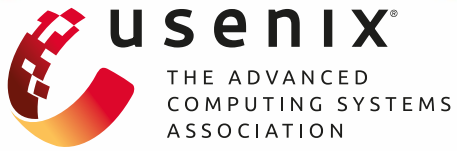

Justinian's GAAvernor: Robust Distributed Learning with Gradient Aggregation Agent

Xudong Pan, Mi Zhang, Duocai Wu, and Qifan Xiao, *Fudan University;*
Shouling Ji, *Zhejiang University/Ant Financial;* Min Yang, *Fudan University*

https://www.usenix.org/conference/usenixsecurity20/presentation/pan

This paper is included in the Proceedings of the
29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the
29th USENIX Security Symposium
is sponsored by USENIX.

Justinian's GAAvernor: Robust Distributed Learning with Gradient Aggregation Agent

Xudong Pan[†], Mi Zhang[†], Duocai Wu[†], Qifan Xiao[†], Shouling Ji[*,‡], and Min Yang[†]

[†]*Fudan University,* [*]*Zhejiang University,* [‡]*Ant Financial*

Emails: {xdpan18, mi_zhang, dcwu18, qfxiao16}@fudan.edu.cn, sji@zju.edu.cn, m_yang@fudan.edu.cn

Abstract

The hidden vulnerability of distributed learning systems against *Byzantine attacks* has been investigated by recent researches and, fortunately, some known defenses showed the ability to mitigate Byzantine attacks when a minority of workers are under adversarial control. Yet, our community still has very little knowledge on how to handle the situations when the proportion of malicious workers is 50% or more. Based on our preliminary study of this open challenge, we find there is more that can be done to restore Byzantine robustness in these more threatening situations, if we better utilize the auxiliary information inside the learning process.

In this paper, we propose *Justinian's GAAvernor* (GAA), a Gradient Aggregation Agent which *learns to be robust* against Byzantine attacks via reinforcement learning techniques. Basically, GAA relies on utilizing the historical interactions with the workers as *experience* and a *quasi-validation set*, a small dataset that consists of less than 10 data samples from similar data domains, to generate reward signals for policy learning. As a complement to existing defenses, our proposed approach does not bound the expected number of malicious workers and is proved to be robust in more challenging scenarios.

Through extensive evaluations on four benchmark systems and against various adversarial settings, our proposed defense shows desirable robustness as if the systems were under no attacks, even in some case when 90% Byzantine workers are controlled by the adversary. Meanwhile, our approach shows a similar level of time efficiency compared with the state-of-the-art defenses. Moreover, GAA provides highly interpretable traces of worker behavior as by-products for further mitigation usages like Byzantine worker detection and behavior pattern analysis.

Justinian I, an emperor of Byzantium, reorganized the imperial government to revive the empire's greatness in a dark time. Gradient Aggregation Agent, a new GAAvernor (pronounced as *governor*) of distributed learning system, bases its learning policy on historical and auxiliary information to fight against Byzantine attacks.

1 Introduction

Over the past few decades, deep learning has achieved abundant breakthroughs driven by big data [38, 52]. To deal with the fast scaling-up of data volume, many efficient distributed learning algorithms have been proposed in the past decade [3, 22, 29], yet their hidden vulnerability to *Byzantine attacks* [37] have also been observed by a series of recent works [11, 16, 31, 62].

In a typical distributed learning system [3, 34, 41, 43, 50, 64], a group of *workers* participate in building a global learning model under the coordination of one parameter server. In each round, the server first distributes current parameters of the global learning model to each worker, requiring them to compute the corresponding gradient based on their local data. Once receiving all the submissions from the workers, the server then applies certain *Gradient Aggregation Rule* (GAR) to yield the next weight update. As an optimal choice in theory [12, 47], most existing distributed learning algorithms implemented their GAR simply by averaging over the whole set of submitted gradients [42, 56, 63].

However, the behaviors of real-world workers are far from ideal. As is suggested in [62], a worker may probably submit abnormal gradients due to various causes such as biased batch sampling, computation error, network instability or even malicious attacks. In [11], a worker with the aforementioned abnormal behavior is usually referred to as a *Byzantine worker*. As first observed by Blanchard et al., the classical GAR (i.e., GAR by averaging) is so fragile that even a single Byzantine worker can have a catastrophic effect on the whole learning process, from degraded prediction accuracy [31] to total stagnation [11]. These facts highly emphasize the urgency and significance of effective defense against this type of adversarial behavior, namely *Byzantine attack*.

To fight against Byzantine attacks, most previous studies implement alternative GARs to the classical one [4, 11, 16, 31, 62]. These methods view gradients abstractly as high-dimensional vectors to apply robust statistical methods such as clustering [11], median [31] or geometric median [4, 16, 62].

Although it allows previous methods to be highly decoupled with the underlying learning systems, the simplicity is accompanied with several weaknesses: First, as previous GARs computes the weight update direction as the only product, they are unable to provide interpretable information of the workers' behaviors for further mitigation; Second, due to the theoretical bottleneck of robust statistics [48], most known defenses expect that only a minority of workers are compromised. As a result, they are inadequate and cannot be directly extended to cover more challenging scenarios where the adversary has gained control over a majority of workers and iteratively manipulates an uncertain ratio of workers to play the Byzantine roles.

Our Work. In this paper, we propose the design of *Justinian's GAAvernor* (GAA), a Gradient Aggregation Agent which serves as a novel server-side defense that leverages Reinforcement Learning (RL) techniques to *learn to be Byzantine-robust* from interactions with the workers and from the auxiliary information on the server. Our defense aims at restoring the robustness of distributed learning in more challenging scenarios characterized by the existence of the malicious majority.

By viewing the historical interactions with the workers as its *experience* and the relative decrease of loss on *a quasi-validation set* as its *reward*, GAA searches over a *simplex* as its *policy space* for the optimal policy. Intuitively, each coordinate of a policy of GAA can be interpreted as its current *credit* on the corresponding worker. By proposing the weight update at each iteration as a linear combination of the received gradients weighted with its credits, GAA receives the reward signal after the global learning model is updated with the current weight update and it then optimizes its current policy by RL techniques [54]. It is worth to notice, we introduce the notion of a *quasi-validation set* to denote a collection of data samples that follows a similar but not necessarily identical distribution as the true sample distribution. In practice, when a golden-labeled validation set (i.e., a set of samples from the true sample distribution) is available during the learning process, GAA can utilize it as its quasi-validation set. Otherwise, GAA randomly collects a small number of data samples (empirically, less than 10 samples) from similar data domains to form its quasi-validation set.

With extensive experiments, we evaluate GAA's robustness on four diverse case studies (i.e., MNIST [39], CIFAR-10 [35], Yelp reviews [1] and CMS public healthcare records [2]), against various attacking settings. We find our proposed approach shows near-optimal Byzantine robustness in most cases, whenever the ratio of Byzantine workers (i.e., *Byzantine ratio*) is below or over 50% or fluctuates unboundedly. Meanwhile, GAA shows comparable time efficiency to known defenses. We also evaluate GAA's robustness against several adaptive attacks on this novel defense mechanism. Moreover, we present the application of GAA to Byzantine worker detection, which shows high accuracy, and to behavior pattern analysis of Byzantine attacks, which demonstrates high interpretability of its traces.

Contributions. In summary, we mainly make the following contributions.

- We propose the design of GAA, a novel RL-based defense against Byzantine attacks which requires no upper bound on the Byzantine ratio (§4).
- We implement and evaluate our proposed defense on four diverse case studies, against various adversarial settings. Empirical results suggest in most cases, GAA with an easily accessible quasi-validation set helps the distributed learning systems achieve almost indistinguishable performance as if the systems were under no attacks (§5 & §6).
- We also provide a number of analytic results on GAA's robustness in different settings as theoretical evidences (§4.4).
- Additionally, we demonstrate the interpretability of GAA's traces with visualizations and with applications to Byzantine worker detection and behavior analysis (§4.5), which we hope will facilitate future mitigation studies.

2 Background and Preliminaries

Gradient-based Distributed Learning and GAR. In this paper, we focus on the data-parallel distributed learning system with one parameter server (abbrev. the server) and n workers. This system model is widely used as one of the commonest implementations of distributed learning algorithms [3, 34, 41, 43, 50, 64]. We denote the loss function to be minimized as $f(\theta, \mathcal{D})$, where $\theta \in \mathbb{R}^d$ collects all the free parameters of the underlying model (e.g., a deep neural network) and $\mathcal{D}$ denotes the sample distribution. Usually, the true loss function $f(\theta, \mathcal{D})$ is the expectation over the sample distribution, i.e. $f(\theta, \mathcal{D}) := \mathbb{E}_{z\sim\mathcal{D}}[f(\theta, z)]$ where $\mathcal{D}$ is unknown to the server. In practice, the optimization happens on the empirical version of the loss $f(\theta, D) := \frac{1}{|D|}\sum_{z\in D} f(\theta, z)$, where D is a collection of training samples. For simplicity, we denote the true loss function as f and the empirical loss function calcuated on dataset D as $\hat{f}_D$.

The distributed learning process starts with an initial guess θ_0 on parameters. At iteration t, the server first sends the current parameter θ_t to each worker. Ideally, a worker i then computes the estimated gradient V_i^t of loss f at parameter θ_t based on its local data and submits V_i^t back to the server. Once the server receives the candidate set of gradients $Q_t := \{V_1^t, \ldots, V_n^t\}$, it executes certain GAR $\mathcal{F} : (\mathbb{R}^d)^n \to \mathbb{R}^d$ to aggregate the received gradients into a single weight update direction. Such a procedure is executed in iterations until a provided termination condition is reached. Formally, the update rule at iteration t follows $\theta_{t+1} = \theta_t - \lambda\mathcal{F}(V_1^t, \ldots, V_n^t)$, where λ is the learning rate.

In the literature of distributed learning, the following GARs are the common choices for implementation of $\mathcal{F}$ [3, 22, 29, 34, 61], while their vulnerability to Byzantine attacks have

been studied in a series of recent works [11, 16, 31, 62].

Definition 1 (Classical GAR). $\mathcal{F}(V_1,\ldots,V_n) = \frac{1}{n}\sum_{i=1}^{n} V_i$

Definition 2 (Linear GAR). *As a generalization of classical GAR, a linear GAR $\mathcal{F}$ with parameter $\alpha \in \mathbb{S}^n$ is defined as $\mathcal{F}(V_1,\ldots,V_n) = \sum_{i=1}^{n} \alpha^i V_i$, where $\mathbb{S}^n := \{\alpha \in \mathbb{R}^n : \alpha^i \geq 0, \sum_{i=1}^{n} \alpha^i = 1\}$ is called* an n-dimension simplex.

