SparseFed: Mitigating Model Poisoning Attacks in Federated Learning with Sparsification

Ashwinee Panda and **Saeed Mahloujifar**
and Arjun N. Bhagoji and Supriyo Chakraborty and Prateek Mittal
Princeton University University of Chicago IBM

Abstract

Federated learning is inherently vulnerable to model poisoning attacks because its decentralized nature allows attackers to participate with compromised devices. In model poisoning attacks, the attacker reduces the model's performance on targeted sub-tasks (e.g. classifying planes as birds) by uploading "poisoned" updates. In this report we introduce `SparseFed`, a novel defense that uses global top-k update sparsification and device-level gradient clipping to mitigate model poisoning attacks. We propose a theoretical framework for analyzing the robustness of defenses against poisoning attacks, and provide robustness and convergence analysis of our algorithm. To validate its empirical efficacy we conduct an open-source evaluation at scale across multiple benchmark datasets for computer vision and federated learning.

1 INTRODUCTION

The federated learning paradigm enables training models across consumer devices without aggregating data, but deployed systems are not robust to model poisoning attacks (Wang et al., 2020a; Bhagoji et al., 2019; Bagdasaryan et al., 2020). There are two main settings for federated learning: the cross-device setting and the cross-silo setting (Kairouz et al., 2019). In the cross-device setting, the goal is to train a model across disjoint data distributed across many thousands of devices (Kairouz et al., 2019). In the cross-silo setting, data distributions are less extreme and fewer devices participate (Kairouz et al., 2019). Compromised devices are easily able to participate in federated learning (Bonawitz et al., 2019) and the models trained are often redeployed to serve millions or billions of requests (Hard et al., 2018). Attackers often have an incentive to compromise the behavior of trained models (Bhagoji et al., 2019; Bagdasaryan et al., 2020). In this work we focus on targeted model poisoning attacks, wherein the attackers' goal is to reduce the model's performance on a specific set of datapoints from the test distribution or on certain sub-tasks using corrupted model updates, without compromising test accuracy.

写在综述里面，联邦学习的另外一个分类方法

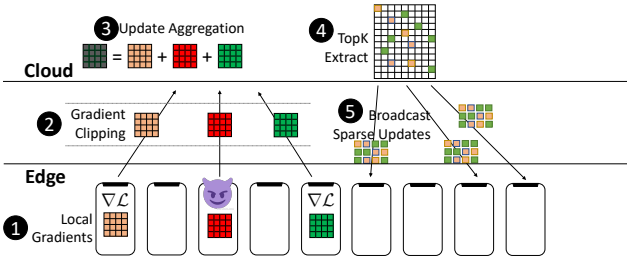


Figure 1: **Algorithm Overview**. The `SparseFed` algorithm **(1)** computes gradients locally, and then **(2)** the gradients are clipped. In the cloud, updates are aggregated **(3)**, and the top_k values are then **(4)** extracted and **(5)** broadcast as sparse updates to devices participating in the next round. The clipping and top_k extraction serve to mitigate the impact of the malicious update (red matrix).

The constraints of operating in the cross-device federated setting present challenges that make it difficult to train a model without enabling attackers. The data available across devices is not independent and identically distributed (non-i.i.d.). For example: when training a classification model on the camera roll of smartphone users, devices belonging to cat and dog owners will generate data from different distributions, but we are still interested in training one model to distinguish between cats and dogs (Hard et al., 2018). Therefore many benign device gradients will be very

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

设备只使用一次，利用历史梯度特征就失效了。

far apart in ℓ_2 distance, so heuristics that eliminate gradients that are outliers may not function well (Zhao et al., 2018; Rousseeuw, 1985). Devices only participate once during all of training (Kairouz et al., 2019), and this makes it difficult to use historical reputation mechanisms to shut out attackers (Cao et al., 2020).

Contributions. In this work, we present `SparseFed`, a new optimization algorithm for federated learning that can *train high-quality models under these constraints while greatly mitigating model poisoning attacks*. We describe `SparseFed` in detail in Section 2, but the main idea is intuitive: at each round, participating devices compute an update on their local data and clip the update. The server computes the aggregate gradient, and only updates the top_k highest magnitude elements. Because attackers will necessarily be moving in distinct directions from the majority of benign devices, the coordinates the attackers need to update in order to poison the model usually will not be updated. Our protocol is a defense at training time, and is complementary to the line of work that proposes test-time modifications for robustness such as smoothing (Xie et al., 2021; Weber et al., 2021). Prior defenses at training time use Byzantine-robust learning algorithms that bound the single iteration deviation between poisoned and clean models (Mhamdi et al., 2018; Blanchard et al., 2017a). However, the iterative nature of learning ensures that small deviations at the start of training compound exponentially.

We propose a framework for analyzing the robustness of defenses under the *certified radius* metric from prior work (Xie et al., 2021). The certified radius is an upper bound on the distance that a poisoned model can drift from a benign model, and limits the impact that an attacker can have on the model. Under our framework, `SparseFed` minimizes the certified radius by sparsifying the aggregate model updates.

We validate the effectiveness of our method empirically on four benchmark computer vision datasets and one natural language processing dataset, training models with between 6 and 40 million parameters on non-i.i.d. datasets that range between 50,000 and 800,000 examples. We evaluate `SparseFed` against four attacks from prior work (Bhagoji et al., 2019; Bagdasaryan et al., 2020; Fang et al., 2020; Sun et al., 2019) and two new attacks we introduce, in the cross-silo and cross-device settings. As we show in Table 2, `SparseFed` does not degrade test accuracy by more than 1%, mitigates attack accuracy, e.g. by over 97% on the FEMNIST dataset, and significantly outperforms prior work. The code to implement our defense is open-source.

2 SPARSEFED

In this section we introduce a framework for analyzing the robustness of machine learning protocols against poisoning attacks. We use this framework to motivate `SparseFed`, that uses gradient sparsification to mitigate attackers, and provide a theoretical analysis of its robustness, convergence and efficiency. The key tool we use is the *certified radius*, that is the upper bound on the distance between poisoned and benign models.

2.1 Certified radius as a framework for robustness

Notation: Let Z be the data domain and D^t be data sampled (not necessarily i.i.d.) from Z at iteration t. Let Θ be the class of models in d dimensions, and $\mathcal{L} : \Theta \times Z^* \to \mathcal{R}$ be a loss function. A protocol $f = (\mathcal{G}, \mathcal{A}, \lambda)$ consists of a gradient oracle $\mathcal{G}(\theta, D, t) \to \mathcal{R}^d$ that takes a model, a dataset and a round index and outputs the update vector u^t. f also includes an update algorithm $\mathcal{A} : u^t \in \mathcal{R}^d \to \mathcal{R}^d$, e.g. momentum. $\lambda(t) \in \mathcal{R}$ is a learning rate scheduler, possibly static, and $\Lambda(t)$ the cumulative learning rate $\Lambda(t) = \sum_{i=1}^{t} \lambda(t)$. The update rule of the protocol is then defined as $\theta_{t+1} = \theta_t - \lambda(t)\mathcal{A}(u^t)$.

