红色文本框表示定义

联邦学习本质上是一种分布式机器学习技术，或机器学习框架。
联邦学习的分类：横向联邦、纵向联邦、联邦迁移。

Mitigating Sybils in Federated Learning Poisoning

Sybils:Sybil Attack，中文翻译为“女巫攻击”，2002年由John R. Douceur在《the Sybil Attack》提出，它是p2p网络中的一种攻击形式，攻击者利用单个节点来伪造多个身份，是攻击数据冗余机制的一种有效手段。

Clement Fung
University of British Columbia
cfung1@cs.ubc.ca

Chris J.M. Yoon
University of British Columbia
yoon@alumni.ubc.ca

Ivan Beschastnikh
University of British Columbia
bestchai@cs.ubc.ca

***Abstract*—Machine learning (ML) over distributed data is relevant to a variety of domains. Existing approaches, such as federated learning, compose the outputs computed by a group of devices at a central aggregator and run multi-round algorithms to generate a globally shared model. Unfortunately, such approaches are susceptible to a variety of attacks, including model poisoning, which is made substantially worse in the presence of sybils.**

In this paper we first evaluate the vulnerability of federated learning to sybil-based poisoning attacks. We then describe *FoolsGold*, a novel defense to this problem that identifies poisoning sybils based on the diversity of client contributions in the distributed learning process. Unlike prior work, our system does not assume that the attackers are in the minority, requires no auxiliary information outside of the learning process, and makes fewer assumptions about clients and their data.

In our evaluation we show that FoolsGold exceeds the capabilities of existing state of the art approaches to countering ML poisoning attacks. Our results hold for a variety of conditions, including different distributions of data, varying poisoning targets, and various attack strategies.

I. Introduction

To train multi-party machine learning (ML) models from user-generated data, users must provide and share their training data, which can be expensive [20] or privacy-violating. Federated learning [23], [24] is a recent solution to both problems: while training on data stored on mobile devices, data is kept on the client device and only model parameters are transferred to a central aggregator when constructing the model. This provides a basic level of privacy and allows clients to compute their model updates locally and independently. Federated learning can be further augmented with differential privacy [15] and secure aggregation [10] to provide additional client-side privacy and security.

However, federated learning introduces a risky design tradeoff: clients, who previously acted only as passive data providers, can now observe intermediate model state and contribute arbitrary intermediate values as part of the decentralized training process. This creates an opportunity for malicious clients to manipulate the training process: adversaries posing as honest clients can pollute the shared model [6], [27], influencing the prediction probabilities of the final model. This type of attack, in which an adversary maliciously influences the properties of a trained model, has been well explored in centralized settings: poisoning attacks [21], [8], [27] and backdoor attacks [3], [12], [17] are two prevalent examples.

Sybils [14] are a common strategy for adversaries to increase the effectiveness of attacks on crowd-sourced systems. For example, using a mobile device simulator, an adversary can simulate fake accounts to push the result of crowd-sourced computation towards a malicious goal [35]. To perform a similar attack on federated learning, an adversary can simulate multiple user accounts that gain admission and contribute to the federated learning model.

TABLE I. The accuracy and attack success rates for baseline (no attack), and attacks with 1 and 2 sybils in a federated learning context with MNIST dataset.

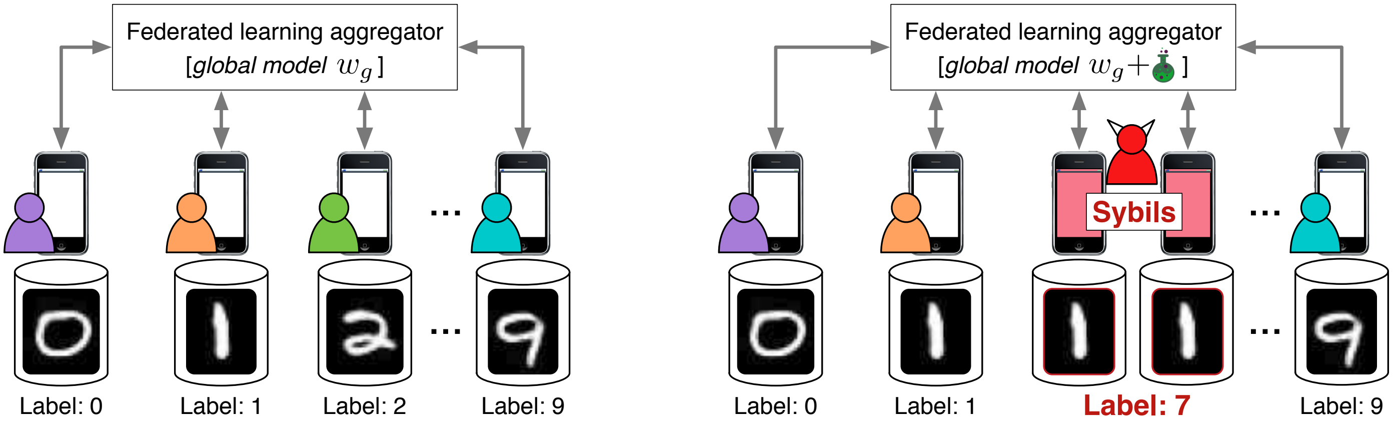
	Baseline	Attack 1	Attack 2
# honest clients	10	10	10
# malicious sybils	0	1	2
Accuracy (digits: 0, 2-9)	90.2%	89.4%	88.8%
Accuracy (digit: 1)	96.5%	60.7%	0.0%
Attack success rate	0.0%	35.9%	96.2%

A federated learning poisoning experiment. We illustrate the vulnerability of federated learning to sybil-based poisoning with three experiments based on the setup in Figure 1 and show the results in Table I. First, we recreate the baseline evaluation in the original federated learning paper [23]: training an MNIST [22] digit classifier across non-IID data sources (Figure 1(a) and Baseline column in Table I). We train a softmax classifier across ten honest clients, each holding a partition of the original MNIST dataset that contains only one of the ten digits. Each model is trained for 3000 synchronous iterations, in which each client performs a local SGD update using a batch of 50 examples, sampled at random.

We then re-run the baseline evaluation with a targeted ”1-7” poisoning attack [6]: each malicious client in the system has a set of 1s labeled as 7s (Figure 1(b)). A successful attack generates a model that is unable to correctly classify 1s and incorrectly predicts them to be 7s. We define the *attack success rate* as the proportion of 1s predicted to be 7s by the final model in the test set. We perform two experiments, with 1 or 2 malicious sybil clients (Attack 1 and Attack 2 in Table I).

Table I shows that with only two sybils, 96.2% of 1s are predicted as 7s in the final shared model. Since one honest client held the data for digit 1, and two malicious sybils held the poisoned ”1-7” data, the malicious sybils were twice as influential on the shared model as the honest client. This attack illustrates a problem with federated learning: *all clients have equal influence* on the system[1]. Thus, to overcome honest clients, an adversary may gain influence over the global model by generating sybils. The number of sybils required by the adversary for successful poisoning is unknown: as the number of honest clients opposing the attack increases, the number of sybils to poison the model also increases, creating a blind tug-of-war scenario.

[1] The only weighting factor in federated learning is the number of examples that each client possesses, but this weight is easy for adversaries to inflate by generating more data or by cloning their dataset.



(a) Federated learning setting **(b) Federated learning with sybil-based label-flipping poisoning**

Fig. 1. Federated learning with and without colluding sybils mounting a sybil-based poisoning attack. In the attack (b) two sybils poison the model by computing over images of 1s with the (incorrect) class label 7.

Modern solutions to sybil attacks on crowd-sourced computation systems involve detecting malicious users and removing their influence. This leverages social network information [33] or incorporates auxiliary information into the algorithm [34], [35]. In our work, we do not make such assumptions.

In federated learning, the server only observes model parameters sent as part of the iterative ML algorithm. One may consider that a simple outlier detection method is sufficient to detect and remove sybils, but this is difficult for a variety of reasons. Since the server is unable to view the training data of clients in federated learning and does not possess a validation dataset, they cannot easily verify which incoming parameter updates are genuine. Furthermore, these updates are a product of a random stochastic process and will exhibit high variance. The key insight in this work is that when a shared model is being manipulated by a group of sybils, they will, in expectation over the entire training process, contribute updates towards a shared specific malicious objective, exhibiting behavior that is more similar amongst themselves than what is naturally observed in the multi-party ML setting.

To mitigate malicious clients in federated learning, existing solutions assume either costly third party user verification control [10], [37] or use explicit parameterization of the expected number of sybils [9]. We propose a design that does not rely on external user authentication to defend against sybils in federated learning, and to the best of our knowledge, we are the first to mitigate sybil-based poisoning attacks without explicit parameterization of the number of attackers.

We propose **FoolsGold**: a new defense for sybil attacks on federated learning that penalizes sybil contributions based on *contribution similarity*. FoolsGold defends federated learning from attacks performed by an arbitrary number of sybils, with minimal changes to the original federated learning procedure and no assumed parameterization of expected attackers. The core of our approach is to use a similarity distance function on client contributions to the learning process. We evaluate FoolsGold on 3 diverse data sets (MNIST [22], KDDCup [13], Amazon Reviews [13]) and find that our approach mitigates attacks under a variety of conditions, including different distributions of client data, varying poisoning targets, and various attack strategies.