Benign Workers vs. Byzantine Workers. In order to have a precise understanding of what a Byzantine worker is, we start from a formal definition of *benign worker*.

As is discussed, at iteration t, each worker is expected to estimate the true gradient $g_t = \mathbb{E}_z[\nabla_\theta f(\theta_t, z)]$ based on its local data set D. Optimally, it computes $V^t := \frac{1}{|D|}\sum_{z\in D} \nabla_\theta f(\theta_t, z)$ as its submission, due to the well-known fact that V^t is *an unbiased estimator* of g_t if D is i.i.d. sampled from $\mathcal{D}$ [12]. Generally, it inspires us to make the following definition.

Definition 3 (Benign Worker). *A worker which submits a gradient V^t at iteration t is said to be benign if V^t is an unbiased estimator of the true gradient g_t, i.e., $\mathbb{E}V^t = g_t$.*

With such a definition of benign worker, it is rather simple to define a Byzantine worker as its opposition.

Definition 4 (Byzantine Worker). *Otherwise, a worker is said to be Byzantine at iteration t if V^t is biased, i.e., $\mathbb{E}V^t - g_t \neq 0$.*

A well-established theorem from statistics states that classical SGD is guaranteed to converge if the gradient estimation at each descent step is unbiased [12, 14]. If the system is ideally correct, classical GAR is almost the optimal choice. However, it is usually not the case in real-world settings [62]. In fact, as first noticed by [11], *classical GAR and its variants are so fragile that even a single Byzantine worker can totally break the whole learning process*, as is stated by the following lemma.

Proposition 1. *[11, Lemma 1] For any linear GAR $\mathcal{F}$ with fixed parameter α, the adversary with only one single Byzantine worker can fool $\mathcal{F}$ into yielding any arbitrary weight update continually regardless of other submissions.*

3 Security Settings

3.1 Threat Model

Throughout this paper, we consider the same threat model as in previous studies [4, 11, 16, 31, 62]. Generally speaking, this threat model assumes that, the adversary compromises a proportion β (s.t. $\beta \in (0,1)$) of all workers throughout the learning process and he/she commands the compromised workers to present arbitrary behaviors at each iteration. In other words, the adversary is able to choose the submitted gradients of each manipulated worker. Noteworthily, at iteration t, the Byzantine ratio can be also smaller than β if some Byzantine workers pretend benign. To provide a finer-grained description on the threat model, we introduce the following notions.

Table 1: Comparisons among different defenses against Byzantine attacks.

	Constraint	Time Complexity	Space Complexity
Brute-Force [31, 48]	$n \geq 2m+1$	$O(\binom{n}{m}(n-m)d)$	$O(\binom{n}{m}+nd)$
GeoMed [16, 62]	$n \geq 2m+1$	$O(n^2 d)$	$O(n^2 d)$
Krum [11]	$n \geq 2m+3$	$O(n^2 d + n^2 \log n)$	$O(n^2 d)$
Bulyan [31]	$n \geq 4m+3$	$O(n^2 d)$	$O(n^2 + nd)$
GAA (ours)	$n \geq m+1$	$O(n^3 d)$	$O(n^2 + nd)$

Role Function. As is discussed, each worker behaves either benignly or maliciously at iteration t. Therefore, we introduce the notion of the *role function* of worker i to characterize its temporal behaviors. Formally, the role function is defined as a binary-valued function on $\mathbb{Z}_+$, i.e., the timeline. Intuitively, $r_i(t) = 1$ means worker i behaves normally at iteration t and otherwise, worker i is a Byzantine worker.

Tampering Algorithm. Byzantine workers can choose different tampering algorithms to produce malicious gradients. In previous studies, several realizations of tampering algorithms have been used for evaluation of defenses, such as *random fault* [11] (More details can be found in Section 5.1). In general, we denote the tampering algorithm as $\mathcal{T}$, which, with the estimated gradient as the input, outputs the tampered gradient for submission. As in previous studies, we assume the identity of the tampering algorithm for each malicious worker.

With the notions above, the behavior of the manipulated worker i at iteration t can be described as

1. First, the adversary selects the current role of the worker i as $r_i(t)$.
2. If the role is benign, i.e., $r_i(t) = 1$, then the worker honestly computes the gradient on its local data, that is, V_i^t.
3. Otherwise, i.e., $r_i(t) = 0$, it tampers the gradient V_i^t with certain tampering algorithm $\mathcal{T}$ (e.g., random fault) and produces $\mathcal{T}(V_i^t)$.
4. Finally, the produced gradient is sent back to the server.

3.2 Previous Defenses

In order to fight against the aforementioned threat model, previous works proposed several alternative GARs to classical GAR and its linear variants. We briefly review the state-of-the-art defenses as follows, where m out of n workers are assumed to be Byzantine at certain iteration, s.t. $m/n \leq \beta$. For an overview, please refer to Table 1.

Brute-Force [31, 48] is based on a brute-force search for an optimal subset $\mathcal{C}^*$ in $\mathcal{Q}$ of size $n-m$ with the minimal maximum pairwise distance. Formally, the optimal set can be written as $\mathcal{C}^* = \arg\min_{\mathcal{C}\in\mathcal{R}} \max_{(V_i,V_j)\in\mathcal{C}\times\mathcal{C}} \|V_i - V_j\|$, where $\mathcal{R} := \{\mathcal{C} \subset \mathcal{Q} : |\mathcal{C}| = n-m\}$. Then the proposed weight up-

date direction is calculated as $\mathcal{F}(V_1,\ldots,V_n)=\frac{1}{n-m}\sum_{V\in\mathcal{C}^*}V$. It was proved to be perfectly robust when $n\geq 2m+1$ [48], while it is almost intractable in highly distributed learning systems.

GeoMed [16, 62] computes the geometric median of $\mathcal{Q}$ as the proposed estimator, which assumes the Byzantine ratio satisfies $n\geq 2m+1$ [16, 62]. In consideration of the computational complexity of geometric median when n is large [18], recent works on Byzantine robustness proposed to approximate it with the vector in $\mathcal{Q}$ which has the smallest sum of distance with other gradients, i.e., $\mathcal{F}(V_1,\ldots,V_n):=\arg\min_{V_i}\sum_{j\neq i}\|V_i-V_j\|$.

Krum [11] was recently proposed in [11] as an approximate algorithm to Brute GAR, which assumes the Byzantine ratio satisfies $n\geq 2m+3$. It first finds the $n-m-2$ closest vectors in $\mathcal{Q}$ for each V_i, which is denoted as $i\rightarrow j$ in their original work. Next, it computes a score for each vector V_i with the formula $s(V_i)=\sum_{i\rightarrow j}\|V_i-V_j\|^2$. Finally, it proposes the vector V_i with the smallest score as the next update step, i.e., $\mathcal{F}(V_1,\ldots,V_n)=\arg\min_{V_i\in\mathcal{Q}}s(V_i)$.

Bulyan [31] was originally designed for Byzantine attacks that concentrate on a single coordinate. First, it runs Krum over $\mathcal{Q}$ without replacement for $n-2m$ time and collect the $n-2m$ gradients to form a *selection set*. It then computes $\mathcal{F}$ coordinate-wise: the i-th coordinate of $\mathcal{F}$ is equal to the average of the $n-4m$ closest i-th coordinates to the median i-th coordinate of the selection set. Bulyan has the strictest assumption as $n\geq 4m+3$ (and otherwise it is not executable), which significantly limits its practical usage.

As we can see, the aforementioned approaches only considered the limited situation when β is expected to be smaller than $1/2$. In more general cases, e.g., when there is no explicit upper bound on the Byzantine ratio in the system, merely no defenses above could remain robust any longer. The following proposition provides a typical failure case.

Proposition 2. *Consider the submitted gradients at iteration t as $(V_1,\ldots,V_{n-m},B_1,\ldots,B_m)$ where $\{B_i\}_{i=1}^m$ are Byzantine gradients. For the slightest violations in each case, i.e., $n=2m$ for Brute GAR, GeoMed and $n=2m+2$ for Krum, the adversary can simply take $B_1=B_2=\ldots=B_m=E$ to tempt these GARs to always yield E, any arbitrary direction specified by the adversary.*

In practice, this more challenging situation could happen for distributed learning systems in open network environments [61]. When the adversary has already compromised a majority of workers at the beginning or continuously gains malicious control over each worker during the learning process, the Byzantine ratio in system could go over $1/2$ or even fluctuate with uncertainty. In either cases, the system robustness is no longer under guard with the above defenses.

4 Defense with Gradient Aggregation Agent

4.1 Overview

In order to restore robustness in a more general scenario, we suggest the defender to be combined more tightly with the underlying learning process, by utilizing some auxiliary information inside the distributed learning system for mitigation purposes. Before providing an overview of our methodology, we first clarify our security assumptions and present our goals of defense.

4.1.1 Security Assumptions. We make the following assumptions on the distributed learning system where GAA is to be deployed.

Assumption 1. The server is secure.

Assumption 2. There is one worker that is never controlled by the adversary.

Assumption 3. The local datasets on workers are i.i.d. sampled from the unknown distribution $\mathcal{D}$.

Assumption 4. GAA has access to a *quasi-validation set* B of size S, which consists of i.i.d. samples from a sample distribution P_m s.t. $\mathrm{KL}(P_m||\mathcal{D})<\infty$, i.e., the KL-divergence between P_m and D is upper bounded by a constant.

Here, Assumptions 1 & 3 are commonly adopted in previous studies [4, 11, 16, 31, 62]. As GAA is deployed on the server, Assumption 1 guarantees its correct execution. Noticeably, Assumption 2 relaxes the known slightest requirements on the tolerable Byzantine ratio to $1-1/n$. As a trade-off, we require Assumption 4 to introduce an additional condition on the availability of a quasi-validation set that follows a similar but not necessarily identical distribution as the true sample distribution. In theory we prove the lower the divergence, the better the model performance will be (Thm. 1 & 2). Through empirical evidences, we show this assumption can be easily satisfied with the quasi-validation set that consists of few samples from similar data domains, if there is no provided golden validation set [34, 61].

4.1.2 Defender's Goals. Towards Byzantine robustness, the defender's primary goal is to guarantee the distributed learning process can minimize the loss function f to an acceptable threshold, usually compared to the global minimum of the loss function [31]. In practice, it is also reasonable to measure the robustness of certain defense by the gaps among the model's utility (e.g., the accuracy of an image classifier) when the defense is equipped, unequipped with or without attacks. We will provide more details in Section 5.