Definition 1 (Poisoning Attack)**.** *For a protocol $f = (\mathcal{G}, \mathcal{A}, \lambda)$ we define the set of **poisoned** protocols $F(\rho)$ to be all protocols $f^* = (\mathcal{G}^*, \mathcal{A}, \lambda)$ that are exactly the same as f except that the gradient oracle $\mathcal{G}^*$ is a ρ-corrupted version of $\mathcal{G}$. That is, for any round t and any model θ_t and any dataset D we have we have $\mathcal{G}^*(\theta_t, D) = \mathcal{G}(\theta_t, D) + \epsilon$ for some ϵ with $||\epsilon||_1 \leq \rho$.*

Remark 1. *Under our attack model, the attacker can contribute to the update with a vector ϵ of ℓ_2 mass at most ρ. This model generalizes existing defenses, e.g. ℓ_2 clipping and Byzantine resilient aggregation rules (El Mhamdi et al., 2018).*

Definition 2 (Certified Radius)**.** *Let f be a protocol and $f^* \in F(\rho)$ be the a poisoned version of the same protocol. Let θ_T, θ_T^* be the benign and poisoned final outputs of the above protocols. We call R a certified radius for f if $\forall f^* \in F(\rho); R(\rho) \geq |\theta_T - \theta_T^*|_1$.*

Robustness Against Poisoning The *certified radius* has been established as a metric of the strength of defenses (Xie et al., 2021). Prior work has analyzed the certified radius in two ways. The first is minimizing the divergence between the benign and poisoned protocols in a single iteration, as in (Blanchard et al., 2017a; El Mhamdi et al., 2018; Blanchard et al., 2017b; Xie et al., 2021). As per (Xie et al., 2021), we know that a small certified radius improves robustness because models that are very close to each other are likely to predict the same label for the same datapoint. However, these papers assume i.i.d. data (El Mhamdi et al.,

2018; Blanchard et al., 2017b,a) and do not consider the *propagation error*: that small changes in early iterations can quickly compound and create a large divergence in the model. Therefore, defenses that aim to minimize the divergence in a single iteration via outlier detection or any other strategy cannot provide guarantees in the cross-device setting. The second is combinatorial bounds via ensembling (Jia et al., 2020; Cao et al., 2021). Combinatorial bounds do not compute the certified radius, and instead directly bound the change in the label probabilities. However, combinatorial bounds do not scale to the cross-device setting. For instance, the guarantees of (Cao et al., 2021) only hold so long as $\binom{n}{k} < 2\binom{n-m}{k}$ where n is the number of devices, m is the number of compromised devices, and k is the size of the ensemble (equation 4 in (Cao et al., 2021)) which is generally 1% of n. For $n \gtrsim 10^4$ (the cross-device setting), this means that (Cao et al., 2021) and other ensembling strategies cannot provide any guarantees when $m > 0.5\%$ of n.

In this section we introduce a framework for analyzing the certified radius of poisoning attacks in the cross-device setting.

Analyzing Propagation Error We conduct T rounds of the protocol f: at round $i \in [T]$ we receive an update, and use the output of the update algorithm $\mathcal{A}(u^t)$ to compute the new model θ_{t+1}. At each iteration, the upper bound ρ on ϵ gives the *additive error* introduced by poisoning. Because the protocol is adaptive, small additive errors introduced at early iterations can build upon each other and create large divergence. We refer to this as the *propagation error*. To analyze the propagation error we use the protocol Lipschitzness, defined in Definition 3.

Definition 3 (Coordinate Lipschitz)**.** *A protocol $f(\mathcal{G}, \mathcal{A}, \lambda)$ is c-coordinatewise Lipschitz if for any round $t \in [T]$, models $\theta_t, \theta_t^* \in \mathcal{M}$, and a dataset D we have that the outputs of the gradient oracle on any coordinate cannot drift too much farther apart. Specifically, for any coordinate index $i \in [d]$*

$$\left|\mathcal{G}(\theta_t^*, D)[i] - \mathcal{G}(\theta_t, D)[i]\right| \leq c \cdot |\theta_t^* - \theta_t|_1.$$

Example 1 (Training a single layer neural network with SGD)**.** *In this example, we compute the coordinatewise Lipschitz constant of the SGD protocol for a single layer neural network defined as $\sigma(\theta x)$, where σ is the softmax function and $\theta \in \mathcal{R}^d$ are the network parameters. For cross-entropy loss-based training using dataset D, we show that the constant $c = \frac{1}{4}$. Formally,*

$$\sup_{D,\theta_1,\theta_2} |g(D,\theta_1)[i] - g(D,\theta_2)[i]|_1 \leq \frac{1}{4}|\theta_1 - \theta_2|_1 \quad \forall i \in [d]$$

where $g(D,\theta)[i] = \frac{\partial \mathcal{L}}{\partial \theta_i}$. We provide the full computation in Appendix A.3.1.

Analyzing the Certified Radius In Theorem 1, we account for the propagation error and obtain a certified radius for general protocols. We provide a procedure for computing the certified radius exactly in Appendix A.3.2. Unlike prior work, we do not make any assumptions on the distribution of data across devices (El Mhamdi et al., 2018), the number of iterations where the attacker is present (Xie et al., 2021), the number of devices (Cao et al., 2021), or the number of poisoned points (Jia et al., 2020). We can account for these factors by adjusting the relevant quantities. Although the computed certified radius from Theorem 1 may not be tight, we expect protocols that improve the bound to benefit from improvements in their robustness. In the next section, we show one way to improve this bound with sparsification by decreasing the propagation error.

Theorem 1. *Let f be a c-coordinatewise-Lipschitz protocol on a dataset D. Then $R(\rho) = \Lambda(T)(1 + dc)^{\Lambda(T)}\rho$ is a certified radius for f.*

2.2 Security analysis of `SparseFed`

In this section we use our certified radius framework to motivate `SparseFed`, that uses gradient sparsification and norm clipping to mitigate attackers, and provide a theoretical analysis of its robustness.

The building blocks of robustness The two components of the certified radius are the additive error and the propagation error. The additive error represents the attacker's power in terms of an upper bound ρ on the noise vector ϵ. We can enforce this with device level ℓ_2 gradient norm clipping, that is a standard technique employed by prior work (Sun et al., 2019; Wang et al., 2020a). If $p\%$ of devices are compromised and the parameter of ℓ_2 clipping is L then $\rho = pL$. The propagation error represents the protocol's inherent robustness in terms of the Lipschitz constant $c \cdot d$.

Update sparsification techniques reduce the number of non-zero entries in the aggregated stochastic gradient before it is applied to the global model. Global top_k sparsification (Stich et al., 2018) is one such method that updates only the k coordinates with the largest magnitude, where $k||d$, and converges at the same rate as SGD (Karimireddy et al., 2019). To the best of our knowledge, we are the first to propose the use of global update sparsification as a building block for robust federated learning.