In summary, we make the following contributions:

- ⋆ We consider sybil attacks on federated learning architectures and show that existing defenses against malicious adversaries in ML (RONI [5] and Krum [9]) are inadequate.
- ⋆ We design, implement, and evaluate a novel defense for the federated learning setting that employs gradient similarity between clients to thwart sybil attacks.
- ⋆ In this context, we discuss optimal and intelligent attacks that adversaries can perform, while suggesting possible directions for further mitigation.

thwart:阻止

II. Background

Machine Learning (ML). Many ML problems can be represented as the minimization of a loss function in a large Euclidean space. For example, an ML binary classification task uses features from a single data example to predict a discrete binary output; more prediction errors result in a higher loss. Given a set of training data and a proposed model, M algorithms *train*, or find an optimal set of parameters, for th given training set.

Stochastic gradient descent (SGD). One approach in M is to use stochastic gradient descent (SGD) [11], an iterative algorithm that selects a batch of training examples, uses them to compute gradients on the parameters of the current model, and takes gradient steps in the direction that minimizes the loss function. The algorithm then updates the model parameters and another iteration is performed. SGD is a general learning algorithm that can be used to train a variety of models, including deep learning [11]. We assume SGD as the optimization algorithm in this paper.

In SGD, the model parameters w are updated at each iteration t as follows:

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \frac{1}{b}\sum_{(x_i,y_i)\in B_t} \nabla l(w_t, x_i, y_i)) \qquad (1)$$

where η_t represents a degrading learning rate, λ is a regularization parameter that prevents over-fitting, B_t represents a gradient batch of training data examples (x_i, y_i) of size b and ∇l represents the gradient of the loss function.

Batch size is what separates SGD from its counterpart, gradient descent (GD). In GD, the entire dataset is used to compute the gradient direction, but in SGD, a subset is used at each iteration. This subset may be selected in a pre-determined order or sampled randomly, but the overall effect of SGD is that the gradient directions seen over time vary (and have higher variance as the batch size b decreases). In practice, SGD is preferred to GD for several reasons: it works well on large datasets and theoretically scales to datasets of infinite size.

A typical heuristic involves running SGD for a fixed number of iterations or halting when the magnitude of the gradient falls below a threshold. When this occurs, model training is considered complete and the parameters w_t are returned as the optimal model w^*.

Federated learning [23]. In FoolsGold, we assume a standard federated learning context, in which data is distributed across multiple data owners and cannot be shared. The distributed learning process is performed in *synchronous update rounds* over a set of clients, in which a weighted average of the k client updates, based on their proportional training set size n_k out of n total examples, is applied to the model atomically.

$$w_{g,t+1} = w_{g,t} + \sum_k \frac{n_k}{n} \Delta_{k,t}$$

Even if the data is distributed such that the data is not independent and identically distributed (non-IID), federated learning can still attain convergence. For example, we need federated learning to train an MNIST [22] digit recognition classifier in a setting where each client only held data for 1 or 2 of the digit classes (0-9), as in Figure 1(a).

Federated learning also enables model training across a set of clients with highly private data [30]. For example, differentially private [15] and securely aggregated [10] additions have been released; however, the use of federated learning in multi-party settings still presents new privacy and security vulnerabilities [19], [3]. In this paper we design FoolsGold to address sybil poisoning attacks.

Sybil attacks. A system that allows clients to freely join and leave may be attacked by sybils [14], in which an adversary gains leverage by joining a system under multiple colluding aliases. In FoolsGold, we assume that adversaries leverage sybils to mount more powerful poisoning attacks.

Poisoning attacks on ML. In a poisoning attack [6], [25], an adversary meticulously creates adversarial training examples and inserts them into the training data set of an attacked model. This may be done to degrade the accuracy of the shared model (a random attack), or to increase/decrease the probability of a targeted example being predicted to be a targeted class with the trained model (a targeted attack) [21] (see Figure 1(b)). For example, such an attack could be mounted to avoid fraud detection or to evade email spam filtering [27]. In FoolsGold, we only consider targetted attacks.

Generally, it is also important that a poisoning attack does not significantly change the prediction outcomes of other classes. Otherwise, the attack will be detected by users of the shared model.

In federated learning, the aggregator does not see any training data, and we therefore view poisoning attacks from the perspective of model updates: a subset of updates sent to the model at any given iteration are poisoned [9]. This is functionally identical to a centralized poisoning attack in which a subset of the total training data is poisoned. Figure 2 illustrates a targeted poisoning attack in an SGD context.

Poisoning defenses for ML. A **RONI** (Reject on Negative Influence) defense [5] counters ML poisoning attacks. When evaluating a set of suspicious training examples, this defense trains two models: one model using a trusted dataset, and another model using the union of the trusted and suspicious data. If the performance of ths model is significantly worse than the performance on the trusted model, the data is rejected.

In FoolsGold, we do not rely on the availability of such a validation dataset to defend against poisoning attacks. We compare FoolsGold against RONI in Section V-B.

Robust methods in ML. Training models with robust loss functions creates models that resist the presence of outliers [18]. These loss functions place a lower weight to points that are far from the global training data distribution than points that fall within the distribution.

Multi-Krum [9] is a robust learning method that is specifically designed to counter adversaries in federated learning. In Multi-Krum, contributions to the model that are too dissimilar from the mean client contribution are flagged as anomalous and removed from the aggregated gradient. Multi-Krum uses the Euclidean distance to determine which gradient contributions are removed, requires parameterization of the maximum number of adversaries expected, and can theoretically withstand Byzantine attacks of up to 33% adversaries in the client pool.

In FoolsGold, we do not assume that the maximum number of attackers is known, and do not rely on any outlier-based techniques. We compare FoolsGold against Multi-Krum in Section V-B.

III. Assumptions and Threat Model

Setting assumptions. We are focused on federated learning and therefore assume a non-IID setting: honest clients do not have similar data. More formally, the degree of dissimilarity between client training data sets fits the federated learning motivation: any local model fit on an individual client's data is dissimilar enough from the globally fit model such that the local model's performance would be poor.

The adversary can only access and influence ML training state through the federated learning API, and adversaries cannot observe the training data of other honest clients. This means that sybils are unable to observe the gradient updates coming from honest clients in federated learning. By observing the total change in model state, adversaries can observe the total averaged gradient across all clients, but cannot view the gradient from any individual honest client.

On the server side of the algorithm, the aggregator is uncompromised and is not malicious. Similarly, we also assume that some number of honest clients (who possess training

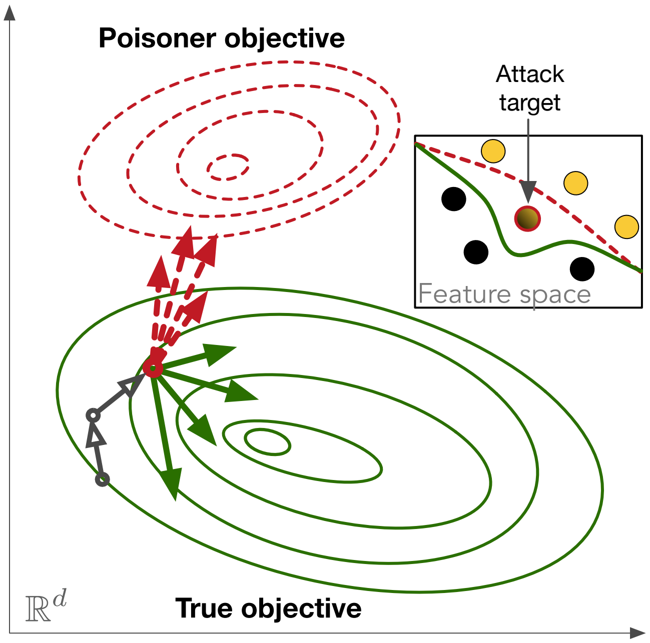


Fig. 2. Targeted poisoning attack in SGD. The dotted red vectors are sybil contributions that drive the model towards a poisoner objective. The solid green vectors are contributed by honest clients that drive towards the true objective.

data which opposes the attacker's poisoning goal) honestly participate in federated learning. More precisely, our solution requires that every class defined by the model is represented in the data of at least one honest client. Without these honest clients, no contribution-based defense is possible since the model would be unable to learn anything about these classes in the first place.

Poisoning attacks. In our setting, targeted poisoning attacks are performed by adversaries against the globally trained model. An adversary has a defined poisoning goal: increase the probability of one class being classified as a different, incorrect class, without influencing the output probabilities of any other class. To minimize the possibility of the poisoning attack being detected, the prediction outcomes of classes auxiliary to the attack should be changed minimally.

We assume that poisoning attacks are performed at the class level. Poisoning attacks on specific unseen test examples, such as backdoor attacks [17], [12], [3] assume that few or no honest clients possess data that opposes these attacks, and therefore do not require sybils to succeed. We do not consider these attacks in our solution.

Another possible attack involves sending scaled malicious gradients to overpower honest clients. However, state of the art magnitude-based detection methods exist [9] to prevent these attacks, and we also do not consider them in our solution.

Adversarial sybils perform poisoning attacks on federated learning by providing poisoned updates that direct the shared model towards a poisoned objective, as shown in Figure 2. We do not constrain how a sybil selects these poisoned updates, for example, they can be sourced from poisoned data [6] or computed directly through other methods.

Attacker capability. For a poisoning attack to succeed in a deployment with many honest clients, an attacker must exert more influence on their target class than the total honest influence on this class. This is already a problem for classes that are not well represented in the training data, but in federated learning, where each client is given an equal share of the aggregated gradient, attackers can attack any class with enough influence by generating additional sybils.

If a third party account verification process exists in the system, we assume that the adversary has the means to bypass it, either by creating malicious accounts or by compromising honest clients/accounts. Sybils controlled by multiple sets of non-colluding adversaries may perform poisoning attacks concurrently.