4.1.3 Methodology Overview. Before detailing the implementations, we provide an overview of our proposed approach (Fig. 1). Robust distributed learning with GAA follows the following procedures: First, on receiving the submitted gradients from each worker, GAA, an additional module deployed on the server, executes certain policy to pose credit on each worker. Intuitively, GAA has limited credit in total and it will pose higher credit on the worker it trusts more (*Step 1*). Next,

GAA aggregates the gradients based on the credit and then proposes the weight update decision to the underlying learning process (*Step 2*). Finally, the learning process produces a reward signal based on the quasi-validation set, which is used to indicate the quality of the update direction (*Step 3*) and can further help GAA adjust its policy dynamically (*Step 4*).

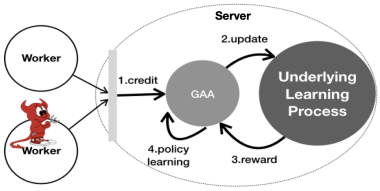

Figure 1: Overview of our proposed defense.

4.2 Distributed Learning as a Markov Decision Process

Following the conventions of Reinforcement Learning (RL) [53], we first define the notion of *environment*, with which an agent interacts. Standardly, the environment of a Markov Decision Process (MDP) is represented as a tuple $(\mathcal{S}, \mathcal{A}, R, p_0, p, \gamma)$, where $\mathcal{S}, \mathcal{A}$ are respectively the set of *states* and of *actions*, $R : \mathcal{S} \to \mathbb{R}$ is the reward function, $p_0 : \mathcal{S} \to \mathbb{R}^+$ is the initial probability density over states and $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition probability density, with $\gamma \in (0, 1]$ the discount factor. In the context of distributed learning, our specifications for these components are stated as follows. Fig. 2 shows an overview of our MDP settings.

Set of States $\mathcal{S}$. In the terminology of MDP, a state usually has the intuitive meaning as a context, based on which the agent makes a decision. Naturally, our GAA at iteration t refers to the tuple $s_t := (Q_t, \theta_t, \hat{f}_B(\theta_t))$ as the current state to decide the next weight update direction. Recall θ_t, Q_t are respectively the parameter and the received gradients at iteration t, while $\hat{f}_B(\theta_t)$ is defined as the loss at θ_t estimated by the server on the quasi-validation set B.

Set of Actions $\mathcal{A}$. Taking advantage of the simplicity of linear GAR, we propose to define the action space as an n-dimension simplex, where n is the number of workers. Generally speaking, our motivation here is to regularize the action space with prior knowledge and therefore the cost on searching the optimal policy can be largely scaled down. By restricting the feasible action to the space of linear GARs, GAA at each iteration chooses a candidate internal action $\alpha_t \in \mathbb{S}^n$ based on the current state s_t and the previous action α_{t-1}. Intuitively, this process can be considered as GAA's posing credit on each worker. Based on α_t, GAA then proposes the current update step as $\theta_{t+1} = \theta_t - \lambda(\sum_{i=1}^n \alpha_t^{(i)} V_n^t)$.

It is worth to notice, although the aggregation rule of GAA is linear in its form, it largely differs from linear GARs in that the coefficient α_t is chosen by a sophisticated agent adaptively at each iteration rather than predefined, which therefore makes our model immune to the vulnerability innate to linear GARs [11].

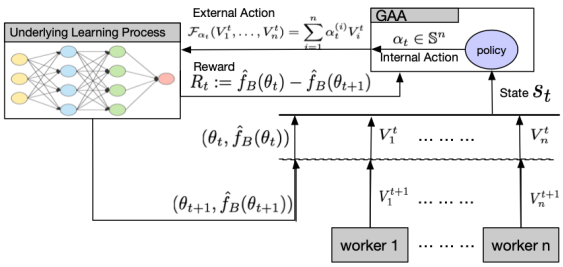

Figure 2: Distributed learning as an MDP.

Reward Function R. Reward function is usually defined as a function from each state s to a scalar value, which provides heuristics for policy learning. In our context, we set the reward at iteration t as $R_t := \hat{f}_B(\theta_t) - \hat{f}_B(\theta_{t+1})$, namely the relative loss decrease on the quasi-validation set B. Intuitively, if $\mathrm{KL}(P_m || \mathcal{D})$ is 0, the reward R_t highly reflects the changes in the true loss f [47] and thus provides a good guidance for GAA's policy learning. For other situations when P_m is similar but not necessarily identical with the true distribution, empirical studies show the reinforcement learning techniques still work well, probably due to its innate tolerance of noises in rewards [53].

Initial and Transition Probability Density p_0, p. Usually, these terms are partially unknown to an agent, which could only be estimated implicitly from observed *trajectories* [57]. Similarly, our GAA only has the partial knowledge regarding θ and $\hat{f}_B(\theta)$ of p_0, with random initialization of parameters, and of p, with the updating rule above, but totally ignorant of the initial distribution of Q_0 and its transition. In fact, the learning of GAA is exactly paralleled with an incrementally accurate estimation of p_0 and p, which equivalently means a better knowledge of the undertaking Byzantine attacks.

Discount Factor γ. Discount factor as a constant in $(0, 1]$ describes how the rewards in history influence the current decision, the value of which is determined by different application scenarios. Our configurations can be found in the evaluation parts.

4.3 Learning Optimal Policy for GAA

In the MDP setting above, our GAA is required to search for certain optimal *policy* $\pi(\alpha|s)$ to maximize the expectation of accumulated reward [54], where $\pi(\alpha|s)$ denotes a parametrized distribution over the action space $\mathcal{A}$, conditioned on the currently observed state s. Formally, the optimization objective for training GAA is defined as $\max_\pi \mathbb{E}_{s_0, a_0, \ldots, s_T, a_T}[\sum_{t=0}^T \gamma^t R(s_t)]$, where $(s_0, a_0, \ldots, s_T, a_T)$ is called a *trajectory* (or, *experience*) of length $T+1$, which has the joint probability density $p(s_0, \alpha_0, \ldots, s_T, \alpha_T) =$

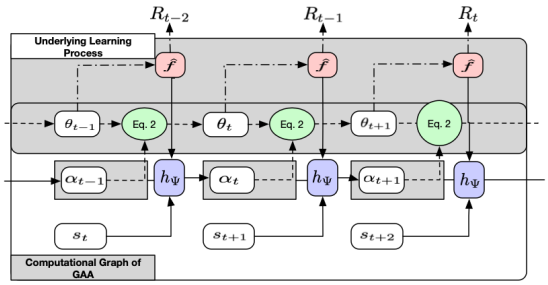

Figure 3: Implementation of GAA's policy as a general recurrent neural network.

$p_0(s_0)\prod_{t=1}^{T} p(s_t|s_{t-1},\alpha_{t-1})\pi(\alpha_{t-1}|s_{t-1})$.

In the context of RL, the objective above has been intensively studied and various mature algorithms such as policy gradient descent [54] or Q-learning [57] have been proposed to solve it. We expect our GAA can be seamlessly fused into the learning process of the underlying model with a similar behavior as statistical GARs. Therefore, we propose to approximately model the chained term $\prod_{t=1}^{T} p(s_t|s_{t-1},\alpha_{t-1})\pi(\alpha_{t-1}|s_{t-1})$ in the joint probability density with a general Recurrent Neural Network (RNN [27, 59]). The full computational graph of our proposed implementation is illustrated in Fig. 3. Starting from the initial state $s_0 \sim p_0$ and initial action $\alpha_0 := (\frac{1}{n},\ldots,\frac{1}{n})$, we formulate the auxiliary RNN as follows $\forall t \in \{0,\ldots,T-1\}, \alpha_{t+1} = h_\psi(s_{t+1},\alpha_t)$, where h_ψ denotes certain recurrent unit with parameter ψ, with its range as a subset of $\mathbb{S}^n$. Practically, such a condition can be easily realized with a softmax layer [10]. For details, please see Section 5.1.

Therefore, the optimization objective of GAA is reformulated as $\min_\psi \mathbb{E}_{s_0\sim p_0}[\sum_{t=0}^{T-1}\gamma^t(\hat{f}_B(\theta_{t+1})-\hat{f}_B(\theta_t))]$, where θ_t is uniquely determined with the update rule conditioned on α_{t-1} and θ_{t-1}. By expansion of $\hat{f}_B$, we can formulate the final optimization objective of GAA in episode i as $\min_\psi \frac{1}{S}\sum_{t=0}^{T-1}\gamma^t\sum_{z\in B} f(\theta_{t+1},z) - f(\theta_t,z)$, where θ_0 is initialized randomly while α_0 in episode i always inherits value from α_T in episode $i-1$. Our learning algorithm is listed in Algorithm 1.

4.4 Analytical Results

In this part, we present theoretical evidence on Byzantine robustness of distributed learning with GAA when the Byzantine ratio is fixed or fluctuates with uncertainty. Please note in the following analysis we focus on the empirical version of f on the training set, as the omitted leap from our proved results to f is guaranteed by standard results in generalization theory [58]. For the same reason, we maintain the notation f for its empirical version. We assume the loss function f is convex and η-smooth with pointwise bounded gradient $\|\nabla f\|_2 \le M$. For non-convex objective, our results can be extended with quadratic approximations [13]. Due to the page limit, we provide the detailed proofs for the results in this part at the website pertaining to this paper [1].

Algorithm 1: Robust Distributed Learning against Byzantine attacks with GAA

Initialize parameters of recurrent unit h_ψ randomly ;
Initialize $\alpha_{\text{old}} = \alpha_0 = (\frac{1}{n},\ldots,\frac{1}{n}) \in \mathbb{S}^n$;
for $i \in \{1,\ldots,N\}$ **do**
  Initialize parameters of f as θ_0 randomly ;
  for $k \in \{1,\ldots,K\}$ **do**
    $\alpha_0 \leftarrow \alpha_{\text{old}}, \ell_{\text{GAA}} \leftarrow 0$;
    for $t \in \{0,\ldots,T-1\}$ **do**
      Send the current parameters θ_t to each worker ;
      Receive submitted gradients $Q_t := (V_1^t,\ldots,V_n^t)$;
      $\theta_{t+1} \leftarrow \theta_t - \lambda(\sum_{i=1}^n \alpha_t^i V_i^t)$;
      $\ell_{\text{GAA}} \leftarrow \ell_{\text{GAA}} + \frac{1}{S}\gamma^t\sum_{z\in B} f(\theta_{t+1},z) - f(\theta_t,z)$;
      $\alpha_{t+1} \leftarrow h_\psi(s_{t+1},\alpha_t)$
    end
    Update ψ with a step of gradient descent on ℓ_{GAA} ;
    $\alpha_{\text{old}} \leftarrow \alpha_T$;
  end
end

4.4.1 Provable Robustness with a Fixed Byzantine Ratio.

Theorem 1. *After t steps of gradient descent with GAA when the Byzantine ratio is fixed as β, Algorithm 1 yields a parameter θ_t s.t.*

$$f(\theta_t) - f(\theta^*) < \frac{2RM}{\sqrt{(1-\beta)nt}} + \frac{S\eta R^2}{t} + \sqrt{2}\|f\|_\infty\sqrt{KL(P_m||\mathcal{D})} + O(e^{-t}) \tag{1}$$

where R is the diameter of parameter space.