We propose `SparseFed`, presented in full in Algorithm 1, by combining sparsification and norm clipping. At each round of federated learning, each device downloads the current global model and computes an update on their local dataset. This update is clipped according to a specified ℓ_2 norm. This controls ρ and allows us

Algorithm 1 `SparseFed`

Input: number of coordinates to update each round k, learning rate λ, number of timesteps T, local batch size b, number of devices selected per round n, norm clipping parameter L, local epochs τ, local learning rate γ, device datasets $D_{j=1}^{n}$, momentum ρ
Initialize model θ_0 using the same random seed on the devices and aggregator
Initialize memory vector $\mathrm{W}_t = 0$, momentum vector $\mathrm{R}^t = 0$
for $t = 1, 2, \cdots T$ **do**
  Randomly select n devices $d_1, \ldots d_n$
  loop {In parallel on devices $\{d_i\}_{i=1}^{n}$}
    Download new model weights $\theta_t = \theta$
    for $m \in \tau$ **do**
      Compute gradient $\mathrm{g}_t^i = \frac{1}{b}\sum_{j=1}^{l} \nabla_\theta \mathcal{L}(\theta^t, \mathrm{D}_j)$
      Accumulate gradient $\theta_t = \theta_t - \gamma(t, m)\mathrm{g}_t^i$
    end for
    Compute update $\mathrm{u}_t^i = \theta_t - \theta$
    Clip update $\mathrm{u}_t^i = \mathrm{u}_t^i \cdot \min(1, \frac{L}{|\mathrm{u}_t^i|_2})$
  end loop
  Aggregate gradients $\mathrm{u}_t = \frac{1}{n}\sum_{i=1}^{n} \mathrm{u}_t^i$
  Momentum: $\mathrm{R}^t = \rho \mathrm{R}^{t-1} + u^t$
  Error feedback: $\mathrm{W}_t = \mathrm{u}_t + \mathrm{W}_t$
  Extract top_k: $\Delta_t = top_k(\mathrm{W}_t)$
  Error accumulation: $\mathrm{W}_{t+1} = \mathrm{W}_t - \Delta_t$
  Momentum factor masking: $\mathrm{R}_{t+1} = \mathrm{R}_t - \Delta_t$
  Update $\theta_{t+1} = \theta_t - \lambda(t)\Delta_t$
end for
Output: $\{\theta^t\}_{t=1}^{T}$

to control the additive error. The server aggregates all updates with a simple average. The aggregated update is added to an error feedback vector. The server extracts the top_k magnitude coordinates from the error feedback vector, and zeroes out these coordinates from the error feedback vector. The top_k coordinates are used to update the global model. Because we update $k << d$ coordinates, we reduce the propagation error.

We first define a notion of sparsity for a protocol and use it to prove our main theorem. In Appendix A.1.1 we discuss why `SparseFed` satisifies this notion.

Definition 4 ((k,γ)-sparsity)**.** *A federated learning protocol $d = (\lambda, \mathcal{G}, \mathcal{A})$ is (k,γ)-sparse on a dataset D if for all $u_t = \mathcal{G}(\theta_{t-1}, D)$ generated during the process of training on D $\mathcal{A}(u_t)$ only has k non-zero elements and we have*

$$|\mathcal{A}(u_t) - u_t|_1 \leq \gamma.$$

Theorem 2. *Let f be a c-coordinatewise-Lipschitz and (k,γ)-sparse protocol on a dataset D. Let $w = min(d, 2k)$ then $R(\rho) = \Lambda(T)(1+wc)^{\Lambda(T)}(\rho + 2\gamma)$ is a certified radius for f.*

In Theorem 2 we improve the base term in propagation error term by a factor of $\frac{d}{2k}$, that can be multiple orders of magnitude.

In summary, `SparseFed` aggregates clipped updates from devices and only updates the top_k coordinates of the aggregated update. We show that the use of top_k update sparsification improves the certified radius.

2.3 Efficiency and Convergence Analysis of `SparseFed`

Convergence Analysis: We show that `SparseFed` converges as well as SGD in the base setting (e.g. when no attackers are present). We make standard assumptions on the smoothness of the loss function and bounded gradient which are only necessary for our convergence analysis (Rothchild et al., 2020; Karimireddy et al., 2019; Xie et al., 2021).

Assumption 1 (Smoothness)**.** *$\mathcal{L}$ is ℓ-smooth if* $\forall x, y \in \mathcal{R}^d \;\; |\mathcal{L}(x) - (\mathcal{L}(y) + \langle \nabla \mathcal{L}(x), x - y\rangle)| \leq \frac{\ell}{2}\|x - y\|_2^2$

Assumption 2 (Moment Bound)**.** *For any x, our oracle returns $\boldsymbol{g}$ s.t. $\mathbb{E}[\boldsymbol{g}] = \nabla\theta(x)$ and $\mathbb{E}\|\boldsymbol{g}\|_2^2 \leq \sigma^2$*

Theorem 3 (Asymptotic Convergence of `SparseFed`)**.** *For a protocol $f, \lambda(t) = \sqrt{t+1}^{-1}, \tau = 1, \mathcal{A} = top_k, \mathcal{L}$ satisfying Assumption 1, $\mathcal{G}$ satisfying Assumption 2, we get the convergence rate of*

$$\min_{t \in T} \mathbb{E}[\|\nabla \mathcal{L}(\theta_t)\|] \leq \frac{4(\theta_0 - \theta_*) + \ell\sigma^2}{2\sqrt{T+1}} + \frac{4\ell^2\sigma^2(1-\delta)}{\delta^2(T+1)}$$

Therefore, f converges asymptotically at the SGD rate.

Communication efficiency of `SparseFed`: In practical deployments of federated learning systems, communication efficiency must be prioritized. The top_k sparsification used in `SparseFed` requires communicating the full gradient at every iteration and therefore is not communication efficient. `FetchSGD` is a communication efficient approximation of top_k sparsification using the Count Sketch data structure (Rothchild et al., 2020). Because `FetchSGD` provably approximates the heavy hitter recovery properties of top_k (Rothchild et al., 2020), it inherits these robustness guarantees. In Appendix B.9, we compare implementations of `SparseFed` using both top_k and `FetchSGD` and find that when using the latter, we are able to prove robustness and communication efficiency.

3 EVALUATION

We empirically demonstrate the effectiveness of our `SparseFed` defense against strong attackers in a variety of realistic experimental settings. To this end, we set up the first environment to simulate model poisoning attacks on the cross-device setting of federated learning with tens of thousands of devices, aiming to emulate a real-world deployment as closely as possible. In contrast, prior work has mostly evaluated attacks in the cross-silo setting with 10s to 100s of devices (Bhagoji et al., 2019; Bagdasaryan et al., 2020; Wang et al.,

2020a; Fang et al., 2020). We evaluate `SparseFed` in both the cross-silo and cross-device settings against a breadth of attacks and find that we significantly outperform prior defenses.

3.1 Experimental setup

All methods are implemented in PyTorch (Paszke et al., 2019). We conduct experiments on computer vision (CIFAR10, CIFAR100, FashionMNIST, FEMNIST), and natural language processing (Reddit) datasets.

Federated Extended MNIST (FEMNIST) dataset (Caldas et al., 2018) is a dataset constructed specifically as a benchmark for federated learning. Our goal is to train a model in a true federated fashion, i.e. we can only view each datapoint once. We use a 40M-parameter ResNet101 for this task. FEMNIST has 63 classes and a natural non-i.i.d. partitioning with an average of 226.83 datapoints for each of 3550 users, for a total of 805,263 datapoints. Our goal is to simulate the cross-device setting as closely as possible, so we aim to have $\gtrsim 50$ devices participating in each round, with each device participating exactly once (Kairouz et al., 2019), without exceeding a batch size of ≈ 600. We split each user evenly into $9-10$ devices, yielding $35,000$ simulated devices and 35 devices participating in each iteration. Each device has a non-i.i.d. dataset that includes data from multiple classes.