Sybils observe global model state and send any arbitrary gradient contribution to aggregator at any iteration. Sybils can collude by sharing state amongst themselves and by sending updates in an intelligent, coordinated fashion.

IV. Designed Defenses

In the traditional non-IID federated learning setting, the federated learning service only has access to the outputs of local SGD computations. Our solution does not change this feature of federated learning.

A. SGD challenges

From the aggregator's perspective, detecting sybils from a stream of SGD iterations is difficult. If GD, and not SGD, is used as the optimization algorithm, then detection becomes easier: in this setting, gradients are deterministic functions of data and duplicate gradients are easy to detect. The challenge with SGD is three-fold:

Challenge 1. Each client houses a local, unseen non-IID partition of the data that independently may not represent the global learning objective. When receiving a gradient from an individual client, it is difficult for the aggregator to tell whether the objective it points towards is malicious or not.

Challenge 2. Since only a small subset of the original dataset is used in each iteration, the stochastic objective changes with each iteration. The aggregator cannot assume that gradients pointing in sporadic directions are not malicious. As well, the aggregator cannot assume that gradients that are similar came from similar underlying datasets.

Challenge 3. As the batch size b of updates decreases, the variance of gradients contributed by clients increases. At any given iteration, sampling a smaller portion of the dataset results in a higher variance of gradient values, producing the sporadic behavior described above. Since adversaries can send arbitrary gradient updates, we cannot trust that they will conform to a specified batch size configurations.

B. Deficiency of existing defenses

The challenges above make existing poisoning defenses ineffective against *sybil-based attacks*, particularly in non-IID settings.

RONI [27] is ineffective in this setting since, at any given iteration, an honest gradient may update the model in an incorrect direction, resulting in a drop in validation accuracy. This is confounded by the problem that clients may have data that is not accurately modeled by the RONI validation set. For example, we can see in Table I that the prediction accuracy across the other nine digits only dropped slightly when executing a poisoning attack in a non-IID setting. Depending on the RONI threshold, this poisoning attack may go undetected. We further evaluate the effectiveness of RONI in this setting in Section V-B.

Multi-Krum [9] assumes that the number of client adversaries is bounded and known, and given the gradient updates from all clients at each iteration, malicious contributions will appear anomalous. Multi-Krum also assumes the majority of gradient updates from clients point in the correct direction. This is not necessarily true in a non-IID setting and is made worse by SGD: the aggregator will observe a high variance in the gradients even from honest clients.

In our context, attackers can spawn a large number of sybils. This means that assumptions about proportions of honest gradients are unrealistic, and a Multi-Krum majority can be arbitrarily influenced by sybil contributions.

C. FoolsGold design

指示性特征？？还是不理解
；需要看一下代码叭，不太
理解指示性特征是类型2？

Our goal is to desi [illegible] n-IID setting that does not make assumptions about the proportion of honest clients in the system. Instead the metric relies only on information about the nature of targetted poisoning attacks and state from the learning process itself.

Our key insight is that honest clients can be separated from sybils by the diversity of their gradient updates. In the non-IID federated learning setting, each client's training data possess its own particularities, and sybils will contribute gradients that appear more similar to each other than honest clients. We therefore design FoolsGold to value clients that provide unique gradient updates, while reducing the weight of clients that repeatedly contribute similar-looking gradient updates.

With this in mind, FoolsGold has five design goals:

Goal 1. When the system is not being attacked, FoolsGold should preserve the performance of federated learning.

Goal 2. FoolsGold should devalue contributions from clients that point in similar directions.

Goal 3. FoolsGold should be robust to an increasing number of sybils in an attack.

Goal 4. FoolsGold should preserve the gradients of the honest clients that mistakenly appear malicious due to the variance of SGD, even while penalizing the sybil gradients.

Goal 5. FoolsGold should not rely on external assumptions about the clients or require parameterization about the number of attackers.

We now explain the FoolsGold approach (Algorithm 1). In the federated learning protocol, gradient updates are collected and aggregated in synchronous update rounds. FoolsGold reweighs the collected gradients based on (1) the gradient similarity among indicative features in any given iteration, and (2) historical information from past iterations.

Cosine similarity. We use cosine similarity to measure the angular distance between gradients. Sybils can manipulate the magnitude of a gradient to achieve dissimilarity, but the direction of a gradient cannot be manipulated without reducing attack effectiveness. Furthermore, the magnitude of honest gradients can be affected by client-side hyper-parameters such as the learning rate, while the direction of the gradient in SGD is a direct function of the local client objective.

Feature importance. From the perspective of a potential poisoning attack, there are three types of features in the model: (1) features that must be modified for a successful attack, (2) features that are relevant to the correct model, but irrelevant for the attack, and (3) features that are irrelevant to both the attack and the model.

By modifying features of type (1), the adversary mitigates their own attack, and by modifying features of type (2), the adversary impacts the probabilities of other classes, which increases the chance of detection. Thus, an adversary could modify type (3) features, which increases sybil dissimilarity and does not adversely impact the attack.

Similar to other decentralized poisoning defenses [31], we look for similarity only in the indicative features (type 2) in the model. This prevents adversaries from manipulating irrelevant features while performing an attack, which is evaluated in Section 13.

The indicative features are found by measuring the magnitude of model parameters in the global model. Since training data features and gradient updates are normalized while performing SGD, the magnitude of a model parameter maps directly to its influence on the prediction probability [32]. These features can be filtered (hard) or re-weighed (soft) based on their influence on the model, and are normalized across all classes to avoid biasing one class over another.

Updates history. FoolsGold maintains a history of updates from each client. It does this by aggregating the updates at each iteration from a single client into a single aggregated client gradient (line 3). To better estimate similarity of the overall contributions made by clients, FoolsGold computes the similarity between pairwise aggregated historical gradients instead of just the gradients from the current

Figure 3 shows that even for two sybils with a common target objective, gradients at a given iteration may diverge due to the problems mentioned in Challenge 2. However, the cosine similarity between the sybils' aggregated historical gradients is high, satisfying Goal 2.

We interpret the cosine similarity on the indicative features, a value between -1 and 1, as a representation of how strongly two clients are acting as sybils. We define $\max_i(cs)$ as the maximum pairwise similarity for a client i, ensuring that as long as one such interaction exists, we can devalue the contribution robust to the number of sybils, as prescribed by Goal 3.

$$c_{i,t} = \max_j(cosine\_similarity(\sum_{t=1}^{T} \Delta_{i,t}, \sum_{t=1}^{T} \Delta_{j,t}))$$

```
Data: $\Delta_t$ from all clients at iteration $t$
Result: A client weight vector $v$
1  for iteration $t$ do
2      for All clients $i$ do
           // Updates history
3          Let $H_i$ be the aggregate historical vector $\sum_{t=1}^{T} \Delta_{i,t}$
           // Feature importance
4          Let $S_t$ be the weight of indicative features at iteration $t$
5          for All other clients $j$ do
6              Find weighted cosine similarity $cs_{ij}$ between $H_i$ and $H_j$ using $S_t$
7          end
           // Pardoning
8          for All other clients $j$ do
9              if $\max_j(cs) > \max_i(cs)$ then
10                 $cs_{ij}$ *= $\max_i(cs)/\max_j(cs)$
11             end
12         end
13         Let $v_i = 1 - \max_j(cs_i)$
14     end
       // Logit function
15     $v = v/\max(v)$
16     $v = \ln(\frac{v}{(1-v)} + 0.5)$
17     return $v$
18 end
```

Algorithm 1: FoolsGold algorithm.

Pardoning. Since we have weak guarantees on the cosine similarities between an honest client and sybils, honest clients may be incorrectly penalized under this scheme. We introduce a pardoning mechanism that avoids penalizing such honest clients by re-weighing the cosine similarity by the ratio of $\max_i(cs)$ and $\max_j(cs)$ (line 10), satisfying Goal 4.

However, even for very similar gradients, the cosine similarity may be less than one. An attacker may exploit this by increasing the number of sybils to remain influential. We therefore want to encourage a higher divergence for values that are near the two tails of this weight vector, and avoid penalizing honest clients with a low, non-zero similarity value. Thus, we use the logit function (the inverse sigmoid function) centered at 0.5 (line 16), for these properties.

Since we assume at least one client in the system is honest, we rescale the vector such that the maximum value of the weight vector is 1 (line 15). This also encourages the system towards Goal 1: a system containing only honest nodes will not penalize their contributions.

$$c_t = \frac{c_t}{\max(c_t)}$$
$$v_t = \ln(\frac{c_t}{(1-c_t)} + 0.5)$$

When taking the result of the logit function, any value exceeding the 0-1 range is clipped and rounded to its respective boundary value. Finally, the overall gradient update is calculated by applying the final weight vector:

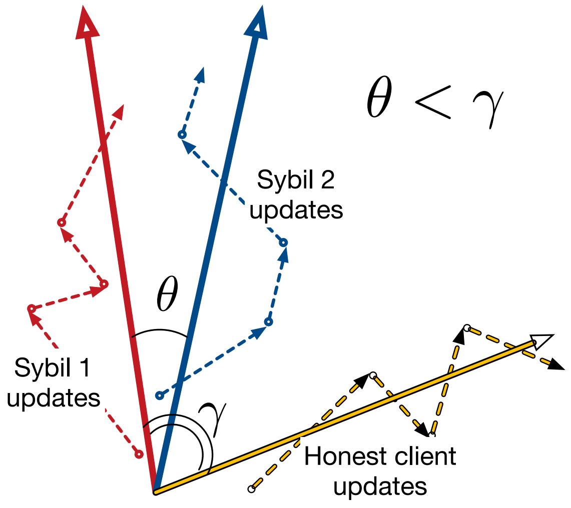


Fig. 3. Dashed lines are gradient updates from three clients (2 sybils, 1 honest). Solid lines are aggregated update vectors. The angle between the aggregated update vectors of sybil clients (θ) is smaller than between those of the honest client and a sybil (γ). Cosine similarity would reflect this similarity.

$$w_{g,t+1} = w_{g,t} + \sum_k v_{k,t}\Delta_{k,t}$$

Note that this design does not require parameterization of expected number of sybils (Goal 5) and is independent of the underlying data features, distribution and SGD details such as batch size.