Corollary 1. *As long as β is smaller than 1 and $P_m = \mathcal{D}$ a.e., Algorithm 1 in the above setting will asymptotically converge to the global optimum with rate $O(1/\sqrt{t})$.*

Intuitively, Theorem 1 suggests, when the Byzantine ratio is fixed over time, GAA is proved to help the underlying system attain a sub-optimal parameter with error $\varepsilon + O(\sqrt{\text{KL}(P_m||\mathcal{D})})$ in $O(\frac{1}{(1-\beta)\varepsilon^2})$ steps. It suggests a lower KL-divergence bound (at the scale of 10^{-2} in our case studies with a quasi-validation set constructed from similar data domains) and a smaller Byzantine ratio will lead to a more accurate sub-optimum. When the quasi-validation set is from the true distribution, Corollary 1 further guarantees the convergence of the learning process with rate $O(1/\sqrt{t})$, which is relatively larger than the optimal rate $O(1/t)$ in Byzantium-free learning case [14]. We provide a more detailed explanation on the meaning of each term and an empirical validation of Theorem 1 in Appendix A.4.

[1] https://bit.ly/2wjR2bb

4.4.2 Provable Robustness with a Fluctuated Byzantine Ratio.

Theorem 2. *After t steps of gradient descent with GAA when the Byzantine ratio fluctuates randomly other than* 1*, Algorithm 1 yields a parameter θ_t s.t.*

$$f(\theta_t) - f(\theta^*) < \frac{2RM + M}{\sqrt{t}} + \frac{S\eta R^2}{t} + \sqrt{2}\|f\|_\infty \sqrt{KL(P_m||\mathcal{D})} \tag{2}$$

where R is the diameter of parameter space.

Corollary 2. *Specifically, if $P_m = \mathcal{D}$ a.e., the learning process will asymptotically converge to the global optimum with convergence rate $O(1/\sqrt{t})$.*

Intuitively, Theorem 2 suggests, although there is still a guarantee for GAA to attain the sub-optimum in this case, the error term on the right of (2) is independent from β and is slightly larger than the one in (1). It is mainly because GAA in this case would pose all its credit on one single worker that is never compromised and therefore the distributed learning system degrades to a single-noded version when Byzantine ratio fluctuates. Similarly, Corollary 2 proves the convergence of GAA in this more challenging case when a golden-labeled validation set is available.

4.5 Byzantine Worker Detection & Behavior Analysis

In principle, when a policy is learned on how to determine an optimal action α_t according to the current state s_t and the historical information, our GAA is expected to master a good knowledge of the undertaking Byzantine attacks. Generally speaking, since the action proposed by our GAA is always constrained in $\mathbb{S}^n$, it is therefore reasonable to view each component of α_t as the *credit* on the corresponding worker. Specifically, we present its application in detection and behavioral pattern analysis of Byzantine workers below.

4.5.1 Byzantine Worker Detection. When the Byzantine ratio is fixed, accurate detection of Byzantine workers can help accelerate the learning process by eliminating potential Byzantine workers at an early stage. Therefore, we suggest detection algorithms should aim at selecting K most suspicious workers at iteration t. Although most statistical GARs are not directly applicable for detection tasks, we find one exception is GeoMed, for which we provide a straightforward extension as follows.

Procedure 1 (GeoMed+). *Given $Q_t = \{V_1^t, \ldots, V_n^t\}$,*
Step 1. *Initialize $O_t = \{\}$*
Step 2. *$O_t \leftarrow i^* := arg\max_{i\in\{1,\ldots,n\}} \sum_{V_j^t \in Q_t} \|V_i^t - V_j^t\|$*
Step 3. *$Q_t \leftarrow Q_t \backslash \{V_{i^*}^t\}$*
Step 4. *If $|O_t| = K$, output O_t. Otherwise, go to Step 2.*

As a comparison, Byzantine worker detection with GAA can be conducted in a more natural way.

Procedure 2 (GAA+).
Step 1. *Find K smallest coordinate of α_t.*
Step 2. *Output the corresponding index set as O_t*

4.5.2 Byzantine Behavior Analysis. When the Byzantine ratio fluctuates with unknown patterns, detecting temporal characteristics is a much more challenging task compared with the aforementioned case. Barely any previous statistical GARs can be adapted for addressing this task due to their lack of interpretability, while our proposed GAA can be applied directly for Byzantine behavior analysis with visualizations. In this case, we can visualize the policy sequence $\{\alpha_t\}$ to understand the temporal patterns of Byzantine attacks. A concrete demonstration on a situation when the Byzantine ratio fluctuates periodically is presented in Section 6.5.

5 Overview of Evaluations

5.1 Overall Settings

5.1.1 Benchmark Systems. We build GAA into the distributed learning process of four benchmark systems for text and image classification listed in Table 2. On MNIST and CIFAR-10, each worker shares a copy of the training set, while on Yelp and Healthcare, each worker has its local dataset. In all the cases, the loss function f is set as the cross entropy loss between the prediction of classifier g and the ground-truth. More details are provided in Appendix A.3.

Table 2: Summary of the benchmark systems.

	MNIST	**CIFAR-10**	**Yelp** [1]	**Healthcare** [2]
Model	MLP	ResNet-18	MLP	MLP
Task	Hand-Written Digits (10-class)	Objects (10-class)	Sentiment (2-class)	Disease (10-class)
# Samples	60k (Shared)	60k (Shared)	20k per worker (Local)	20k per worker (Local)
# Parameters	25,450	11,173,962	10,272	33,130
# Workers	50	50	10	50

5.1.2 Attacking Patterns. We consider the following three attack patterns of the adversary.

- **Static Attack:** All the βn compromised workers play the role of Byzantine workers during the whole learning process.
- **Pretense Attack**: In this case, the βn manipulated workers pretend to be benign in the first L rounds and start the attack from the $(L+1)$-th round.
- **Randomized Attack**: At beginning, each compromised worker (βn in total) is assigned with its role $r_i(0)$ by the adversary. During the learning process, it changes its role with a probability q at a period of p rounds.

It is worth to notice, the first pattern is a realization for the case in Section 4.4.1, when the Byzantine ratio is fixed over time, while the pretense and randomized attacks correspond to the setting in Section 4.4.2 when the Byzantine ratio fluctuates with or without uncertainty. Moreover, the latter two patterns are designed as adaptive attacks on the RL mechanism adopted by GAA. Both randomized attack and pretense attack attempt to mislead GAA into making wrong credit assignments, by letting the manipulated workers pretend to be benign and submit normal gradients in a certain time span of the learning process.

5.1.3 Tampering Algorithms. In experiments, we evaluate the impact of two realizations of the tampering algorithm $\mathcal{T}$.

- **Random Fault (RF) [11].** For RF, Byzantine workers submit noisy gradients sampled from a multi-dimensional Gaussian $\mathcal{N}(\mu, \sigma^2\mathbf{I})$. In our experiments, we take $\mu = (0.5, \ldots, 0.5) \in \mathbb{R}^d$ and $\sigma = 2 \times 10^{-6}$.
- **Adaptive Fault (AF).** For AF, we consider an adversary has some knowledge of the quasi-validation set, which allows the manipulated workers to submit well-crafted gradients that can tempt GAA to assign them with high credits and meanwhile maximize the overall training loss. We provide the details on the implementation of this fault in Section 6.3.

5.1.4 Implementation Details of GAA. We implement the recurrent unit h_ψ of GAA in the following experiments as a fully connected, feed-forward neural network with no hidden layer, with an input layer of size $(3n+2) \times d$ (i.e., the dimension of concatenation of s_t and α_t) and an output layer of size d with softmax activation. For other common hyperparameter settings in Algorithm 1, we set the learning rate λ as 0.05, discount factor γ as 0.9, the episode length T as 5, the number of episode N as 5. Each benign worker computes the gradient on randomly sampled mini-batch of size 64 for MNIST & CIFAR-10 and 256 for Yelp & Healthcare.

5.1.5 Choice of the Quasi-Validation Set B. For MNIST and CIFAR-10, we set the quasi-validation set as a random mini-batch of training samples. For Yelp and Healthcare, we implement the quasi-validation set as a small subset of samples from similar data domains. On Yelp, each worker holds 20k restaurants' reviews (randomly selected from the raw restaurant reviews) from one of the 10 US states with the most recorded Yelp reviews (including *Arizona, Illinois* and so on). We randomly sampled 1k reviews from *South California*, which is not in the top-10 states, as the full quasi-validation set. On Healthcare, each worker holds 20k treatment descriptions from local hospitals in one of the 50 different states, while we use a subset of descriptions from *Alaska* as the full quasi-validation set, which contains 1k records in total. For all our experiments on Yelp and Healthcare, we use less than 10 random samples from the full quasi-validation set as the working quasi-validation set.

5.2 Summary of Results

We highlight some experimental findings below.

- *Robustness* - GAA effectively defends the 4 benchmark systems against 3 attacking patterns and 2 tampering algorithms, with a wide range of configurations. It helps the underlying systems achieve comparable performance in limited rounds as if the systems were not under attacks.
- *Efficiency* - The time efficiency of GAA is on a similar scale with previous statistical defenses.
- *Interpretablity* - A well-trained GAA provides informative and interpretable traces that can be used for Byzantine worker detection and behavior pattern visualization.

6 Results & Analysis

6.1 Robustness against Static Attacks

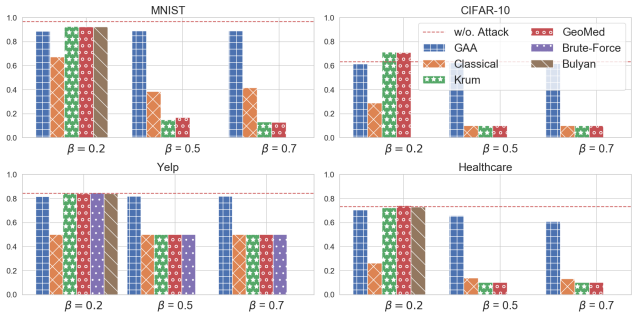

Figure 4: Test accuracy of the benchmark systems under static attacks when different defenses are applied up to a fixed round.