We also conduct experiments on Fashion MNIST (FMNIST) (Xiao et al., 2017), CIFAR10/CIFAR100 (Krizhevsky et al., 2009), that are benchmark tasks for computer vision. We provide the experimental parameters in Table 1 for the cross-silo and cross-device settings, for the number of devices d, number of devices participating at each iteration w, percentage of attackers p, and the auxiliary set size s: the number of datapoints we are attempting to modify model behavior on for the targeted model poisoning attack. A key design choice is how to distribute the training data among simulated devices. In the cross-silo setting, we simply distribute data i.i.d. across devices. In the cross-device setting, we follow previous work (Rothchild et al., 2020) and artificially create non-i.i.d. datasets by giving each device images from only a single class. At each round of federated learning, a subset of devices are randomly selected to participate. Our 7M-parameter ResNet9 model architecture, data preprocessing, and most hyperparameters follow (Page, 2019).

Parameter	Cross-silo	Cross-device
i.i.d.	TRUE	FALSE
d (# devices)	1000	100000
w (# participating)	10	100
p (% compromised)	1	2
a ($\mathbb{E}$[#] attackers per iter)	0.1	2
s (auxiliary set)	50	500
b (local batch size)	50	5

Table 1: Parameters for CIFAR10, CIFAR100, MNIST, FashionMNIST in cross-silo and cross-device settings

3.2 Attack details:

We experiment with a number of attacks: targeted model poisoning, untargeted model poisoning, semantic backdoor, model replacement, colluding attack, and adaptive attack. In all attacks, the attacker controls a number of devices and realizes the attack by uploading poisoning gradients to the server. $p\%$ of the d simulated devices are attackers. We sample w devices randomly at every iteration to participate, so we expect $a = p \cdot w$ devices to be compromised at each iteration. Empirically we find that the attacker does not need to be present until the last $\approx 20\%$ of training to insert the attack, in line with prior work Bagdasaryan et al. (2020).

Targeted model poisoning: We follow the attack procedure of (Bhagoji et al., 2019). We construct an auxiliary dataset of size s with the following procedure: First, we sample s points from the test distribution. We then flip the label to one of the labels that is not the ground truth. The objective of the attacker is to maximize the accuracy of the trained model on the auxiliary dataset (*attack accuracy*), typically while ensuring that the model performance *on the remaining data* does not degrade significantly. The attacker is present throughout the course of training.

Untargeted model poisoning attack: Also known as a Byzantine attack, the attacker attempts to decrease the test accuracy of the trained model (Blanchard et al., 2017a; Mhamdi et al., 2018). The attacker is present throughout the course of training, and succeeds when the model parameters diverge and can no longer be trained without resetting to an earlier checkpoint.

Semantic backdoor via model poisoning: We follow the backdoor attack described in (Sun et al., 2019). We train a model on FEMNIST and simulate 35,000 devices, 1000 of which are attackers. We consider the semantic backdoor task of misclassifying the digit 7 as 1, creating 3000 backdoors, the number of instances of the digit 7 in the unperturbed validation set, and include results in Table 2. We include experiments that vary the semantic backdoor task in Appendix B.5.

Model replacement: In Appendix B.5 we evaluate `SparseFed` against the model replacement attack of (Bagdasaryan et al., 2020) on the Reddit dataset. The attacker participates in a single iteration toward the end of training and scales their gradient so that they can entirely replace the trained global model. In order

to optimize for the ℓ_2 norm clipping constraint, the attacker uses Projected Gradient Descent (PGD) with knowledge of the norm clipping parameter.

Colluding attack: In Alg. 2 we propose the colluding attack for the cross-device setting, where multiple attackers can be present in a single iteration. The attackers collude by *each* sending the same update. In the cross-device setting, we combine the colluding attack with the targeted model poisoning attack, untargeted model poisoning attack, or semantic backdoor attack.

Algorithm 2 Attack

Input: learning rate η, local batch size ℓ, norm clipping parameter L, number of local epochs e

This procedure is used by all attackers in a round to ensure that they upload the same update
for number of PGD epochs $e_i \in e$ **do**
  Compute stochastic gradient g_i^t on batch B_i of size ℓ: $\mathrm{g}_i^t = \frac{1}{\ell}\sum_{j=1}^{l} \nabla_{\mathrm{M}} \mathcal{L}(\mathrm{M}_{e_i}^t, \mathrm{D}_j)$
  Update local model $\widehat{M}_{e_{i+1}}^t = \mathrm{M}_{e_i}^t - \eta \mathrm{g}_i^t$
  Project accumulated update onto the perimeter of the ℓ_2 constraint $\mathrm{M}_{e_{i+1}}^t = M_0^t - CLIP(\widehat{M}_{e_{i+1}}^t - M_0^t)$
end for

Output: M_e^t

3.3 SparseFed is an effective defense in the cross-silo setting

We first evaluate `SparseFed` in the cross-silo setting common to prior work to show the improvement of `SparseFed` over the baseline ℓ_2 clipping defense. In Figures 2a and 2b we see that appropriately choosing k allows us to mitigate the attack without harming convergence. As we explain in Section 2, ℓ_2 norm clipping is insufficient to mitigate the attack because minor perturbations at early iterations can propagate over the course of training. This intuition validated by our results, that show that the use of norm clipping is not sufficient to deter the attacker. From this, we can see the importance of coupling both norm clipping and update sparsification in `SparseFed`. The tradeoff that `SparseFed` introduces for the attacker is forcing them to have large magnitude elements in order to have their component of the update appear in the top_k, however these are clipped due to the use of ℓ_2 norm clipping, leading to ineffective attacks.

Impact of sparsification parameter k: `SparseFed` requires the sparsification parameter k. We provide an algorithm for selecting k in Alg. 3. When using ResNet9, we obtain a value of $k = 1e3$ that does not significantly compromise convergence and use this across all datasets that use ResNet9 (FMNIST, CIFAR10, CIFAR100). When using ResNet101, we obtain a value of $k = 4e4$ and use this for all FEMNIST experiments. Fig. 2 shows the sparsification-utility-robustness tradeoff for the cross-silo and cross-device settings. For small k and large k neither the attack nor the model converge. When k is too small, `SparseFed` approaches a no-op as $k \to 0$. When k is too large, the use of momentum factor masking (Stich, 2019; Lin et al., 2017) prevents convergence to a benign optimum, which in turn makes it difficult for the attacker to perform model replacement (Bagdasaryan et al., 2020). Most choices of k mitigate the attack, and the best choice of k does not significantly degrade test accuracy. We expect that practitioners will be able to easily tune the correct value of k for their purpose, because the parameter can be tuned on a single device and does not need to be finetuned across datasets for the same architecture.