Augmenting FoolsGold with other methods. Simply re-weighing clients based on their aggregate gradient similarity will not handle all poisoning attacks. Clearly, an attack from a single adversary will not exhibit such similarity properties. FoolsGold is best used when augmented with existing solutions that detect poisoning attacks from a bounded number of attackers (e.g., Robust methods discussed in Section II). We explore the combination of FoolsGold with Multi-Krum in Section V-B.

FoolsGold security guarantees. We claim that our desig mitigates an adversary performing a targeted poisoning attac by limiting the influence they gain through creating sybils in federated learning system. We also claim that FoolsGold satisfies the specified design goals; in particular that it preserves the gradients of honest nodes while penalizing the contributions of sybils. In the next section we empirically validate these claims across several different dimensions using a prototype of FoolsGold.

V. Evaluation

We evaluate FoolsGold by implementing a federated learning prototype in 600 lines of Python. The prototype includes 150 lines for FoolsGold, implementing Algorithm 1. We use scikit-learn [29] to compute cosine similarity of vectors. For each experiment below, we partition the original training data into disjoint non-IID training sets, locally compute SGD

TABLE II. DATASETS USED IN THIS EVALUATION.

Dataset	Examples	Classes	Features
MNIST	60,000	10	784
KDDCup	494,020	23	41
Amazon	1,500	50	10,000

updates on each dataset, and aggregate the gradients using the described method.

We evaluate our prototype on three well-known classification datasets: MNIST [22], a digit classification problem, KDDCup [13], which contains classified network intrusion patterns, and Amazon [13], which contains product review text data. Table II describes the key features of each dataset.

Each dataset was selected for one of its particularities. MNIST was chosen as the baseline dataset for evaluation since it was used extensively in the original federated learning evaluation [23]. The KDDCup dataset has a relatively low number of features, and contains a massive class imbalance: some classes have as few as 5 examples, while some have over 280,000. Lastly, the Amazon dataset is unique in that it has few examples and contains text data: each review is converted into its one hot encoding, resulting in a large feature vector of size 10,000.

For all the experiments in this section, targeted poisoning attacks are performed that attempt to encourage a *source label* to be classified as a *target label* while training on a non-IID federated learning prototype. When dividing the data, each class is always completely represented by a single client, which is consistent with the non-IID federated learning baseline. In all experiments the number of honest clients in the system varies by dataset: 10 for MNIST, 23 for KDDCup, and 50 for Amazon. We consider more IID settings in Section V-D.

We randomly partition a portion of the test data and use it to evaluate two metrics that represent the performance of our algorithm: the *attack rate*, which is the proportion of source labels that are incorrectly classified as the target label, and the *accuracy*, which is the proportion of examples in the test set that are correctly classified.

Both the MNIST and KDDCup datasets were executed with 3,000 iterations and a batch size of 50 unless otherwise stated. For Amazon, due to the high number of features and low number of samples per class, we train for 100 iterations and a batch size of 10. In each of the non-attack scenarios, we ran these experiments to convergence. In all attack scenarios, we found that our selected number of iterations was sufficiently high such that the performance of the attack changed minimally with each iteration, indicating the result of the attack to be consistent. Each reported data point is the average of 5 experiments.

A. Canonical attack scenarios

Our evaluation uses a set of 6 attack scenarios (that we term canonical for this evaluation) across the three datasets (Table III). Attack **A-1** is a traditional poisoning attack: a single client joins the federated learning system with label-flipped data. Attack **A-5** is the same attack performed with 5 sybil clients joining the system. Each client sends gradients for a subset of its data through SGD, meaning that their gradients are not identical. Attacks **A-2x5** and **A-5x5** evaluate FoolsGold's ability to thwart multiple attacks at once. Multiple sets of 5 client sybils attack the system concurrently, and for attack evaluation purposes we assume that the classes in these attacks do not overlap[2].

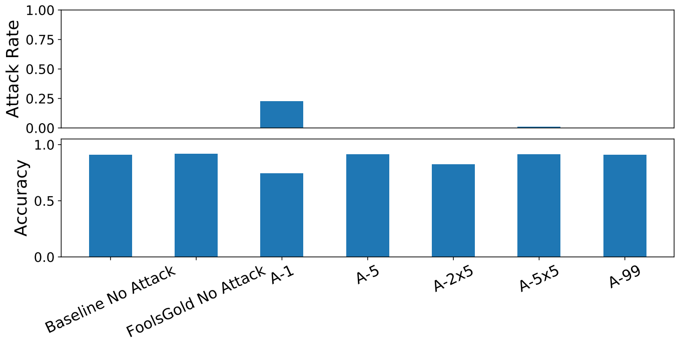


Fig. 4. Training attack rate (top) and accuracy (bottom) for canonical attack against the MNIST dataset.

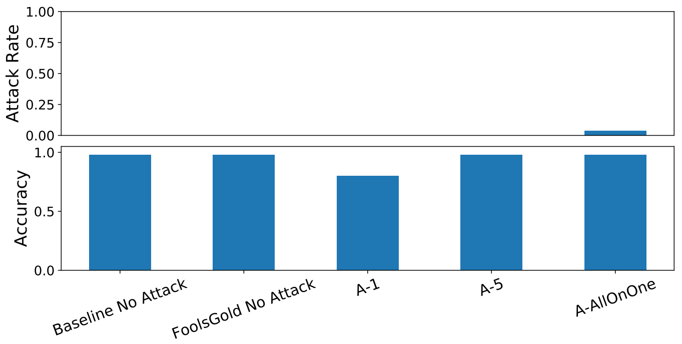


Fig. 5. Training attack rate (top) and accuracy (bottom) for canonical attack against the KDDCup dataset.

Since KDDCup99 is a unique dataset with severe class imbalance, instead of using A-2x5 and A-5x5 we choose to perform a different attack, **A-AllOnOne**, on this dataset. In KDDCup99, data from various network traffic patterns are provided. Class "Normal" identifies patterns without any network attack, and is proportionally large (approximately 20% of the data) in the overall dataset. Therefore, when attacking KDDCup99, we assume that adversaries mis-label malicious attack patterns, which are proportionally small, (approximately 2% of the data) and poison the malicious class towards the "Normal" class. A-AllOnOne is a unique attack for KDDCup in which 5 different malicious patterns are each labeled as "Normal", and each attack is performed concurrently.

Finally, we use **A-99** to illustrate the robustness of FoolsGold to massive attack scenarios. In this attack, an adversary generates 990 sybils to overpower a network of 10 honest clients and all of them attempt a single 1-7 attack against MNIST.

Since we use these canonical attacks throughout this work, we first evaluate each attack on the respective datasets (Table III). Figures 4, 5, and 6 plot the attack rate and test accuracy

[2] We do not perform a 1-2 attack in parallel with a 2-3 attack, since evaluating the 1-2 attack would be biased by the performance of the 2-3 attack.

TABLE III. CANONICAL ATTACKS USED IN OUR EVALUATION.

Attack	Description	Dataset
A-1	Single client attack.	All
A-5	5 clients attack.	All
A-2x5	2 sets of 5 clients, concurrent attacks.	MNIST, Amazon
A-5x5	5 sets of 5 clients, concurrent attacks.	MNIST, Amazon
A-AllOnOne	5 clients executing 5 attacks on the same target class.	KDDCup99
A-99	99% adversarial clients, performing the same attack.	MNIST

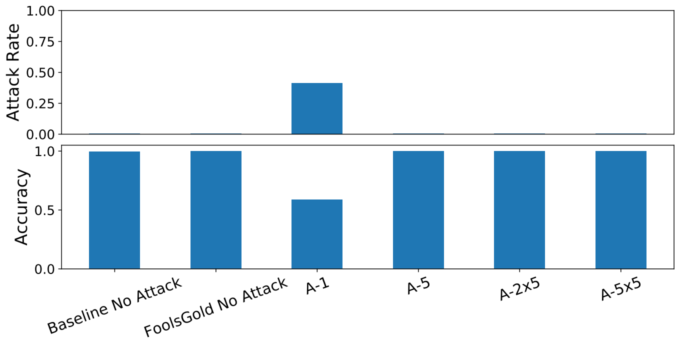


Fig. 6. Training attack rate (top) and accuracy (bottom) for canonical attack against the Amazon dataset.

