLUSIONS

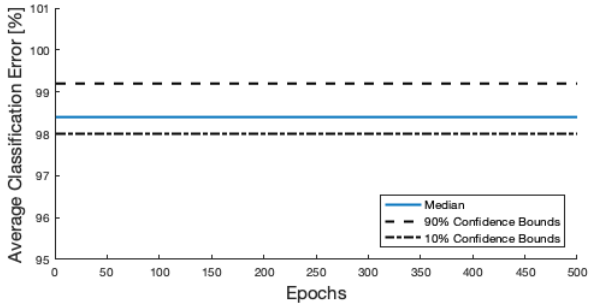
In this paper, we have presented a robust federated learning algorithm that can operate in the presence of data injection attacks. We have provided conditions for the identification of malicious agents. We have also demonstrated the performance of the proposed technique on various attacks. Detailed proofs of the lemmas as well as bounds on the attacker detection probability and the false-alarm probability are to be presented in an extended version of this work.



(a) Classification Error Without Detection (100 Experiments).



(b) Classification Error with Detection (100 Experiments).



(c) Classification Error Without Attack (100 Experiments).

Fig. 10: Evaluating Constant-Output Attack Impact on the ECG MIT-BIH dataset: Plots (a) and (b) present the outcomes of 100 experiments with (a) and without (b) detection on an attacked network, showing classification error. Plot (c) demonstrates the performance of the network without attack, offering a comparative perspective. All plots include 10% and 90% confidence bounds.

6. APPENDIX

6.1. Appendix A.

This appendix states the available labels in the MIT-BIH database [30]. The labels are 1,2,3,4,5,6,7,8,9,... **[OS:** TODO]

7. REFERENCES

[1] W Nicholson Price and I Glenn Cohen, "Privacy in the age of medical big data," *Nature Medicine*, vol. 25, no. 1, pp. 37–43, 2019.

[2] Priyank Jain, Manasi Gyanchandani, and Nilay Khare, "Big data privacy: a technological perspective and review," *Journal of Big Data*, vol. 3, pp. 1–25, 2016.

[3] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, pp. 106775, 2021.

[4] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[5] Yang Li, Xinhao Wei, Yuanzheng Li, Zhaoyang Dong, and Mohammad Shahidehpour, "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4862–4872, 2022.

[6] Liang Zhao, Jiaming Li, Qi Li, and Fangyu Li, "A federated learning framework for detecting false data injection attacks in solar farms," *IEEE Transactions on Power Electronics*, vol. 37, no. 3, pp. 2496–2501, 2021.

[7] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[8] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[9] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[10] Gokberk Yar, Cristina Nita-Rotaru, and Alina Oprea, "Backdoor attacks in peer-to-peer federated learning," *arXiv preprint arXiv:2301.09732*, 2023.

[11] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16937–16947, 2020.

[12] Lingjuan Lyu, Han Yu, and Qiang Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[13] Priyanka Mary Mammen, "Federated learning: Opportunities and challenges," *arXiv preprint arXiv:2101.05428*, 2021.

[14] Sissi Xiaoxiao Wu, Hoi-To Wai, Anna Scaglione, Angelia Nedić, and Amir Leshem, "Data injection attack on decentralized optimization," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 3644–3648.

[15] Nikhil Ravi and Anna Scaglione, "Detection and isolation of adversaries in decentralized optimization for non-strongly convex objectives," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 381–386, 2019.

[16] Or Shalom, Amir Leshem, and Anna Scaglione, "Localization of data injection attacks on distributed m-estimation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 655–669, 2022.

[17] Jian Xu and Shao-Lun Huang, "Byzantine-resilient decentralized collaborative learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 5253–5257.

[18] Jinhyun So, Başak Güler, and A Salman Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[19] Cheng Fang, Zhixiong Yang, and Waheed U. Bajwa, "BRIDGE: Byzantine-resilient decentralized gradient descent," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 610–626, July 2022.

[20] Zhixiong Yang and Waheed U Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019.

[21] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.

[22] Nuria Rodríguez-Barroso, Eugenio Martínez-Cámara, M Victoria Luzón, and Francisco Herrera, "Backdoor attacks-resilient aggregation based on robust filtering of outliers in federated learning for image classification," *Knowledge-Based Systems*, vol. 245, pp. 108588, 2022.

[23] Idan Achituve, Wenbo Wang, Ethan Fetaya, and Amir Leshem, "Communication efficient distributed learning over wireless channels," *arXiv preprint arXiv:2209.01682*, 2022.

[24] Farzin Haddadpour and Mehrdad Mahdavi, "On the convergence of local descent methods in federated learning," *arXiv preprint arXiv:1910.14425*, 2019.

[25] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor, "Fast-convergent federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2020.

[26] ITHB, "Lisa traffic light detection dataset," https://universe.roboflow.com/ithb-5ka4m/lisa-traffic-light-detection-8vuch, Roboflow Universe.

[27] Mark Philip Philipsen, Morten Bornø Jensen, Andreas Møgelmose, Thomas B Moeslund, and Mohan M Trivedi, "Traffic light detection: A learning algorithm and evaluations on challenging dataset," in *intelligent transportation systems (ITSC), 2015 IEEE 18th international conference on*. IEEE, 2015, pp. 2341–2345.

[28] Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, 2016.

[29] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[30] George B Moody and Roger G Mark, "The impact of the mit-bih arrhythmia database," *IEEE engineering in medicine and biology magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[31] Hanshi Sun, Ao Wang, Ninghao Pu, Zhiqing Li, Junguang Huang, Hao Liu, and Zhi Qi, "Arrhythmia classifier using convolutional neural network with adaptive loss-aware multi-bit networks quantization," in *2021 2nd International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 2021, pp. 461–467.