6.1.1 Comparison with Baselines. We compare the Byzantine robustness of our proposed GAA with 6 baselines under static attacks with RF: (A) Classical GAR (B) Brute-Force (C) GeoMed (D) Krum (E) Bulyan and (F) Classical GAR without attack. We include the last baseline for measuring the degradation of each method under attacks. We set the Byzantine ratio β in the static attack as 0.2, 0.5, 0.7, where 0.2 is a tolerable Byzantine ratio for all the baselines and 0.5 corresponds to the breaking point of the baselines. Fig. 4 shows the final test accuracy of the four benchmark systems with different defenses equipped, up to $5k, 10k, 20k, 40k$ rounds respectively. As Bulyan is not executable when $n \geq 4m+3$, the corresponding result is not collected when $\beta \geq 0.5$. Moreover, Brute-Force on MNIST, CIFAR-10 & Healthcare and Bulyan on CIFAR-10 fail to finish the learning in 10 days due to the high time complexity (we provide evaluations in Section 6.1.2 and Table 1), the corresponding results are not reported.

Results & Analysis. As we can see from Fig. 4, when the Byzantine ratio is as small as 0.2, each baseline method is observed to be Byzantine robust, which conforms to the reported results in previous works [31]. In this case, our GAA also

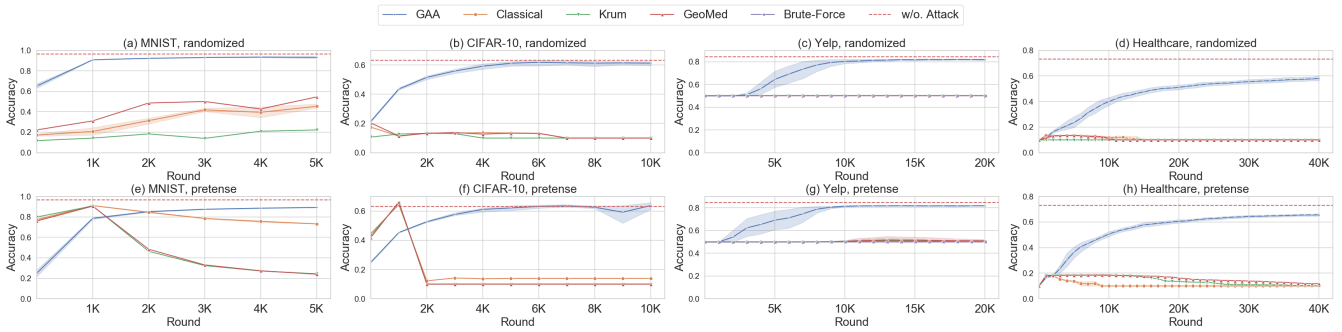

Figure 5: Learning curves of GAA against randomized attacks and pretense attacks.

helps the underlying model achieve a similar test accuracy. Noticeably, the robustness of our GAA is strongly demonstrated by its comparable performance to classical GAR without attack, when the Byzantine workers are in majority. For example, as the $\beta = 0.5$ cases represent the breaking point of Brute-Force, Krum and GeoMed, on Yelp the benchmark systems with the baseline defenses perform no better than a random guesser, while GAA helps the system achieve over 80% accuracy, which is very close to the 84.5% accuracy when the system is under no attack. A similar phenomenon was observed even when we further enlarge the Byzantine ratio to 0.7. These results imply GAA does complement the existing defenses when the Byzantine ratio is larger than 0.5.

6.1.2 Time Efficiency. We measure the time cost of our defense and provide a tentative comparison with previous defenses. We run the four benchmark systems with different defenses under the same static attack in the previous part and record the time cost of 100 iterations with 10 repetitions in the same environment described in Appendix A.1. Table 3 lists the running time of different defenses in each case. As the results imply, GAA brings computation overheads on a similar scale compared with previous defenses, which roughly corresponds to the theoretical complexity listed in Table 1.

Table 3: Time cost of distributed learning with each defense (sec. / 100 iterations), where - means the 100 iterations have not finished in one hour.

	Classical	GAA	GeoMed	Krum	Bulyan	Brute-Force
MNIST	6.32	8.14	15.85	15.79	698	-
CIFAR-10	116.85	129.50	118.73	118.69	-	-
Yelp	1.45	2.40	1.76	1.85	13.16	4.76
Healthcare	8.77	11.15	17.70	18.57	1877	-

6.2 Robustness against Adaptive Attacks on the RL mechanism

In this part, we evaluate the robustness of GAA when the adversary attempts to mislead the credit assignment by letting the manipulated workers pretend to be benign.

6.2.1 Comparison with Baselines. First, we evaluate the four benchmark systems under the randomized attack of $q = 0.5$, $p = 5$ and the pretense attack of $\beta = 0.7, L = 1000$, when GAA and other baseline defenses are equipped. Each worker is assumed to play the Byzantine role with RF. For randomized attacks, 24 out of 49 compromised workers are initially malicious on MNIST, CIFAR-10 & Healthcare and 4 out of 9 on Yelp. Fig. 5 plots learning curves of the benchmark systems when different defenses are equipped, where the shaded part of the curves denotes the variance of the accuracy within 10 repetitions.

Results & Analysis. As we can see from Fig. 5, , GAA is the only defense that is robust against both randomized and pretense attacks. For example, Fig. 5(a)&(e) shows GAA helps the benchmark system on MNIST achieve about 90% accuracy on average, which is close to the 96.4% accuracy of the system under no attack. As a comparison, the systems equipped with the baseline defenses either has final performance much lower than the expected or totally stagnate. Moreover, from Fig. 5(e)-(h), we find no fluctuation happens when the manipulated workers begin to attack after $1K$ rounds, which implies the RL mechanism of GAA is robust against pretense. Below, we present a more careful evaluation of GAA under a wide range of attack configurations.

6.2.2 GAA under Adaptive Attacks with Varied Configurations. Besides, we further evaluate GAA's robustness against the randomized attacks and the pretense attacks with diverse configurations on Yelp and Healthcare. Fig. 6 presents the learning curves of the underlying benchmark systems under attacks of varied configurations listed in the legends, where the shaded part of the curves denotes the variance of the accuracy within 10 repetitions.

Results & Analysis. As we can see from Fig. 6, under randomized Byzantine attacks of most configurations, GAA helps the benchmark systems on Yelp and Healthcare achieve desirable performance, compared with the accuracy of systems without Byzantine attacks. For example, in most configurations for Yelp, the final accuracy is around 83%, which is close to the optimal accuracy 84.5%. Although from Fig. 6(b) we notice the $q = 0.0$ case on Yelp has a larger variance, the average final accuracy is only about 10% lower compared with the optimal accuracy, which is still acceptable considering the high Byzantine ratio up to 0.7. Similarly, from Fig.

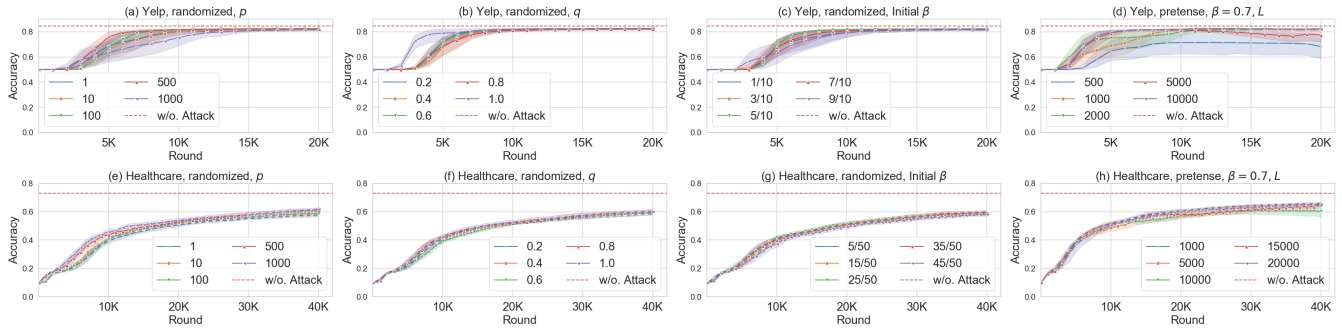

Figure 6: Learning curves of the benchmark systems on Yelp and Healthcare when GAA is applied for defending against randomized attacks with varied role-change period (the first column), role-change probability (the second column), initial Byzantine ratio (the third column) and against pretense attacks with varied pretense rounds (the last column). The legend describes the detailed configurations.

6(d)&(g), we also find the different configurations of the pretense attacks has very limited influence on GAA's defense quality.

6.3 Robustness against Adaptive Attacks on the Quasi-Validation Set

Although Assumption 1 and the randomness in the composition of the Quasi-Validation set (abbrev. QV set) imply the exact samples in the QV set is hard to be known by the adversary, we further examine the following two worst-case leakages of the QV set, which may allow the adversary to submit carefully crafted gradients (or called Adaptive Fault (AF)) based on the knowledge of the QV set to attempt to mislead GAA.

- *Case A.* The adversary knows the distribution where the QV set is sampled.
- *Case B.* Some classes are missing in the QV set and the malicious worker can target on the missing classes.

Intuitively, Case A is possible when the adversary expects GAA would use samples from similar data domains as the QV set, while Case B is possible when the QV set is too small to cover all different classes. It is worth to notice, for the adversary in Case A, the probability of determining the exact samples in the QV set is very low in theory, as the QV set contains less than 10 samples that are chosen independently by the server while the sample space of the distribution known to the adversary, practically the local dataset held by the manipulated worker, can contain as large as 10^3 samples when deep learning models are deployed.

In both cases, we consider the AF follows the same principle: it minimizes the loss on a dataset $\mathcal{D}_0$, which is chosen based on the knowledge about the QV set, to tempt GAA to assign the manipulated worker with high credit. In the meanwhile, the AF maximizes the overall loss on $\mathcal{D}_1$, (a subset of) its own training set, to compromise the whole distributed learning process. Accordingly, we formulate the gradient V_i^t submitted by a malicious worker (i.e., Worker i) at iteration t with AF by $V_i^t \propto \nabla_\theta(\ell(\theta^t, \mathcal{D}_0) - \alpha\ell(\theta^t, \mathcal{D}_1))$, where α is a hyperparameter that controls the stealthiness of the adaptive fault.

6.3.1 Adaptive Faults in Case A. We choose the $\mathcal{D}_0$ as the full QV set, and the $\mathcal{D}_1$ as the local training set of the manipulated workers. The parameter α in AF is set as 10. We conduct the GAA defense under three typical attack patterns listed in the legends of Fig. 7(a)&(b), which show the learning curves of the benchmark systems under the considered adaptive attack on the QV set.