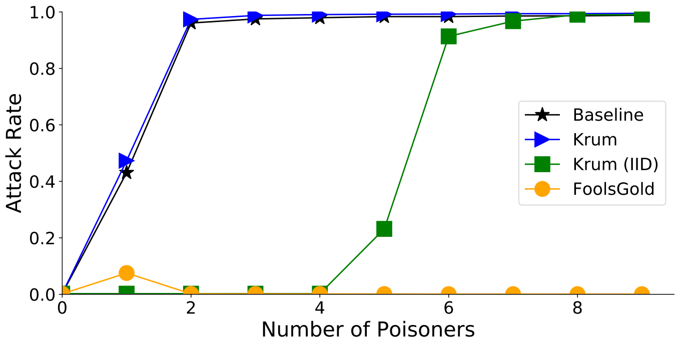


Fig. 7. Attack rate for varying number of sybils, for federated learning (Baseline), Multi-Krum (non-IID and IID), and FoolsGold.

for each of the attacks in Table III. Each Figure also plots results for the system without attacks: the original federated learning algorithm (Baseline No Attack) and the system with the FoolsGold algorithm (FoolsGold No Attack).

For most attacks, including the A-AllOnOne attack and the A-99 attack, FoolsGold effectively prevents the attack while maintaining high training accuracy. As FoolsGold faces larger groups of sybils, it has more information to more reliably detect similarity between sybils. FoolsGold performs worst on the A-1 attacks in which only one adversarial client attacked the system. The reason is simple: without multiple colluding sybils, adversaries and honest clients are indistinguishable to the FoolsGold aggregator.

Another point of interest is the prevalence of false positives. In A-1 KDDCup, our system incorrectly penalized an honest client for colluding with the attacker, lowering the prediction rate of the honest client's as the defense was applied. We observe that the two primary reasons for a decreased training error are either a high attack rate (false negatives) or a high target class error rate (false positives). We also discuss false positives from data similarity in Section V-D

B. Comparison to prior work

We compare FoolsGold to two existing techniques: Multi-Krum aggregation, and RONI (see Section II).

Comparison to Multi-Krum. In this experiment we compare FoolsGold to Multi-Krum and an unmodified federated learning as a baseline as we vary the number of sybils.

We implemented Multi-Krum as specified in the original paper [9]: at each iteration, the total Euclidean distance from the $n-f-2$ nearest neighbors is calculated, and the average of these updates is calculated. Multi-Krum relies on the f parameter: the maximum number of Byzantine clients tolerated by the system. Prior knowledge of this parameter is an unrealistic assumption in our setting and we arbitrarily set $f = 2$ in our evaluation.

While running Multi-Krum, we found that the performance was poor in the non-IID setting. One client is always removed in each iteration, and if this client was an honest one, it resulted in the complete removal of updates for an entire class. This resulted in both a low final training accuracy and a high attack rate, as the adversary contributions were not detected by Multi-Krum. Therefore, we also include the performance of Multi-Krum while training an MNIST classifier with 10 honest clients and a varying number of attackers. Each honest client holds uniformly sampled MNIST data to evaluate Multi-Krum in its intended IID setting.

Figure 7 shows the performance of the four approaches against an increasing number of poisoners: a 1-7 attack is performed on an unmodified non-IID federated learning system (Baseline), a federated learning system with Multi-Krum aggregation (both non-IID and IID), and our proposed solution.

We see that as soon as the proportion of poisoners for a single class increases beyond the corresponding number of honest clients that hold that class (which is 1), the attack rate increases significantly for naive averaging (Baseline).

In addition to exceeding the parameterized number of expected adversaries, an adversary can also influence the mean client contribution at any given iteration, and Multi-Krum will fail to distinguish between honest clients and sybils.

Multi-Krum works with up to 33% adversaries [9], but fails above this threshold. By contrast, FoolsGold penalizes attackers further as the proportion of sybils increases, and in this scenario FoolsGold remains robust even with 9 attackers.

Consistent with the results in Figures 4 – 6, FoolsGold in Figure 7 performs the worst when only one poisoner is present.

Comparison to RONI. RONI has not been applied to federated learning before. To extend RONI to a federated learning

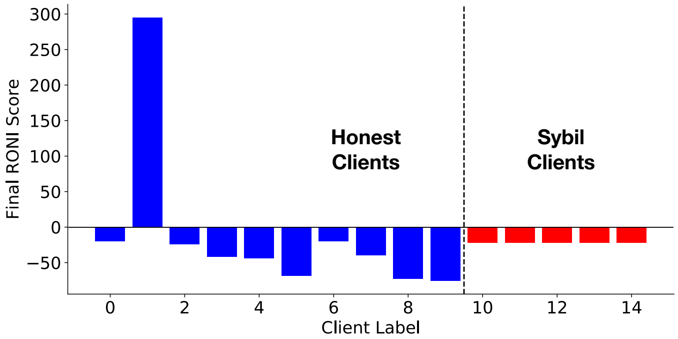


Fig. 8. The final sum of the RONI validation scores for the A-5 attack executed on federated learning: RONI is unable to distinguish the malicious clients (5 rightmost) from most of the honest ones (the first 10).

setting, the server can capture the influence of a single gradient update, rather than an entire dataset. This update can be validated by checking if a model's accuracy improves after a single update, and rejecting malicious updates if they degrade the model performance beyond a specified threshold. However, this still implies that the aggregator has access to a true validation dataset, which typically does not exist.

We show that this approach, with a RONI dataset that contains a uniform distribution of data, is insufficient in countering sybil-based poisoning.

We train an MNIST classifier and use a 10,000 example IID RONI validation set. We perform an A-5 1-7 attack on the system and log the total RONI validation score across all 15 clients (10 honest and 5 sybils) for 3,000 iterations. Figure 8 shows the total sum of the RONI score across all iterations (the 5 right-most clients in the Figure are sybils). A RONI score below 0 indicates rejection. Figure 8 shows that all clients except the honest client with the digit ”1” had received a negative score in every iteration.

In the non-IID setting, clients send updates that do not represent an update from the global data distribution. Validating individual updates from a single client in such a fashion produces false positives because the aggregator holds a validation set that contains uniform samples, and RONI flags each of their gradients as malicious for doing poorly on the validation set.

Without prior knowledge of the details of a potential attack, RONI is unable to distinguish between gradient updates coming from sybils and gradient updates coming from honest non-IID clients.

C. Attack generalization

Thus far we have used the 1-7 attack on MNIST as our running example. However, the attacker could be more successful in attacking a different source and target class. We evaluate FoolsGold's ability to generalize to other targeted poisoning attacks against MNIST (i.e., all other possible source and target MNIST labels)[3]. For each possible source-target combination, we execute the A-5 attack scenario against FoolsGold.

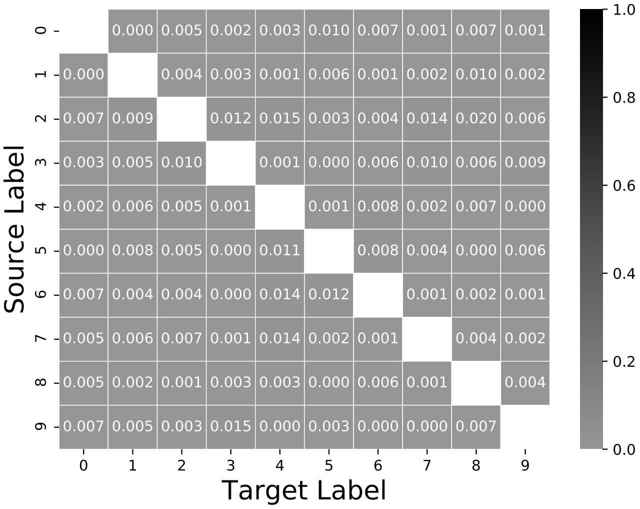


Fig. 9. The attack rates for the A-5 attack executed for all combinations: attempting to mis-label a true (source) 0-9 class to a different (target) 0-9 class.

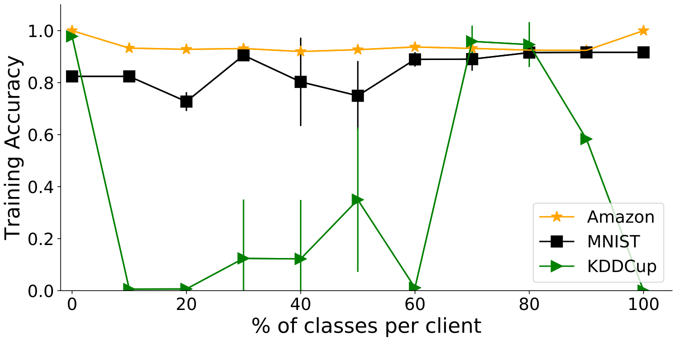


Fig. 10. The training accuracies of the model when defending the A-5 attack, under varying degrees of honest client class label overlap.

Figure 9 shows the resulting attack rate for our defense across all permutations of the A-5 MNIST poisoning attacks. The highest rate in this table is 0.02, with consistently low rates across the table; this indicates that our defense generalizes to other class-based poisoning attacks within MNIST.

D. Non-uniformity of honest data

FoolsGold relies on the assumption that training data is sufficiently dissimilar between clients. We evaluate the limitations of this assumption by increasing the degree to which client training datasets overlaps.

To demonstrate this, we executed the A-5 attack on all datasets, varying only the proportion of classes shared among clients. For each proportion of overlap k, each client held the data for its own class and the data of the next k client classes. This resulted in client datasets that overlap in class labels. We varied the proportion of classes held by each client in 10% increments from 0% (full non-IID setting, each client has 1 class) to 100% (full IID setting).

Figure 10 shows the training accuracy with standard deviation error bars for all three datasets with varying levels of class overlap. In all cases except 60% and 80% overlap on KDDCup (where both the training accuracy and attack rate were high),

[3] Due to space constraints we only report the results for MNIST.

the attack was stopped by FoolsGold, with a near-zero attack rate. As expected (and shown by our baselines), all datasets perform well when each client only possesses a single class. When the proportion of overlap increases, the performance of each dataset drops, especially KDDCup, which becomes highly variable. This is caused by false positives: when two clients with data from the same class send gradient updates, these updates will tend to appear more similar.