Algorithm 3 Selecting k

Input: model θ, maximum information loss ω, number of model parameters d, number of iterations in an epoch r, number of gradients to sample n (more samples gives a better estimate of ω)

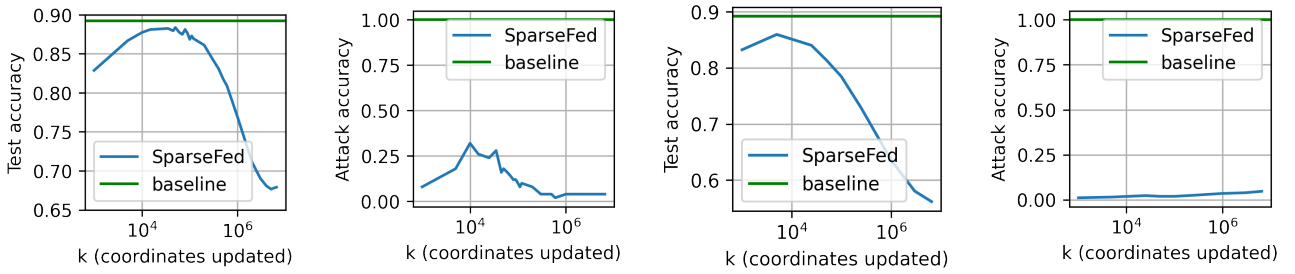
set initial k $k = \frac{d}{r}$
set initial realized information loss $\delta = \infty$
while $\delta > \omega$ **do**
  compute n sample minibatch gradients $\{g\}_{j=1}^{n} | g_j = \nabla_\theta \mathcal{L}(\theta, \mathrm{z}_j)$
  extract top-k $\{u\}_{j=0}^{n} | u_j = top_k(g_j)$
  calculate average l_1 mass lost $\delta^* = \frac{1}{n}\sum_{j=1}^{n} |g_j - u_j|_1$
  update $\delta = \min(\delta, \delta^*)$
  if $\delta > \omega$ **then**
    $k = k + \frac{d}{r}$
  end if
end while

Output: k

3.4 SparseFed is the most effective defense in the cross-device setting

We next evaluate `SparseFed` in the cross-device setting, which includes many more devices and the challenge of optimizing over small, non-i.i.d. datasets. In Figures 2c and 2d we see that appropriately choosing k allows us to mitigate the attack without harming convergence. This is the setting that `SparseFed` is designed for, and we evaluate it against prior work.

Existing defenses cannot handle collusion Prior empirical defenses are designed under the assumption that data is distributed i.i.d. across devices and attackers do not collude amongst each other. We carry out attacks in the cross-device setting, where data is non-i.i.d. and attackers have no restriction on their ability to collude, and conclude that `SparseFed` is the only defense that maintains empirical robustness in this setting. In Table 2 we evaluate all defenses against a population of colluding attackers across all four datasets. We report the attack accuracy; when a defense fails to converge, we mark it with DNC (this is discussed further below). Bulyan and other Byzantine-resilient aggregation rules rely on eliminating outliers (Mhamdi et al., 2018). Specifically, Bulyan determines outliers by measuring



(a) Impact of k on convergence, cross-silo setting

(b) Impact of k on robustness, cross-silo setting

(c) Impact of k on convergence, cross-device setting

(d) Impact of k on robustness, cross-device setting

Figure 2: Tradeoffs between sparsification, convergence and robustness for the targeted model poisoning attack on CIFAR10 in the cross-silo and cross-device settings.

their distance from other updates in the population. Because the attackers are colluding, their updates have a distance of 0 from each other, and as a result Bulyan does not eliminate them. Trimmed mean fails even against a single attacker, because trimmed mean relies on the assumption that a Byzantine attacker will either be the minimum or maximum value. However, this assumption does not hold for a model poisoning attacker. These conclusions are in line with conclusions from prior work (Bhagoji et al., 2019; Fang et al., 2020; Baruch et al., 2019a). Our experimental results demonstrate that even when attackers collude, they are unable to overcome the trade-off that is enforced by `SparseFed`.

Byzantine attacks: In Table 3a we validate the effectiveness of `SparseFed` against untargeted model poisoning attacks, or Byzantine attacks. Byzantine attacks succeed more easily in the cross-device setting against prior defenses for the reasons mentioned above, but `SparseFed` is still able to mitigate these.

Impact of defenses on test accuracy: In Table 3b we evaluate the impact of each defense on convergence in the absence of attacks. Krum and coordinate median do not converge in the cross-device setting. When Krum chooses a single model, it is overfitting the global model to the small local dataset of a single device. Coordinate median does not converge because of the gap between median and mean. Trimmed mean and Bulyan have a minor impact on test accuracy when the robustness parameter f is small. When 2 out of 100 devices are compromised, Bulyan will discard $4f + 2 = 10$ gradients in order to maintain robustness. For the challenging FEMNIST task, this information loss is too much and these methods do not converge. These observations are in line with conclusions from prior work, that make the case for more complex algorithms (Chen et al., 2020; Muñoz-González et al., 2019; Yin et al., 2019) that are out of the scope of this paper's evaluation. Norm clipping acts as regularization and

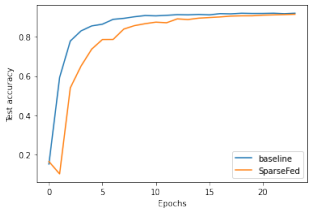


Figure 3: `SparseFed` converges at the same rate as the baseline (FedAvg) on CIFAR10 in the cross-device setting

does not have much impact on the test accuracy. As we show in Fig. 2c, `SparseFed` does not impact test accuracy significantly. In Fig 3 we empirically validate the speed of convergence of `SparseFed` and find that it converges at the same rate as `FedAvg`, even in the presence of attackers.

Verification of theory: In Section 2 we analyze the certified radius of `SparseFed`. In Table 2 we provide observed distances between poisoned and benign models when using various defenses, and conclude that `SparseFed` has both the lowest distance and lowest attack accuracy. This verifies our theoretical guarantees.

ℓ_2 Norm clipping: In Table 4 we improve the Byzantine-resilient defenses by combining them with ℓ_2 norm clipping. All results for all defenses include norm clipping. In Appendix B.2 we further show that norm clipping is necessary in `SparseFed`.

Hyperparameter tuning (Appendix B.3): We tune standard hyperparameters on the `FedAvg` baseline, and use these hyperparameters for all experiments. Krum, Bulyan and trimmed mean require the parameter f, the number of attackers present in the system. `FedAvg` requires the number of local epochs, a batch size for each epoch, and learning rate decay. In Table 5 we vary the number of local epochs and use a single local epoch as the optimal value for the cross-device

Table 2: Krum, Bulyan, trimmed mean, coordinate median, norm clipping (clipping, $\ell_2 = 5$), and `SparseFed` on FMNIST, CIFAR10, CIFAR100, and FEMNIST in the cross-device setting. `SparseFed` reduces the attack accuracy significantly more than other defenses. If a defense cannot converge we denote it with DNC. We report ℓ_1 distances between poisoned and unattacked models at the end of training. `SparseFed` has less than half the distance of the next best defense.

Defense	**Attack Accuracy (%) (Dataset)**				**Distance (thousands)**	
	CIFAR10	CIFAR100	FMNIST	FEMNIST	CIFAR10	FMNIST
Trimmed Mean	44.6	81.4	100	DNC	64	41
Bulyan	36.2	81.8	100	DNC	68	39
Clipping	100	100	100	100	73	40
SparseFed (Ours)	4.6	23	2.2	2.86	31	16

(a) Comparison of Byzantine failure success rates on FashionMNIST. Ours: Cross-device setting. (Fang et al., 2020): Numbers from their paper with 100 devices, 20 attackers (20 % compromised)

Defense	**Test error**	
	Ours	**(Fang et al., 2020)**
Krum	DNC	87
Median	DNC	29
Trimmed mean	90	52
Bulyan	90	38
`SparseFed`	20	N/A

(b) Comparing the impact on test accuracy of the defenses. Cross-device setting, no attackers (averaged over 3 runs).