Results & Analysis. From Fig. 7(a)&(b), we find in most cases the final accuracy of the benchmark systems remains close to the optimal accuracy. For example, under the combo adaptive attack on both the RL mechanism and the QV set (i.e., Config. *b* in Fig. 7(a)&(b)), GAA achieves respectively about 82% and 65% accuracy on Yelp and Healthcare, which is close to the performance of the system under no attack. The results imply that, GAA is robust against the adaptive adversary knowing the distribution where the QV set is sampled. From our perspective, lacking the knowledge of the exact QV set would let the adversary only count on his/her own inexact guess on the QV set. Hence, combining with the malice on maximizing the loss on the local training set, the gradient directions crafted by the malicious workers would be less effective in minimizing the loss on the QV set than the benign workers and therefore would be less trusted by GAA. However, when the adversary somehow knows the exact QV set the server uses, he/she would craft gradients that always minimize the loss on the QV set and mislead GAA to fully trust the manipulated worker, while this case would be rare, if not impossible, depending on the randomness of sampling and the security of the server.

6.3.2 Adaptive Attacks in Case B. In this setting, the manipulated worker can target on the missing classes by maximizing the loss on samples belonging to these missing classes, which forms the $\mathcal{D}_0$, while minimizing the loss of samples from other existing classes, which forms $\mathcal{D}_1$.

Experimental Settings. We first sample 10 records from the

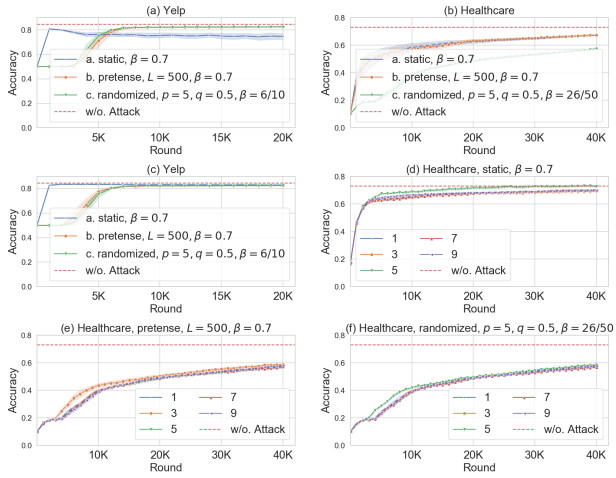

Figure 7: Learning curves of the benchmark systems on Yelp and Healthcare when GAA is applied for defending against adaptive faults in two cases of varied configurations.

full QV set on Healthcare (Yelp) to cover all the classes. For Healthcare, we reduce the number of classes from 9 to 1 with stride 2 by eliminating the samples belonging to the missing classes that we specify. For Yelp, we consider the case when the QV set contains only positive or only negative samples. With the QV sets with missing classes, we conduct the GAA defense against three typical attack patterns listed in the legends and titles of Fig. 7(c)-(f), which present the learning curves of the benchmark systems under the considered adaptive attack on the QV set.

Results & Analysis. As we can see from Fig. 7(c)-(f), even when the adversary targets on the missing classes in the QV set, GAA is still able to guarantee the benchmark systems to reach satisfying performance. For example, under static Byzantine attacks on Healthcare (in Fig. 7(d)), the final performance with 5 missing classes in the QV set is around 75%, even better than the 73.1% accuracy of the system under no attack. Also, Config. c in Fig. 7(c) and Fig. 7(f) demonstrates GAA remains robustness under combo attacks on the RL mechanism and the missing classes. Furthermore, we notice the number of missing classes has minor influence on GAA's defense quality, which strongly demonstrates the robustness of GAA against the adaptive adversary knowing the missing classes in the QV set.

6.3.3 GAA vs. Different Attacks. Despite the robustness of GAA against various attacks, the empirical performance does show subtle differences when GAA is against different attacks. For example, comparing Fig. 5 and Fig. 4, we find that the final accuracy of the benchmark systems under randomized and pretense attacks, two attacks exploiting the knowledge that GAA uses the RL mechanism to learn credit, is overall no better than that under static attacks. Similarly, as we can see from the corresponding results in Fig. 7 and Fig. 4, adaptive attacks that exploits the knowledge on the QV set are relatively more threatening than static attacks, where the threat is not further enlarged when the adversary exploits both the knowledge on the RL mechanism and the QV set, if comparing Config. b & c in Fig. 7(a) & (b) with the corresponding results in Fig. 5. These phenomena interestingly show, the more knowledge the adversary has of the deployed defense, the more threatening the attack could be against GAA.

6.4 Byzantine Worker Detection

In this part, we report the accuracy of Byzantine worker detection when the system is under static Byzantine attacks via our proposed GAA+ in Proc. 2, compared with the baseline method the GeoMed+ algorithm in Proc. 1.

Table 4: Precision-recall of Byzantine worker detection methods.

		GAA+	GeoMed+
$\beta = 0.3$	K=1	99.7%/6.65%	100%/6.67%
	K =5	99.7%/33.2%	100%/33.3%
	K =15	99.8%/99.8%	100%/100%
$\beta = 0.7$	K=1	99.9%/2.85%	0.0%/0.0%
	K =10	99.9%/28.5%	0.0%/0.0%
	K =35	99.9%/99.9%	57.1%/57.1%

Experimental Settings. By choosing Byzantine ratio $\beta = 0.3, 0.7$, we apply two detection algorithms on MNIST with the total number of workers as 50. Since we have defined the task of Byzantine worker detection as a top-K classification task, we report *precision/recall* in Table 4. Both precision and recall are calculated as an average over 1×10^3 randomly subsequent iterations after 1×10^4 iterations of distributed learning with GAA.

Results & Analysis. As we can see from above, with small Byzantine ratio, both GeoMed+ and our method achieve near perfect detection of each Byzantine worker. These empirical results not only justify that GeoMed+ is indeed a strong baseline, but also validates GAA+'s comparable performance with statistical counterparts in slight Byzantium. However, when the Byzantine ratio β is set up to 0.7, GeoMed+ fails to detect Byzantine workers any longer, while our method still detects each Byzantine worker perfectly, regardless of its majority in total.

6.5 Visualizing Byzantine Attack Patterns

In the final part of experiments, we present several interesting visualizations on the policy curve of GAA after learning under randomized attacks of $q = 1.0$, that is, each manipulated worker inverses its role periodically.

Experimental Settings. We consider two specific randomized attacks on MNIST with the following configurations: **(a)** $n = 10, q = 1.0, p = 1k$ with initial $\beta = 0.9$ and **(b)** $n = 10, q = 1.0, p = 400$ with initial $\beta = 0.5$. In other words, we consider the cases when all workers are manipulated and

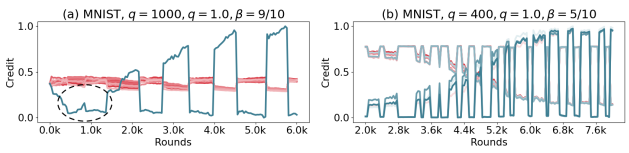

Figure 8: Capture periodic information of randomized Byzantine attack with GAA.

invert their role periodically. We collect GAA's action sequence in each configuration up to $40k$ rounds and plot the policy curves of each worker over a representative slice of iterations in Fig. 8 after normalization, where the policy curves for the initially Byzantine workers are warm-toned and the initially benign workers cool-toned.

Results & Analysis. First, in both cases the periodic characteristic of the undertaking Byzantine attack is captured well by our GAA, as its policy curve presents a period close to the ground-truth. To analyze with more care, we notice, in Fig. 8(b), as GAA's decision on Byzantine workers appears to be correct initially, its policy curve mainly evolves vertically. In other words, GAA tends to behave stable after an optimal policy is attained. Differently in Fig.8(a), although a low credit is assigned to the only initially benign worker in the first half period, GAA wisely skips the other half and swiftly adjust its policy in the subsequent period by heuristics of reward. The phenomenon is highlighted by the slashed region in Fig. 8(a).

7 Discussion

On Assumptions 1 & 2. Assumption 1 is used to guarantee the correct execution of Algorithm 1 and GAA itself would not be compromised by the adversary, while Assumption 2 is used to guarantee GAA has at least one worker to trust. We claim both assumptions are reasonable. On one hand, the former assumption is commonly assumed in previous studies of Byzantine robustness [4,11,15,16,20,31,62], which serves as a standing point of most published defenses, since otherwise the adversary could easily tamper the global model itself. On the other hand, the security level of the central server in real world distributed systems is always on a much higher level than working nodes, due to, e.g., rigorous access control mechanisms [55]. Therefore, the cost of attacks on central server is much higher than that on workers.

Moreover, we find it is quite straightforward to satisfy Assumption 2 if Assumption 1 is valid. For instance, the parameter server can spare certain computation resources to simulate one worker node on its own devices. Therefore, falling back on the properness of Assumption 1, we could claim the simulated worker is an always benign worker and thus satisfies the second assumption.

On Assumptions 3 & 4. These two assumptions regularize the range of learning tasks which GAA can help. Assumption 3 is again a commonly adopted assumption in most known

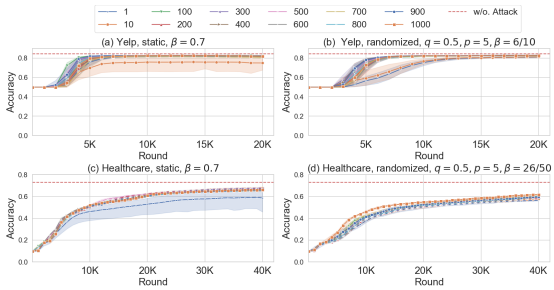

Figure 9: Learning curves on Yelp and Healthcare when GAA is equipped with varied size of the quasi-validation set.

defenses [4,11,15,16,20,31,62]. On one hand, if the workers share a copy of the same training set as in many conventional distributed learning systems (including the MNIST & CIFAR-10 cases) [3,34,41,43,50,64], both Assumptions 3 & 4 can be naturally satisfied due to the availability of a validation set from the same data source. For some newly proposed distributed learning systems (e.g., federated learning [34]) when the workers have their local datasets (including the Yelp & Healthcare cases), we demonstrate with the experimental results in Fig. 9, where we control the size of the QV set on Yelp and Healthcare to be 1 and $10, 100, \cdots, 1000$ by sampling from the full QV set, that the requirement on the QV set is relatively easy to be satisfied with only a small number of samples from similar data domains. For example, from Fig. 9(b), we find the final accuracy on Yelp under randomized attacks is both close to the bottleneck accuracy whenever the QV set size is 1 or $1k$, despite a slightly larger variance of performance and a lower convergence rate when the QV set is smaller. Moreover, experiments in Section 6.3 has proved that a small QV set is not likely to be exploited as a weak spot of the system whenever it may have missing classes or share a similar distribution with the local datasets of the manipulated workers. Despite this, we admit the QV set may be a weak spot for GAA if it is fully known by the adversary, while this case would be rare, if not impossible, in practice due to the randomness in preparing the QV set by the server and the security of the server.