When each client has 100% of the data, the performance of FoolsGold was remarkably good for Amazon and MNIST, while being very poor for KDDCup.

In general, the Amazon and MNIST dataset was more robust to changes in class overlap than KDDCup, for which the performance was very sporadic. We believe that this is a direct function of the number of features in each dataset: KDDcup has very few (41), while MNIST (784) and Amazon have more (10,000). With a higher number of features, the variance between elements of the same class is higher, and it is more likely that two gradients coming from data in the same class will not appear as similar.

In FoolsGold, a model with a higher number of features is more robust to false positives from workloads that contain more IID data, challenging the non-IID assumption of FoolsGold.

E. What if the attacker knows FoolsGold?

If an attacker is aware of the FoolsGold algorithm, they may attempt to send gradients in ways that encourage additional dissimilarity. This is an active trade-off: as attacker gradients become less similar to each other (lower chance of detection), they become less focused towards the poisoning objective (lower attack utility).

We consider and evaluate four ways in which attackers may attempt to subvert FoolsGold: (1) mixing malicious and correct data, (2) changing sybils' training batch size, (3) perturbing contributed updates with noise, and (4) infrequently and adaptively sending poisoned updates.

Resilience to mixed data. Attackers may attempt to subvert our design by creating a dataset with a mixture of honest and poisoned data. This provides an opportunity for sybils to appear honest, as they are less similar when executing SGD. Note that in this case the attacker purposefully mitigates their attack by labeling a proportion of the poisoned dataset *correctly*.

To evaluate this attack, we created datasets with 20%, 40%, 60% and 80% poisoned data for an MNIST 1-7 attack, and the remaining proportion of the data with honestly labeled data. We also perform a second experiment in which the attacker mixes their data with poisoned data for a second 4-9 attack, effectively performing a mixture of two targeted poisoning datasets simultaneously. Both attacks are mounted using the A-5 attack strategy against FoolsGold.

Figure 11 plots the attack rate against the proportion of mixing. This same attack was attempted with the KDDCup dataset and the Amazon dataset, for which all attack rates are 0; we omit these results from the Figure. We observe that FoolsGold is robust to this strategy: the amount of data mixing does not impact FoolsGold's ability to mitigate sybils, when choosing any proportion between 20% and 80%, the maximum average attack rate observed is 0.003.

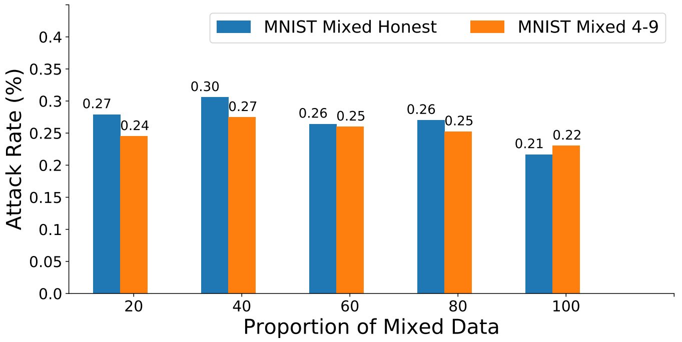


Fig. 11. An adversary cannot subvert FoolsGold by including a proportion of different honest or attack data.

At any individual iteration, sybils are more likely to appear as honest clients. However, as the number of iterations increases, the average will tend to the same objective, which FoolsGold is able to detect through its use of history. The effect of this mechanism is explored more in Section V-F.

Resilience to different batch sizes. Another factor that impacts FoolsGold's effectiveness is the client batch size. Given that variance in SGD decreases with more data points, we expect FoolsGold to perform worse with smaller batch sizes.

To evaluate different batch sizes, we performed A-5 attacks on all three datasets. In each case, all clients and sybils perform SGD with the same batch size for values of 1, 5, 10, 20, 50, and 100. Since no Amazon data partition contains over 50 examples, we do not evaluate Amazon with batch sizes of 50 and 100.

Figure 12 shows the attack rate as the batch size increases. It shows that FoolsGold is resilient to the batch size for A-5 attacks performed on MNIST and KDDCup, achieving an attack rate at or near 0.

The only instance in which performance of the system suffered was for the A-5 Amazon attack with the batch size set to 1; the resulting attack rate was 4.76%. This is due to the curse of dimensionality: there is a higher variance in the similarities between gradients when the dimension size is large (10,000). This variance is maximal when the batch size is lowest.

Adding intelligent noise to gradients. Assuming a set of even more intelligent sybils, adversaries could send pairs of gradients with carefully perturbed noise that is designed to sum to zero. For example, if an attacker draws a random noise vector η, two adversarial gradients a_1 and a_2 could be contributed instead as v_1 and v_2 in such a fashion:

$$v_1 = a_1 + \eta$$
$$v_2 = a_2 - \eta$$

Since the noise vector η has nothing to do with the poisoning objective, its inclusion will add dissimilarity to the adversarial gradients and decrease FoolsGold's effectiveness in detecting them. Also note that the sum of these two gradients is still

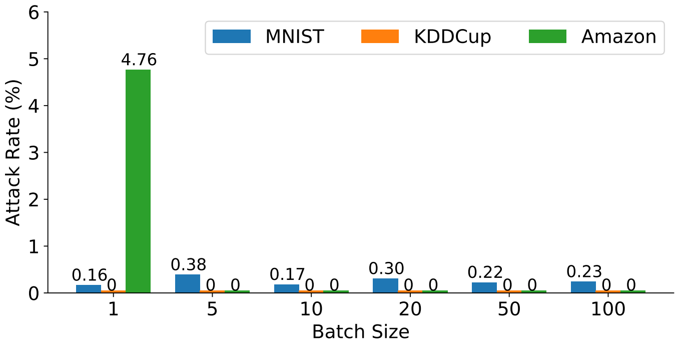


Fig. 12. The performance of the A-5 attack on all datasets with increasing batch size.

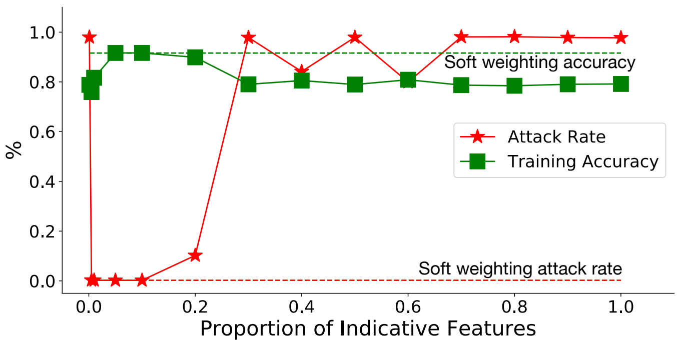


Fig. 13. The performance of the optimal noisy attack on MNIST with varying indicative feature ratio.

the same: $v_1 + v_2 = a_1 + a_2$. This strategy can also be scaled beyond 2 sybils by taking orthogonal noise vectors and their negation: for any subset of these vectors, the cosine similarity is 0 or -1, while the sum remains 0.

This attack is mitigated by filtering for indicative features in the model. Instead of looking at the cosine similarity between gradient contributions across all features in the model, we reweigh features based on feature importance, as described in Section IV.

To evaluate the importance of indicative features to the poisoning attack, we execute the intelligent noise attack described above on MNIST: a pair of sybils send v_1 and v_2 with intelligent noise η. We then vary the proportion of model parameters that are defined as indicative from 0.001 (8 features on MNIST) to 1 (all features).

Figure 13 shows the attack rate and the training accuracy for varying proportions of indicative features. We first observe that when using all of the features for similarity (far right), the poisoning attack is successful.

Once the proportion of indicative features decreases beyond 0.1 (10%), the dissimilarity caused by the intelligent noise is removed from the cosine similarity and the poisoning vector dominates the similarity, causing the intelligent noise strategy to fail with an attack rate of near 0. We also observe that if the proportion of indicative features is too low (0.01), the training accuracy also begins to suffer. When considering such a low number of features, honest clients appear to collude as well, causing false positives.

We also evaluated the soft feature weighing mechanism, which weighs each contribution proportionally based on the model parameter itself. The results of the soft weighting method on the same intelligent MNIST poisoning attack are also shown in Figure 13. For both the attack rate and training accuracy, the performance of the soft mechanism is comparable to the optimal hard filtering mechanism.

Adaptive gradient method. We devised another optimal attack against FoolsGold that manipulates the memory component of FoolsGold. If an adversary knows that FoolsGold employs cosine similarity on the gradient history, and is able to locally compute the pairwise cosine similarity amongst sybils, they can bookkeep this information and decide to send poisoned gradients only when their similarity is sufficiently low (Algorithm 2). This algorithm uses a parameter M, representing the inter-sybil cosine similarity threshold. When M is low, sybils are less likely to be detected by FoolsGold as they will send their gradients less often; however, this will also lower the influence each sybil has on the global model.

```
Data: Desired $\Delta_t$ from all sybils at iteration $t$,
      similarity threshold $M$
for iteration $t$ do
    for All sybils $i$ do
        // Compute local history
        Let $H_i$ be the aggregate historical vector
          $\sum_{t=1}^{T} \Delta_{i,t}$
        // Feature Importance and similarity
        Let $S_t$ be the set of indicative features at
          iteration $t$ (if known)
        for All other sybils $j$ do
            Find cosine similarity $cs_{ij}$ between $H_i$ and
              $H_j$ using features in $S_t$
        end
        // Adaptive decision
        if $\max_i(cs) > M$ then
            Send intelligent noise vector $\eta$ to server
        end
        else
            Send $\Delta_i$ to server
        end
    end
end
```

Algorithm 2: Adaptive attack on FoolsGold.