Defense	**Decrease**	**Test Acc**
No defense	0 $\pm$0	90.0 $\pm$0.1
ℓ_2	2.0 $\pm$0.1	88.0 $\pm$0.1
DP ($\sigma = 0.025$)	20.0 $\pm$0.2	70.50 $\pm$0.2
Krum	80.0 $\pm$0	10.0 $\pm$0
Median	80.0 $\pm$0	10.0 $\pm$0
Trimmed mean ($f = 2$)	9.23 $\pm$0.8	80.77 $\pm$0.8
Bulyan ($f = 2$)	9.56 $\pm$0.79	80.44 $\pm$0.79
Bulyan ($f = 10$)	66.48	23.52
`SparseFed` ($k = 1e3$)	10.21 $\pm$0.7	79.79 $\pm$0.7
`SparseFed` ($k = 5e4$)	3.0 $\pm$0.01	87.0 $\pm$0.01

Table 4: Implementing norm clipping greatly mitigates the effectiveness of the attack against Bulyan and trimmed mean when no colluding attackers are present. CIFAR10, 1e4 devices, 100 attackers.

Defense	**Test acc**	**Attack acc**
Bulyan (ℓ_2)	83.64	10.0
Bulyan	84.94	38.6
Trimmed Mean (ℓ_2)	77.42	71.6
Trimmed	81.99	100.0

Table 5: `FedAvg` convergence does not benefit from doing multiple local epochs. We use local learning rate=0.9, but even for a small number of local epochs convergence does not benefit, and at these small number of local epochs a smaller local learning rate would not have much impact because the exponential decay factor is not large. CIFAR10, 10000 devices, no attackers.

Num. epochs	**Test acc decrease**	**Test acc**
1	0	90
2	0.41	89.59
5	80	10

setting, in line with prior work Rothchild et al. (2020) ℓ_2 clipping requires the clip parameter.

Stealth of attack (Appendix B.5): We validate that the attack is stealthy when it succeeds, insofar as it does not compromise normal model operation significantly. For the targeted model poisoning attack, the auxiliary dataset is divided equally across all classes. Thus, the performance of any one class does not degrade significantly. In the semantic backdoor attack, by definition the model fails on the class that is flipped by the semantic backdoor.

Strength of attack: In Table 6 we increase the fraction of compromised agents until `SparseFed` is no longer robust. Unsurprisingly, the power of collusion enables attackers to quickly overtake even `SparseFed`, the strongest defense we evaluate, when the fraction of compromised agents increases past 5 %. Prior work Shejwalkar et al. (2021) argues that a realistic value for the fraction of compromised agents should not exceed 0.1%. Therefore, only in an unrealistic regime does our defense fail.

In Appendix B.6: In order to validate our defense, we ensure that we test against the strongest available

Table 6: Varying the fraction of compromised devices for `SparseFed` on the CIFAR100 cross-device setting.

Fraction compromised (%)	Attack Accuracy (%)
2	4.4
4	41.20
6	100

attacks. We show that our proposed attack is stronger than previous attacks against both norm-based defenses as well as Byzantine defenses that do not rely on norm-clipping (Bulyan, Trimmed Mean etc.) We compare the attack accuracy of the colluding attack used in this work against prior attacks on Byzantine-resilient aggregation rules, and conclude that our attack is significantly more powerful than prior work considers. The key factor in the strength of the colluding attack is the ability for colluding attackers to send identical gradients and therefore avoid outlier detection by essentially vouching for each other. We also include experiments using an adaptive attack that we design against `SparseFed` which has perfect knowledge of the top_k coordinates.

4 Related Work

Federated learning: There are two main settings for federated learning: the cross-device setting and the cross-silo setting (Kairouz et al., 2019). The cross-device setting features all the complications we introduce earlier, namely hundreds of thousands or millions of devices with non-iid data distributions participating sparsely (Kairouz et al., 2019). This is the original setting for federated learning, and it is the setting where we focus our analysis. Most prior work has operated on the scale of the cross-silo setting, with experiments on at most 100 devices (Wang et al., 2020a; Fang et al., 2019; Bhagoji et al., 2019; Bagdasaryan et al., 2020).

Targeted model poisoning attacks: The goal of the attacker in a targeted model poisoning attack is to modify the model such that particular inputs induce misclassification (Chen et al., 2017; Biggio et al., 2012; Bhagoji et al., 2019; Bagdasaryan et al., 2020; Wang et al., 2020a). This can be a random set of data drawn from the validation distribution, with the labels randomly flipped to another class (Bagdasaryan et al., 2020; Bhagoji et al., 2019). This can also be a semantic backdoor, wherein the attacker tries to flip the label of all data from a target class to another specific class, e.g. classifying all 1s as 7s in the MNIST dataset (Sun et al., 2019). (Wang et al., 2020a) shows that backdoors sampled from the low-probability portion of the distribution can break existing defenses and are a byproduct of the existence of adversarial examples.

Prior defenses: The two main bodies of work on defenses against poisoning attacks are *certified robustness* and *empirical robustness*.

Ensemble methods have been proposed to certify robustness against poisoning attacks (e.g. (Jia et al., 2020; Cao et al., 2021)). As we show in Section 2, ensemble methods do not scale to the cross-device setting because they rely on most subsamples not containing attackers. This assumption breaks down for $n \gtrsim 10^4$, where it is very unlikely to randomly sample enough clients to train a good model without sampling an attacker. Data poisoning defenses are insufficient against malicious clients that can manipulate model updates. Provably secure defenses against data poisoning certify robustness in terms of the number of poisoned examples, but a single compromised device can poison an arbitrary number of their training datapoints, that breaks the core assumption of secure defenses for data poisoning. (Levine and Feizi, 2021) partition the training dataset with a hash function for certified robustness, but their defense is only applicable to deterministic training algorithms for data poisoning. Works that use randomized smoothing at testing time (Rosenfeld et al., 2020; Wang et al., 2020b) are complementary to our work, that is a procedure solely for training. (Xie et al., 2021) use noise during training and provide an inference-time smoothing procedure to certify robustness in federated learning. However, their goal is to finetune an already poisoned model to erase backdoors.

There are a number of defenses that provide empirical robustness against poisoning attacks. In our evaluation at scale we also compare to five of these empirical defenses, each of which we adapt and improve for the federated setting: trimmed mean (Yin et al., 2019), median (Yin et al., 2019), Krum (Blanchard et al., 2017a), Bulyan(Mhamdi et al., 2018), and norm clipping (Sun et al., 2019). We provide exact algorithms for these defenses in Appendix B.1. These defenses only achieve provable guarantees with an i.i.d. assumption on the device data distributions (Mhamdi et al., 2018), that is not valid in the cross-device setting. Further, they have been shown to be ineffective against model poisoning attacks (Baruch et al., 2019b; Bhagoji et al., 2019; Xie et al., 2020; Fang et al., 2020) in practice, e.g. by crafting updates that do not significantly differ from benign updates (Bhagoji et al., 2019).

5 Discussion

Prior work in poisoning attacks on federated learning has demonstrated that existing defenses are vulnerable to attacks. We complement this body of work by introducing `SparseFed`, an optimization algorithm for federated learning which combines update sparsification and norm clipping. We prove a certified radius for `SparseFed` that improves over baseline federated learning. `SparseFed` does not significantly decrease

test accuracy, and mitigates attacks in both the cross-silo and cross-device settings. We evaluate `SparseFed` empirically against existing defenses, and confirm that it outperforms these against multiple strong attacks.