For a validation of the requirement on the QV set in Assumption 4, we numerically estimate the average KL divergence among the local datasets and the full QV set on Healthcare. We find the empirical value is about 0.1. By inserting the empirical values of the KL divergence and the other terms in Section 4.4, we find the convergence rate predicted by Theorem 1 is quite close to the empirical learning curves. We provide more details in Appendices A.2 & A.4. However, GAA could have certain limitations to guarantee Assumption 4 when the server has no knowledge about the data domain of the undergoing distributed learning process or the learning protocol may have privacy requirements [61], which we leave as an interesting future work.

On Threat Model. Does the real world distributed learning

environment really show such malice that the Byzantine ratio has no explicit upper bound or even fluctuate? It may not the case for current distributed learning systems in stable local network environments [52]. Existing real world cases are, for example, distributed systems in unstable network environment with low-specification working machines, where a majority of nodes would send faulty gradients due to network or computation errors in an unpredictable manner. In this situation, GAA turns out to be a promising tool to help the underlying learning process converge to a near-optimal solution. Other possible use cases of GAA can be found in federated learning systems [34, 61], where end users are allowed to build a global learning model in cooperation. From our perspective, we suggest the threat model in this case should be formulated as malicious as possible, since the reliability of end users can be hardly guaranteed, similar to the case of DDoS attack [45].

Limitations and Future Directions. In one repetitive test of GAA, we observed a fluctuated test result on MNIST, which, based on our detailed analysis in Appendix A.5, could probably occur when the reward distribution of malicious workers is almost indistinguishable from that of benign workers. This may weaken the defense capability of GAA against attacks that aim at misclassification of targeted data samples instead of the overall accuracy we focus on in the current work. This kind of targeted attacks can be highly stealthy in terms of worker behavior [8] and remains an open challenge in building robust distributed learning systems [24].

Due to the limited access to distributed learning systems in industry, we have tried our best to cover typical use cases in image classification, sentiment analysis and intelligent healthcare, where the latter two are based on datasets from real-world applications and are minimally preprocessed to reflect the characteristics of data in practice. Nevertheless, more research efforts are required to provide a more thorough evaluation of GAA's security and performance in more application domains within industrial environments, which is very meaningful to be pursued as a future work. Although the distributed learning paradigm we study remains a mainstream techniques, there do exist other distributed learning paradigms such as second-order optimization based paradigms [50] or model-parallel paradigms [33]. To generalize GAA to more distributed learning paradigms will also be an interesting direction to follow.

8 More Related Work

Byzantine Robustness of Gradient-Based Distributed Learning Systems. Recent years, distributed learning systems under Byzantine attacks have aroused emerging research interests. Mainstream works in this field mainly focus on Byzantine robustness of the distributed learning protocol we introduce in Section 2. As we have reviewed in Section 3.2, most previous works are more interested in the defense side and usually utilize statistical approaches towards Byzantine robustness [4, 11, 16, 31, 62]. At the attack side, two very recent works [6, 25] have devised carefully-crafted attacks against Krum and GeoMed, while the attack techniques are highly dependent on the target defense and are hard to be generalized to GAA. Correspondingly, we in turn investigate the robustness of GAA under adaptive attacks on its own mechanism in Sections 6.2 & 6.3. During our paper preparation, we notice one recent work that also attempts to break the $\beta = 0.5$ bound [60]. The work is not learning-based and uses the loss decrease at the current iteration on the training set to rank the workers' credibility, which can be viewed a special case of our algorithm when the workers share the same training set and $T = 1$ in Algorithm 1. Moreover, the work only considers a 4-layer convolutional network on CIFAR-10 as the only benchmark system, while we provide more comprehensive evaluations in four typical scenarios, including the case they studied.

Byzantine Problem in Other Contexts. Aside from the aforementioned works on gradient-based distributed learning, there also exist some researches on other distributed learning protocols. For example, Chen et al. proposed a robust distributed learning protocol by requiring workers submitting redundant information [15]; Damaskinos et al. studied the Byzantine robustness of asynchronous distributed learning [20]; another thread of works exploited the vulnerability of distributed learning protocols where a worker is directly allowed to submit the local model to the master [5, 7, 28]. In this paper, we focus on the gradient-based distributed learning system model as studied by the mainstream defenses and therefore none of the aforementioned works are directly related to this paper.

Besides the Byzantine robustness in the context of machine learning, it has also been studied in many other contexts, like the multi-agent systems [46] and file systems [21], and was first studied in the seminal work by Lamport [37]. From a higher viewpoint on adversarial machine learning, challenges like adversarial example [30], data poisoning [9] and privacy issues [26, 44, 51] remain open problems and require future research efforts on building more robust and reliable machine learning systems.

9 Conclusion

In this paper, we have proposed the design of a novel RL-based defense GAA against Byzantine attacks, which learns to be Byzantine robust from interactions with the distributed learning systems. Due to the interpretability of its policy space, we have also successfully applied our method to Byzantine worker detection and behavioral pattern analysis. With theoretical and experimental efforts, we have proved GAA, as a promising defense and a strong complement to existing defenses, is effective, efficient and interpretable for guaranteeing the robustness of distributed learning systems in more general and challenging use cases.

Acknowledgement

We sincerely appreciate the shepherding from Yuan Tian. We would also like to thank the anonymous reviewers for their constructive comments and input to improve our paper. This work was supported in part by the National Natural Science Foundation of China (61972099, U1636204, U1836213, U1836210, U1736208, 61772466, U1936215, and U1836202), the National Key Research and Development Program of China (2018YFB0804102), the Natural Science Foundation of Shanghai (19ZR1404800), the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and the Ant Financial Research Funding. Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Institute for Advanced Communication and Data Science, and Engineering Research Center of CyberSecurity Auditing and Monitoring, Ministry of Education, China.

References

[1] https://www.yelp.com/dataset. Accessed: 2019-09-10.

[2] https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier2016.html. Accessed: 2019-09-10.

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.

[4] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *NeurIPS*, 2018.

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *ArXiv*, 1807.00459.

[6] Moran Baruch, Gilad Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *ArXiv*, 1902.06156.

[7] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *ArXiv*, 1811.12470.

[8] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin B. Calo. Analyzing federated learning through an adversarial lens. *ArXiv*, 1811.12470.

[9] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[10] Christopher M. Bishop and Nasser M. Nasrabadi. Pattern recognition and machine learning. *J. Electronic Imaging*, 2007.

[11] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[12] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 1998.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 2015.

[15] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: byzantine-resilient distributed training via redundant gradients. *ArXiv*, 1803.09877.

[16] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *POMACS*, 2017.

[17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending mnist to handwritten letters. *IJCNN*, 2017.

[18] Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *STOC*, 2016.

[19] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. Torch: a modular machine learning software library. Technical report, 2002.

[20] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. Asynchronous byzantine machine learning (the case of sgd). *ArXiv*, 1802.07928.

[21] Miguel Oom Temudo de Castro. Practical byzantine fault tolerance. In *OSDI*, 1999.

[22] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *NeurIPS*, 2012.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, 1810.04805.

[24] Peter Kairouz et al. Advances and open problems in federated learning. *ArXiv*, 1912.04977.

[25] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. *ArXiv*, 1911.11815.

[26] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*, 2015.

[27] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 1993.

[28] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *ArXiv*, 1808.04866.

[29] Rainer Gemulla, Erik Nijkamp, Peter J. Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *KDD*, 2011.

[30] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ArXiv*, 1412.6572.

[31] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2015.

[33] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *ArXiv*, 1811.06965.

[34] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *ArXiv*, 1610.05492.

[35] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[37] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *TOPLAS*, 1982.

[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[40] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *ArXiv*, 1804.08838.

[41] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *NeurIPS*, 2014.

[42] Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *NeurIPS*, 2015.

[43] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *ArXiv*, 1602.05629.

[44] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *S & P*, 2019.

[45] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 2004.

[46] Fabio Pasqualetti, Antonio Bicchi, and Francesco Bullo. Consensus computation in unreliable networks: A system theoretic approach. *IEEE Transactions on Automatic Control*, 2010.

[47] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. 1985.

[48] Peter J Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 1985.

[49] Ahmed Salem, Apratim Bhattacharyya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. *ArXiv*, 1904.01067.

[50] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *ICML*, 2014.

[51] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *S & P*, 2017.

[52] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[53] Richard S Sutton, Andrew G Barto, Francis Bach, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[54] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 2000.

[55] Andrew S Tanenbaum and Maarten Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[56] John N. Tsitsiklis, Dimitri P. Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *American Control Conference*, 1984.

[57] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 1992.

[58] Jon Wellner et al. *Weak convergence and empirical processes: with applications to statistics*. Springer Science & Business Media, 2013.

[59] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 1990.

[60] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *ICML*, 2018.

[61] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *TIST*, 2019.

[62] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *ArXiv*, 1803.01498.

[63] Sixin Zhang, Anna Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *NeurIPS*, 2015.

[64] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *NeurIPS*, 2010.

A Other Details

A.1 Experimental Environments

All the defenses and experiments are implemented with Torch [19], which is an open-source software framework for numeric computation and deep learning. All our experiments are conducted on a Linux server running Ubuntu 16.04, one AMD Ryzen Threadripper 2990WX 32-core processor and 2 NVIDIA GTX RTX2080 GPUs. We simulate the distributed learning setting by sequential computation of gradients on randomly sampled mini-batches.

A.2 Estimate KL-divergence

We design the following procedures to estimate the pairwise KL-divergence between datasets D_i and D_j on Healthcare, which consist of samples of form (x,y) s.t. $x \in \mathbb{R}^n$, $y \in [K]$, where $n = 1024$ and $K = 10$. Fig. 10 shows the heatmap of the KL-divergence among the local datasets on each worker and the full QV set. The empirical KL-divergence is about 0.16 on average.