An adversary could generate an exceedingly large number of sybils for a successful attack, but given that the adversary is uncertain about the required influence needed to overpower the honest clients, this becomes a difficult trade-off to navigate for an optimal attack.

To demonstrate this, the intelligent noise attack described above is executed by 2 sybils on MNIST, with FoolsGold using the soft weighing of features in its cosine similarity (the optimal defense for MNIST against the intelligent noise attack). Figure 14 shows the relationship between M and the resulting expected ratio of sybils needed to match the influence for each honest opposing client.

For instance, if we observed that the sybils only sent poisoning gradients 25% of the time, they would need 4 sybils. Given a prescribed similarity threshold M, the values shown

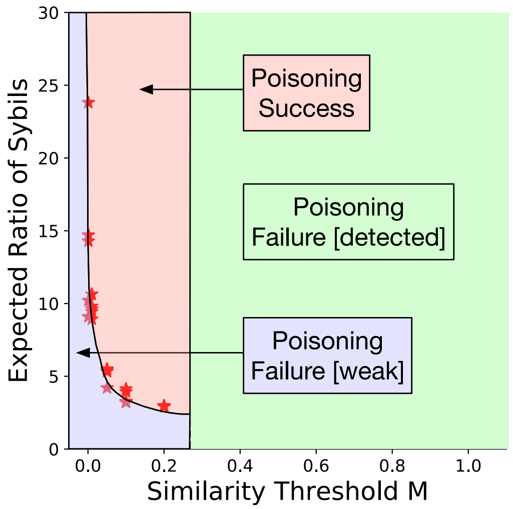


Fig. 14. Relationship between similarity threshold and expected ratio of sybils per honest opposing client for the adaptive attack on FoolsGold (Algorithm 14) with 2 sybils on MNIST

are the expected number of sybils required for the optimal attack. The attack is optimal because using less sybils does not provide enough influence to poison the model, while using more sybils is inefficient.

This is shown on Figure 14 by the three shaded regions: in the green region to the right ($M > 0.27$), the threshold is too high and any poisoning attack is detected and removed. In the blue region on the bottom left, the attack is not detected, but there is an insufficient number of sybils to overpower the honest opposing clients. Lastly, in the top left red region, the attack succeeds, potentially with more sybils than required.

With a sufficiently large number of sybils and appropriately low threshold, attackers can subvert our current defense for our observed datasets. Finding the appropriate threshold is challenging as it is dependent on many other factors: the number of honest clients in the system, the proportion of indicative features considered by FoolsGold, and the distribution of data.

Furthermore, this attack requires a higher proportion of sybils than the baseline poisoning attack on federated learning. For example, when M is set to 0.01, an attacker would require a minimum of 10 sybils per opposing client to poison the model, whereas in federated learning, they would only need to exceed the number of opposing clients. The exact number of sybils required to successfully poison the model is unknown to attackers without knowledge of the number of honest clients and their honest training data.

F. Effects of design elements

Each of the three main design elements (history, pardoning and logit) described in Section IV addresses specific challenges. In the following experiments we disabled one of the three components and recorded the training error, attack rate, and target class error of the resulting model.

History. The two subversion strategies mentioned in the previous section increase the variance of gradients in each iteration. The increased variance in the gradients sent by sybils cause the cosine similarities at each iteration to be an inaccurate approximation of a client's sybil likelihood. Our design uses history to address this issue, and we evaluate this claim by comparing the performance of FoolsGold with and without history using an A-5 MNIST attack with 80% poisoned data and batch size of 1 (two factors which were previously shown to have a high variance).

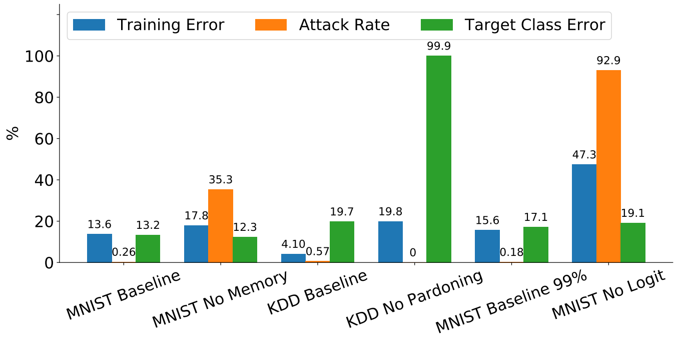


Fig. 15. Metrics for FoolsGold with various components independently removed.

Pardoning. We claim that the gradients of honest clients may be similar to the gradients of sybils, especially if the honest client owns the data for the targeted class. To evaluate the necessity and efficacy of our proposed pardoning system, we compare the performance of FoolsGold on KDDCup with the A-AllOnOne attack with and without pardoning.

Logit. An important motivation for using the logit function is that adversaries can arbitrarily increase the number of sybils to mitigate any non-zero weighting of their gradients. We evaluate the performance of FoolsGold with and without the logit function for the A-99 MNIST attack.

Figure 15 shows the overall training error, sybil attack rate, and target class error for the six different evaluations. The attack rate for the A-AllOnOne KDDCup attack is the average attack rate for the 5 sets of sybils.

Overall, the results align with our claims. Comparing the A-5 MNIST case with and without history, we find that history successfully mitigates attacks that otherwise would pass through in the no-history system. Comparing the results of the A-AllOnOne KDDCup attack, we find that, without pardoning, the training error and target class error increase while the attack rate was negligible for both cases, indicating a high rate of false positives for the target class. Finally, comparing the results for the A-99 MNIST attack, without the logit function, the adversary was able to mount a successful attack by overwhelming FoolsGold with sybils, showing that the logit function is necessary to prevent this attack.

G. FoolsGold performance overhead

We evaluate the runtime overhead incurred by augmenting a federated learning system with FoolsGold. We run the system with and without FoolsGold with 10 – 50 clients by training an MNIST classifier in a non-IID setting for 3,000 iterations.

Figure 16 plots execution time of the system with and without FoolsGold. FoolsGold scales worse that the baseline federated learning algorithm. The most expensive part of the FoolsGold algorithm is computing the pairwise cosine

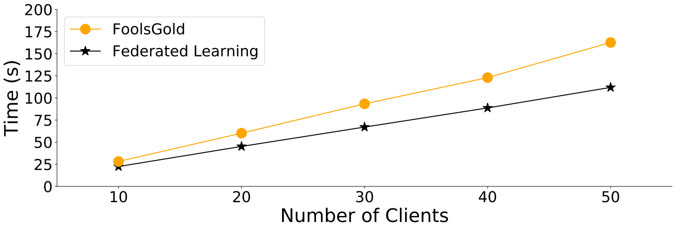


Fig. 16. Runtime overhead of FoolsGold compared to Federated Learning, training MNIST for 3,000 iterations.

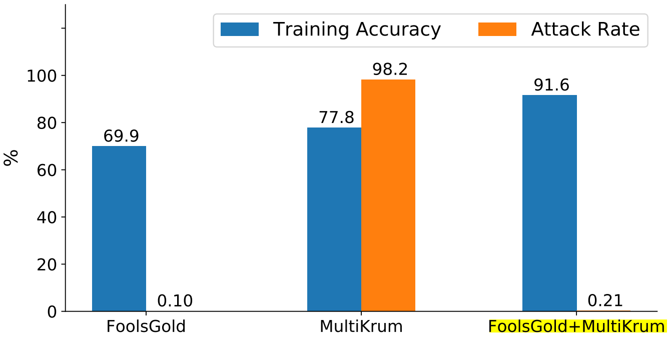


Fig. 17. The performance of Multi-Krum with FoolsGold when combating the optimal strawman attack on MNIST.

similarity. Our Python prototype is not optimized and there are known optimizations to improve the speed of computing angular distance at scale [1].

VI. Limitations

Combating a single client adversary. FoolsGold is designed to counter *sybil-based attacks* and our results in Figure 7 indicate that FoolsGold is not successful at mitigating attacks mounted by a single poisoning client.

However, we note that a single malicious actor could be detected and removed by Multi-Krum [9]. Although Multi-Krum is not designed for and does poorly in the non-IID setting, we performed another experiment in which FoolsGold was augmented with a properly parameterized Multi-Krum solution, where $f = 1$.

We propose an ideal strawman attack that involves a single adversarial client that sends the vector to the poisoning objective at each iteration. This attack does not require sybils and can therefore bypass FoolsGold. Figure 17 shows the training accuracy and the attack rate for FoolsGold, Multi-Krum, and the two systems combined when facing a concurrent A-5 attack and the ideal strawman attack.

When Multi-Krum uses $f = 1$ and FoolsGold is enabled, we see that Multi-Krum and FoolsGold do not interfere with each other. The Multi-Krum algorithm prevents the strawman attack, and FoolsGold prevents the sybil attack. Independently, these two systems fail to defend both attacks concurrently, either by failing to detect the ideal strawman attack (against FoolsGold) or by allowing the sybils to overpower the system (against Multi-Krum).

FoolsGold is specifically designed for handling poisoning attacks from a group of sybils: we believe current state of the art is better suited to mitigate attacks from single actors.