Acknowledgements: This work was supported in part by the National Science Foundation under grant CNS-1553437, Department of Energy's EUREICA grant DE-OE0000920, the ARL's Army Artificial Intelligence Innovation Institute (A2I2), Schmidt DataX award, and Princeton E-ffiliates Award. The work done at IBM research was sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Neural Information Processing Systems*, 2020a.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *Proceedings of the 36th International Conference on Machine Learning*, pages 634–643, 2019.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948, Online, 26–28 Aug 2020. PMLR. URL `http://proceedings.mlr.press/v108/bagdasaryan20a.html`.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe M Kiddon, Jakub Konecny, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. URL `https://arxiv.org/abs/1902.01046`.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Francoise Beaufays, Sean Augenstein, Hubert Eichner, Chloe Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint 1811.03604*, 2018.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Peter Rousseeuw. Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications Vol. B*, pages 283–297, 01 1985. doi: 10.1007/978-94-009-5438-0_20.

Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping, 2020.

Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks, 2021.

Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks, 2021.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, pages 118–128, 2017a.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and N. Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3521–3530, 2018.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 119–129. Curran Associates, Inc., 2017b. URL `https://proceedings.neurips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf`.

Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks, 2020.

Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Provably secure federated learning against malicious clients, 2021.

S. Stich, Jean-Baptiste Cordonnier, and M. Jaggi. Sparsified sgd with memory. In *NeurIPS*, 2018.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *ICML*, 2019.

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito,

Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

David Page. How to train your resnet, Nov 2019. URL `https://myrtle.ai/how-to-train-your-resnet/`.

Sebastian U Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations (ICLR)*, 2019.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Moran Baruch, Gilad Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *arXiv preprint arXiv:1902.06156*, 2019a.

Xiangyi Chen, Tiancong Chen, Haoran Sun, Z. Wu, and Mingyi Hong. Distributed training with heterogeneous data: Bridging median- and mean-based algorithms. *ArXiv*, abs/1906.01736, 2020.

Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *ArXiv*, abs/1909.05125, 2019.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2019.

Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning, 2021.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to Byzantine-robust federated learning. *arXiv preprint arXiv:1911.11815*, 2019.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1467–1474, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. URL `http://dl.acm.org/citation.cfm?id=3042573.3042761`.

Alexander Levine and Soheil Feizi. Deep partition aggregation: Provable defense against general poisoning attacks, 2021.

Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and J. Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing, 2020.

Binghui Wang, Xiaoyu Cao, Jinyuan jia, and Neil Zhenqiang Gong. On certifying robustness against backdoor attacks via randomized smoothing, 2020b.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8635–8645. Curran Associates, Inc., 2019b. URL `https://proceedings.neurips.cc/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf`.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Walter Rudin. *Principles of mathematical analysis / Walter Rudin*. McGraw-Hill New York, 3d ed. edition, 1976. ISBN 007054235. URL `http://www.loc.gov/catdir/toc/mh031/75017903.html`.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

SparseFed: Supplemental Material

The Appendix is organized as follows:

- Appendix A gives full proofs of the theorems in the main body.
  - Appendix A.1 the proof of the main certified radius theorem
  - Appendix A.2 the convergence analysis of the defense
  - Appendix A.3.1 full computation of Lipschitz constant of a single layer network.
  - Appendix A.3.2 procedure for computing certified radius
- Appendix B.1 gives details on the methods and metrics used throughout the main body of the paper and the Appendix.
  - Appendix B.1.1 `FedAvg`
  - Appendix B.1.2 the attack
  - Appendix B.1.3 Krum, Bulyan, trimmed mean, coordinate median
  - Appendix B.1.4 `SparseFed` implemented with true top-k and `FetchSGD`
  - Appendix B.1.5 an adaptive algorithm for selecting k in `SparseFed`.
  - Appendix B.1.6 the metrics used throughout the main body and Appendix.
- The rest of Appendix B gives further experimental results for the conclusions reached in the main body of the paper.
  - Appendix B.2 the use of ℓ_2 norm clipping in `SparseFed` and prior defenses.
  - Appendix B.3 the full range and results of hyperparameters tuned.
  - Appendix B.4 the impact of each defense on convergence.
  - Appendix B.5 the stealth of the attack.
  - Appendix B.6 validates that we are evaluating `SparseFed` against the strongest available attack.
  - Appendix B.7 the compatibility of `SparseFed` with secure aggregation.
  - Appendix B.8 the parameters of the attack and how they are tuned.
  - Appendix B.9 the case for `SparseFed` implemented with `FetchSGD` as an algorithm which achieves security and communication efficiency.
- Appendix C discusses the limitations and societal impact of our work.

A Proofs

A.1 Propagation analysis of sparse aggregation

Here we prove Theorems 2 and 1. Before that, we introduce several definitions that will be used in stating and proving the Theorem.

Notation: Let Z be the data domain and D^t be data sampled (not necessarily i.i.d.) from Z at iteration t. Let Θ be the class of models in d dimensions, and $\mathcal{L} : \Theta \times Z^* \to \mathcal{R}$ be a loss function. A protocol $f = (\mathcal{G}, \mathcal{A}, \lambda)$ consists of a gradient oracle $\mathcal{G}(\theta, D, t) \to \mathcal{R}^d$ that takes a model, a dataset and a round index and outputs the update vector u^t. f also includes an update algorithm $\mathcal{A} : u^t \in \mathcal{R}^d \to \mathcal{R}^d$, e.g. momentum. $\lambda(t) \in \mathcal{R}$ is a learning rate scheduler, possibly static, and $\Lambda(t)$ the cumulative learning rate $\Lambda(t) = \sum_{i=1}^{t} \lambda(t)$. The update rule of the protocol is then defined as $\theta_{t+1} = \theta_t - \lambda(t)\mathcal{A}(u^t)$.

Definition 1 (Poisoning Attack [Restated]) *For a protocol $f = (\mathcal{G}, \mathcal{A}, \lambda)$ we define the set of **poisoned** protocols $F(\rho)$ to be all protocols $f^* = (\mathcal{G}^*, \mathcal{A}, \lambda)$ that are exactly the same as f except that the gradient oracle $\mathcal{G}^*$ is a ρ-corrupted version of $\mathcal{G}$. That is, for any round t and any model θ_t and any dataset D we have we have $\mathcal{G}^*(\theta_t, D) = \mathcal{G}(\theta_t, D) + \epsilon$ for some ϵ with $||\epsilon||_1 \leq \rho$.*

Definition 3 (Coordinate Lipschitz [Restated]) *A protocol $f(\mathcal{G}, \mathcal{A}, \lambda)$ is c-coordinatewise Lipschitz if for any round $t \in [T]$, models $\theta_t, \theta_t^* \in \mathcal{M}$, and a dataset D we have that the outputs of the gradient oracle on any coordinate cannot drift too much farther apart. Specifically, for any coordinate index $i \in [d]$*

$$\left|\mathcal{G}(\theta_t^*, D)[i] - \mathcal{G}(\theta_t, D)[i]\right| \leq c \cdot |\theta_t^* - \theta_t|_1.$$

Definition 4 ((k,γ)-sparsity [Restated]) *A federated learning protocol $d = (\lambda, \mathcal{G}, \mathcal{A})$ is (k,γ)-sparse on a dataset D if for all $u_t = \mathcal{G}(\theta_{t-1}, D)$ generated during the process of training on D $\mathcal{A}(u_t)$ only has k non-zero elements and we have*

$$|\mathcal{A}(u_t) - u_t|_1 \leq \gamma.$$

We will use this definition in our following Theorem. In Subsection A.1.1 we explore the sparsity of the SparseFed algorithm.