1. Train one probabilistic model $p_i(y|x)$ for each dataset D_i to a certain error threshold.
2. Do uniform sampling over $[-0.5, 0.5]^n$ for N times to form a set of points $\{x_k\}_{k=1}^N$.
3. Calculate the empirical KL-divergence between the joint distributions that underlie D_i, D_j by

$$KL(D_i||D_j) = \frac{1}{K \times N} \sum_{k=1}^{N} \sum_{c=1}^{K} p_i(x_k|y=c) \log \frac{p_i(x_k|y=c)}{p_j(x_k|y=c)} \tag{3}$$

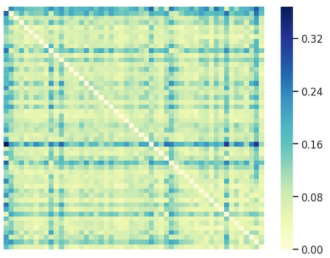

Figure 10: Estimated KL-divergence among local datasets and the prepared validation set on Healthcare.

However, it is true that it is challenging to estimate the KL-divergence when the QV set is very small. To leverage the above algorithm for estimation, ideally we require the knowledge of the distribution where the QV set is sampled, so that we can estimate the conditional distribution $p(y|x)$ via learning-based approaches. Intuitively, if QV set contains more samples, the estimated conditional distribution is less biased and thus the error of estimating the KL-divergence is smaller. To be concrete, the minimum requirement for conducting the estimation is, the QV set should contain at least one sample from each class and thus we can estimate the conditional distribution with support vector classifier or K-Nearest Neighbor (KNN). As a future work, it would be a meaningful direction to study how to guarantee a low KL-divergence in a distributed learning protocol that may have privacy requirements [61].

A.3 Details of the Benchmark Systems

1. **MNIST**: The first case is training a fully connected feed-forward neural network for the hand-written digital classification task on the MNIST dataset [39], with 50 workers. This public dataset contains 60000 28×28 images of 10 digits for training and 10000 for testing. Each worker shares a copy of the training set. The model consists of 784 inputs, 10 outputs with soft-max activation and one hidden layer with 30 rectified linear units (ReLu [36]). The dimension of parameters is 25450.
2. **CIFAR-10**: The second case is training a ResNet-18 [32] model for the image classification on the CIFAR-10 dataset [35] with 50 workers. This dataset contains 60000 $28 \times 28 \times 3$ images of 10 classes of objects for training and 10000 for testing. Each worker shares a copy of the training set. The standard model ResNet-18 has 18 end-to-end layers and 11173962 learnable parameters in total.
3. **Yelp**: The third case is training a fully connected feed-forward neural network for the sentiment classification task (i.e., binary classification on positive or negative attitude), with 10 workers. Each worker has 20000 1024-dimension features of Yelp reviews for restaurants in its local metropolitan area [1]. Each worker corresponds to one metropolitan area. The features are extracted with a pretrained Bert language model by Google [23]. We removed a fraction of data samples from each worker to form the test set, which consists of 1000 samples per class. The model consists of 1024 inputs, 2 outputs with soft-max activation and one hidden layer with 10 sigmoid units. The dimension of parameters is 10272.
4. **Healthcare**: The fourth case is training a fully connected feed-forward neural network for predicting the healthcare provider type (10 classes) from textual treatment descriptions, with 50 workers. Each worker has 20000 1024-dimension Bert features of treatment descriptions from its local hospitals. Each worker corresponds to a state. The dataset is prepared from CMS public healthcare records [2] and we removed a fraction of data samples from each worker to form the test set, which consists of 1000 samples per class. The model consists of 1024 inputs, 10 outputs with softmax activation and one hidden layer with 32 sigmoid units. The dimension of parameters is 33130.

A.4 An Empirical Validation of the Analytic Results

Without loss of generality, we take Theorem 1 as an example. First, we explain the terms R, M, α and S one by one with more care and give the empirical values on Healthcare for demonstration. In general, our terminology follows the conventions in [14], a standard text on optimization theory.

- Diameter R: The diameter R of a parameter space Θ (i.e., the feasible set of parameters of the underlying learning model) is defined as the maximal 2-norm of an element $\theta \in \Theta$. Formally, $R = \sup\{\|\theta\|_2 : \theta \in \Theta\}$. On Healthcare, we estimate the 2-norm of the flattened parameter of the neural network during the learning process to estimate as the scale of R, which is plotted in Fig. 11(a). The average value of R is around 11.05.
- Upper bound of gradient norm M: The term M is used to denote the upper bound of the gradient norm. Formally, $M = \sup_{\theta \in \Theta} \|\nabla_\theta \hat{f}(\theta, D_{\text{train}})\|_2$. On Healthcare task, we compute the 2-norm of the gradient submitted by the always-benign worker during the learning process to estimate the scale of M, which is plotted in Fig. 11(b). The average value of M is around 0.36.
- Smoothness factor η: The term η occurs in our assumption that the loss function f is η-smooth. Formally, the loss function f is said to be η-smooth if $\forall \theta_1, \theta_2 \in \Theta$, $|\hat{f}(\theta_1, D_{\text{train}}) - \hat{f}(\theta_2, D_{\text{train}})| \leq \eta \|\theta_1 - \theta_2\|_2$. We estimate the empirical scale of α by calculating the expressions at both sides of the definition during the learning process, which is plotted in Fig. 11(c). The average value of η is around 0.50.
- Size of mini-batch S: The term S denotes the training size of the mini-batch on which the always-benign worker calculates the gradient. In addition, S is required to be no less than 1 (i.e., the training set contains at least one sample) or otherwise the theorem is invalid. On Healthcare, S is set as 256.
- Finally, the max-norm of the loss function (which is implemented as a cross-entropy) is upper bound by the maximal entropy of the K-class classification task (i.e., $\|f\|_\infty \leq \frac{1}{K} \ln K$, which is about 0.23 for $K = 10$ on Healthcare), while the estimated KL divergence term is about 0.16 from Fig. 10.

Therefore, on Healthcare under static Byzantine attacks with $\beta = 0.7, n = 50$, the numeric form of Theorem 1 writes as

$$f(\theta_t) - f(\theta^*) < \frac{2.05}{\sqrt{t}} + \frac{16.58}{t} + 0.13 + O(e^{-t}) \quad (4)$$

which produces the curve of the predicted training loss in Fig. 11(d). Compared with the empirical training loss curve, we find the prediction from Theorem 1 roughly conforms to GAA's empirical behavior in this case.

A.5 Analysis of a Fluctuated Phenomenon on MNIST under Randomized Attacks

In one repetitive test of GAA, we noticed a fluctuated test result on MNIST under randomized attacks of $p = 0.5, q = 5$, initially $\beta = 26/50$, which we report below in Fig. 12. In fact, through a larger number of repetitive experiments, we have observed this phenomenon only on MNIST but not on other three benchmarks. We would like to clarify that this

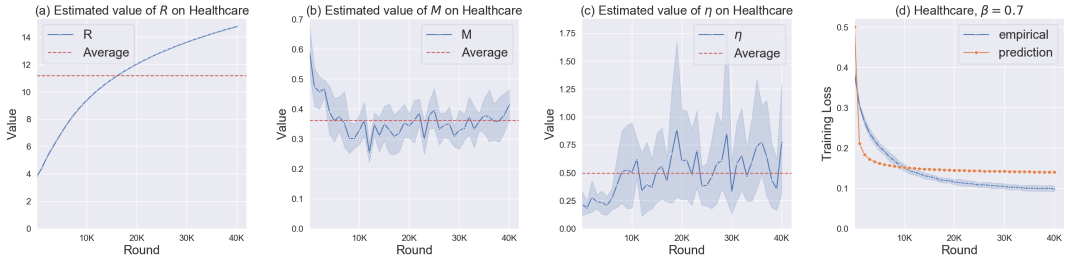

Figure 11: Empirical values of the theoretical terms in Theorem 1, alongside the predicted training loss curves.

phenomenon is not a common case in repetitive tests and we reported this result here mainly because we think this singular phenomenon may help the readers understand the behavior of GAA more thoroughly. Below, we further investigate the possible causes of this phenomenon.

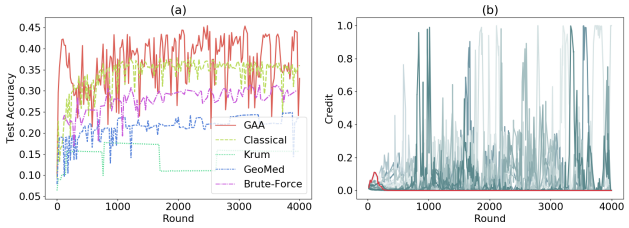

Figure 12: An observed fluctuated run of GAA defense on MNIST under the randomized attack: (a) its learning curve and (b) its policy curves.

As we can see from Fig. 12, the policy curve of GAA is more unstable than that in other cases, which in other words means GAA's credit on each worker fluctuates a lot. This phenomenon indicates that GAA somehow could not recognize the always benign worker in this situation. As a hypothesis, we speculate the reason as the low complexity of the MNIST task [17, 40, 49], which makes the reward from the workers' gradient on MNIST is not as distinguishable as in other cases. To validate this point, we plot the distribution of the rewards (i.e., the relative loss decrease) yielded by the benign workers and the randomized Byzantine workers on each benchmark as follows.

In detail, we set the worker number as 2 and set their roles respectively as benign and Byzantine with the RF tampering algorithm. We execute the classical distributed learning protocol for 10 epochs over the corresponding training set and collect the yielded reward (calculated on the quasi-validation

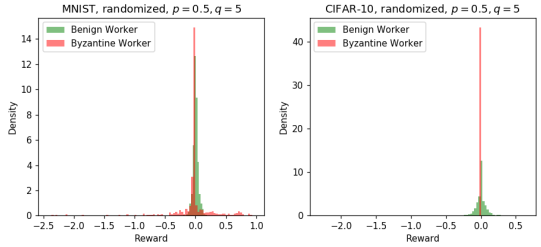

Figure 13: Distribution of rewards from benign workers and from randomized Byzantine workers on MNIST and CIFAR-10.

set of the same settings in Section 5.1) respectively from the benign and Byzantine workers for every $1k$ iterations. We then plot the histogram of rewards on MNIST and CIFAR-10 in Fig. 13.

As we can see from Fig. 13, on CIFAR-10 the Byzantine worker always yields zero reward, which is highly divergent from that of the benign worker. Differently, on MNIST the Byzantine worker and the benign worker yield rewards that follow similar distributions, which thus may bring difficulties for GAA to distinguish one from the other. A noticeable point is the Byzantine worker tends to yield rewards that distribute in a slightly wider range than the benign one, which could be another cause of the instability in GAA's learning curve on MNIST. This speculation is also supported by the MNIST case under static Byzantine attacks of ratio over 0.5 & 0.7 (in Fig. 4), where the baseline methods were observed to perform slightly stronger than the random-guess, while on other datasets they did not. This phenomenon suggests that the model on MNIST still learns something from even incorrect gradients.