Dependence on honest client data distribution. Using the non-IID setting of federated learning allows us to assume that similar gradient updates are more likely to be malicious. If two honest clients possess a high degree of similar data, they would be penalized by FoolsGold. We observed this in some of our canonical experiments in Table III and our non-uniformity experiments in Section V-D. This effect is highly dependent on the learning task itself, and cases with a sufficiently high dimensionality and intra-class variance are less affected by client data distribution.

However, FoolsGold is designed for the federated learning multi-party non-IID setting where data is *not* shared between clients. This is a setting that is motivated by the variance between client training sets, and one of the key contributions of federated learning [23].

Generalization to other datasets. Even within the non-IID setting, the performance of FoolsGold is highly data dependent on the size, distribution and dimensionality of the underlying data. Knowing this, we attempted to capture a wide range of dataset types and sizes; however, further work is necessary to check that FoolsGold generalizes to other types and distributions of data.

VII. Discussion

Convergence properties. Our experimental results exhibit that convergence is achievable by FoolsGold for various data distributions and attack scenarios. FoolsGold reweighs clients in federated learning and is similar to importance sampling, which has been applied to SGD algorithms [26]. In importance sampling, data points are reweighed in a scheme that reduces the variance of the learning algorithm, providing faster convergence.

This is not identical to our setting, since we assume the presence of malicious datasets that oppose the global learning objective. If we assume that no honest clients are flagged as malicious by FoolsGold, contributions from each malicious client will be removed and each honest client will be unmodified, which reduces to the general SGD case, for which convergence properties are clearly defined [11]. We leave the formal convergence properties of a system that builds upon FoolsGold's algorithm as future work.

Improving FoolsGold against informed attacks. In Figure 14, we observe that FoolsGold can be subverted by a knowledgeable adversary with sufficiently many sybils. We believe that that non-determinism may further help to improve FoolsGold. This may involve using a weighted random subset of gradients in the history mechanism or by taking similarity across random subsets of client contributions. With more randomness, it becomes difficult for intelligent adversaries to use knowledge about FoolsGold's design to their advantage.

Another potential solution could involve removing client contributions with a more intelligent similarity metric. This may involve incorporating graph-based similarity, using auxiliary information from an initial bootstrap dataset or mandating a minimum similarity score to reject exceedingly anomalous contributions. While more informed, these solutions require stronger security assumptions about the nature of poisoning attacks on federated learning, for which more work needs to be done.

Extending to deep learning. Based on the results seen in Section V-D, FoolsGold is more robust to false positives when the number of features is high. There is a clear workload that often exhibits this property and is widely used (and attacked) in research today: deep neural networks.

Most deep learning classification architectures use the softmax loss in the output layer, and FoolsGold could be applied as-is on the final layer of a deep network to detect sybils in deep learning.

However, this would not make use of information stored in the deeper layers of the model structure. The interactions between features in deep networks are not as clear as in softmax and more work is necessary to better understand the nature of poisoning attacks in deep learning and how gradient similarity metrics can be defined and applied against these attacks.

VIII. Related work

We reviewed closely related work in Section II, including RONI [5] and Multi-Krum [9]. Here we review the broader literature on secure ML, sybil defenses, and poisoning defenses.

Secure ML. An alternative solution to mitigating sybils in federated learning would be to make the overall process more secure. An alternate secure ML computation model, which relies on trusted execution environments, such as SGX, was proposed by Ohrimenko et al. [28]. This approach requires the use of specialized hardware to ensure that ML performed on local data is trusted, and no sybils can be created. However, we note that these sybil-based poisoning attacks can all performed at the data input level, which is not secured by SGX. Even in a secure enclave running trusted code, a poisoning attack can be performed by supplying malicious data. FoolsGold can be augmented to systems based on SGX to prevent sybil attacks.

Another work proposed by Baracaldo et al. [4] employs data provenance to prevent poisoning attacks. Their solution uses provenance features to segment data points, identifying and removing data points from the model that are likely to be malicious. This solution requires additional assumptions about how training data is collected and in federated learning, where thousands of clients supply data from a variety of sources, this assumption is unrealistic.

Sybil defenses. One approach to mitigate sybil attacks is to use proof of work [2], in which a joining client must solve a computationally expensive problem (that is easy to check) to contribute to the system. More recent alternatives have explored the use of proof of stake [16], which weighs clients by the value of their *stake* in the system.

Some approaches are not only robust to sybil attacks, but actively detect and remove sybils. These approaches use auxiliary interaction information such as an underlying social network [33], or involve detecting and rejecting malicious behavior [34]. Many sybil detection problems involve mapping the interactions between clients into a weighted graph and using sybil defenses from social networks [36] as a reduction. However, federated learning limits the amount of information exposed to the central service, and these defenses may rely on privacy-compromising information, making them infeasible in the federated learning setting. In FoolsGold, no auxiliary information outside of the central learning process is required to defend against sybil-based attacks.

Other ML poisoning defenses. Another poisoning defense involves bagging classifiers [7] to mitigate the effects of outliers in the system. These defenses are effective in scenarios where the dataset is centralized, but are more complicated to apply in a federated learning setting, where access to the data is prohibited. This algorithm also assumes control of the gradient computation while training models, which federated learning does not have access to.

AUROR [31] is a defense designed for the multi-party ML setting. This defense defines indicative features to be the most important model features, and a distribution of modifications to them is collected and fed to a clustering algorithm. Contributions that come from small clusters that exceed a threshold distance are removed. This clustering algorithm assumes that the majority of updates to that feature come from honest clients for all indicative features. The idea of using indicative features was inspired by AUROR, but unlike AUROR we assume the presence of sybils and rely on gradient similarity to detect anomalous clients.

We believe that in the open admission model of federated learning, where clients are free to join and leave the learning process at any time, making assumptions about the total proportion of data and total proportion of honest users is overly optimistic.

IX. Conclusion

The decentralization of ML procedures is driven by growing privacy concerns and scalability challenges. Federated learning is a state of the art proposal that has been adopted in production [24]. However, such decentralization opens the door for malicious clients to participate in training.

We considered the problem of poisoning attacks by malicious clients that use sybil nodes to achieve their poisoning objective. We show that existing defenses to poisoning in the federated learning context, such as Multi-Krum, are ineffective when sybils are added and propose *FoolsGold*, a defense that weighs clients by their *contribution similarity*.

Our evaluation results across three different datasets indicate that FoolsGold can mitigate a variety of attack types and is effective even when sybils overwhelm honest users. We also considered advanced attack types in which sybils mix poisoned data with honest data, add intelligent noise to their updates, and adaptively rate limit their poisoned updates to avoid detection.

Across all scenarios, our defense is able to outperform prior work and requires substantially higher number of sybils to defeat honest opposing clients.

The defense minimally changes the federated learning algorithm, relies on standard techniques such as cosine similarity, and does not require prior knowledge of the expected number of sybils. We hope that our work inspires further research on defenses that are co-designed with the underlying learning procedure.

References

[1] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance," in *Advances in Neural Information Processing Systems 28*, ser. NIPS, 2015.

[2] A. Back, "Hashcash - A Denial of Service Counter-Measure," Tech. Rep., 2002.

[3] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How To Backdoor Federated Learning," *ArXiv e-prints*, 2018.

[4] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: A data provenance based approach," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec, 2017.

[5] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The Security of Machine Learning," *Maching Learning*, vol. 81, no. 2, 2010.

[6] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proceedings of the Asian Conference on Machine Learning*, 2011.

[7] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Multiple Classifier Systems*, 2011.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning Attacks Against Support Vector Machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012.

[9] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30*, ser. NIPS, 2017.

[10] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS, 2017.

[11] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *19th International Conference on Computational Statistics*, ser. COMPSTAT, 2010.

[12] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *ArXiv e-prints*, 2017.

[13] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[14] J. J. Douceur, "The sybil attack," in *IPTPS*, 2002.

[15] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *NIPS Workshop: Machine Learning on the Phone and other Consumer Devices*, 2017.

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP, 2017.

[17] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain," *ArXiv e-prints*, 2017.

[18] B. Han, I. W. Tsang, and L. Chen, "On the convergence of a family of robust losses for stochastic gradient descent," in *European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 9851*, ser. ECML PKDD, 2016.

[19] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[20] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'17, 2017.

[21] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial Machine Learning," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, 2011.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[23] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[24] H. B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," https://research.googleblog.com/2017/04/federated-learning-collaborative.html, 2017, accessed: 2017-10-12.

[25] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare," in *IEEE Journal of Biomedical and Health Informatics*, 2015.

[26] D. Needell, R. Ward, and N. Srebro, "Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm," in *Advances in Neural Information Processing Systems 27*, 2014.

[27] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting Machine Learning to Subvert Your Spam Filter," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[28] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious Multi-Party Machine Learning on Trusted Processors," in *USENIX SEC*, 2016.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[30] M. S. Riazi, B. D. Rouhani, and F. Koushanfar, "Deep Learning on Private Data," *IEEE Security and Privacy Magazine*, 2018.

[31] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ser. ACSAC, 2016.

[32] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS, 2015.

[33] N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI, 2009.

[34] B. Viswanath, M. A. Bashir, M. B. Zafar, S. Bouget, S. Guha, K. P. Gummadi, A. Kate, and A. Mislove, "Strength in numbers: Robust tamper detection in crowd computations," in *Proceedings of the 3rd ACM Conference on Online Social Networks*, ser. COSN, 2015.

[35] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Defending against sybil devices in crowdsourced mapping services," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys, 2016.

[36] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," *IEEE/ACM Transactions on Networking*, 2010.

[37] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: Defending Against Sybil Attacks via Social Networks," in *SIGCOMM*, 2006.