Definition 2 (Certified radius [Restated]) *Let f be a protocol and $f^* \in F(\rho)$ be a poisoned version of the same protocol. Let θ_T, θ_T^* be the benign and poisoned final outputs of the above protocols on a dataset D. We call R a certified radius for f on a dataset D if $\forall f^* \in F(\rho); R(\rho) \geq |\theta_T - \theta_T^*|_1$.*

Theorem 4. *Let f be a c-coordinatewise-Lipschitz and (k,γ)-sparse protocol on a dataset D. Let $w = min(d, 2k)$ then $R(\rho) = \Lambda(T)(1 + wc)^{\Lambda(T)}(\rho + 2\gamma)$ is a certified radius for f.*

Before proving the above theorem, note that that Theorem 4 immediately implies Theorems 1 and 2.

Proof. Let $f^* = (\mathcal{G}^*, \mathcal{A}, \lambda) \in f(\rho)$ be an arbitrary ρ-poisoned version of f. We first define two sequence of models $(\theta_b^0, \ldots, \theta_b^T)$ and $(\theta^0, \ldots, \theta^T)$ where θ_b^t is the model trained in the first t iterations through the benign (non-poisoned) gradient oracle $\mathcal{G}$ and θ^t is the model trained in the first t iterations through a ρ poisoned aggregation $\mathcal{G}^*$. Also, we define $u_b^1, \ldots, u_b^T$ and $u^1, \ldots, u^t$ to be the update vectors that the benign oracle $\mathcal{G}$ would produce on models $\theta_b^1, \ldots, \theta_b^{T-1}$ and $\theta^1, \ldots, \theta^{T-1}$, respectively. We also define $\widehat{u}^1, \ldots, \widehat{u}^T$ to be the output of the adversarial gradient oracle $\mathcal{G}^*$ on models $\theta_1, \ldots, \theta_{T-1}$. By the definition of ρ-poisoning, we have $|\widehat{u}^t - u^t|_1 \leq \rho$.

Note that by the definition of coordinatewise Lipschitzness, for any coordinate $i \in [d]$ we have

$$|u^t[i] - u_b^t[i]| \leq c|\theta^{t-1} - \theta_b^{t-1}|_1.$$

Now, we use the triangle inequality to connect the distance between θ^t and θ_b^t to that of the previous round as follows

$$\left|\theta^t - \theta_b^t\right| = \left|\theta^{t-1}[i] - \lambda(t)\mathcal{A}(\widehat{u}^t) - \theta_b^{t-1} + \lambda(t)\mathcal{A}(u_b^t)\right| \leq \left|\theta^{t-1} - \theta_b^{t-1}\right| + \lambda[t]\left|\mathcal{A}(\widehat{u}^t) - \mathcal{A}(u_b^t)\right| \tag{1}$$

Now we prove the following Lemma that bounds the difference between updates on the benign and poisoned models.

Lemma 1. *We have*

$$|\mathcal{A}(\widehat{u}^t) - \mathcal{A}(u_b^t)|_1 \leq \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + 2\gamma$$

where $I = \{j \in [d] \text{ s.t. } \mathcal{A}(\widehat{u}^t)[j] \neq 0 \text{ or } \mathcal{A}(u_b^t)[j] \neq 0\}$.

Proof. Let τ_1 and τ_2 be two vectors such that $\tau_1[i] = 1$ if $\mathcal{A}(\widehat{u}^t)[i] \neq 0$ and $\tau_1[i] = 0$ otherwise. Similarly, $\tau_2[i] = 1$ if $\mathcal{A}(u_b^t)[i] \neq 0$ and $\tau_2[i] = 0$ otherwise. Let I' be the locations where $\tau_1[i] = 1$ and $\tau_2[i] = 1$. Now we have

$$\begin{aligned}
|\mathcal{A}(u^t) - \mathcal{A}(u_b^t)| &= \sum_{i=1}^{d} |\widehat{u}^t[i]\tau_1[i] - u_b^t[i]\tau_2[i]| \\
&= \sum_{i \in I'} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i]\tau_1[i] - u_b^t[i]\tau_2[i]| \\
&\leq \sum_{i \in I'} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i]\tau_1[i] - u_b^t[i]\tau_1[i]| \\
&\quad + \sum_{i \in I \setminus I'} |\widehat{u}^t[i]\tau_2[i] - u_b^t[i]\tau_2[i]| + \sum_{i \in I \setminus I'} |u_b^t[i]\tau_1[i] - \widehat{u}^t[i]\tau_2[i]| \\
&= \sum_{i \in I'} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i] - u_b^t[i]|(\tau_1[i] + \tau_2[i]) + \sum_{i \in I \setminus I'} |u_b^t[i]\tau_1[i] - \widehat{u}^t[i]\tau_2[i]| \\
&= \sum_{i \in I'} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i] - u_b^t[i]| + \sum_{i \in I \setminus I'} |u_b^t[i]\tau_1[i] - \widehat{u}^t[i]\tau_2[i]| \\
&= \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |u_b^t[i]\tau_1[i] - \widehat{u}^t[i]\tau_2[i]| \\
&\leq \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i]\tau_2[i]| + |u_b^t[i]\tau_1[i]| \\
&= \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + \sum_{i \in I \setminus I'} |\widehat{u}^t[i](1 - \tau_1[i])| + \sum_{i \in I \setminus I'} |u_b^t[i](1 - \tau_2[i])| \\
&\leq \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + |\widehat{u}^t - \mathcal{A}(\widehat{u}^t)| + |u_b^t - \mathcal{A}(u_b^t)| \\
&\leq \sum_{i \in I} |(\widehat{u}^t[i] - u_b^t[i])| + \gamma + \gamma.
\end{aligned}$$

which finishes the proof. □

Now, based on Lemma 1, the (k, γ) sparsity of f, the Lipschitzness, and since $|I| \leq w$ we conclude that

$$|\mathcal{A}(\widehat{u}^t) - \mathcal{A}(u_b^t)|_1 \leq \sum_{i \in I} |(u^t[i] - u_b^t[i])| + \sum_{i \in I} |(\widehat{u}^t[i] - u^t[i])| + 2\gamma \leq wc|\theta^{t-1} + \theta_b^{t-1}| + \rho + 2\gamma. \tag{2}$$

By plugging this into Equation 1 we get

$$\left|\theta^t - \theta_b^t\right| \leq (1 + wc\lambda(t))\left|\theta^{t-1} - \theta_b^{t-1}\right| + (\rho + 2\gamma)\lambda(t). \tag{3}$$

Now using this equation, we inductively prove the Theorem. Assume for $T - 1$ the statement of theorem holds. By Equation 3 and the induction hypothesis we have

$$\left|\theta^T - \theta_b^T\right| \leq (1 + wc\lambda(T))\Lambda(T-1)(1 + wc)^{\Lambda(T-1)}(\rho + 2\gamma) + (\rho + 2\gamma)\lambda(T). \tag{4}$$

Then, by Bernouli's inequality we have

$$\begin{aligned}\left|\theta^T - \theta_b^T\right| &\leq \Lambda(T-1)(1+wc)^{\Lambda(T-1)+\lambda(T)}(\rho+2\gamma) + (\rho+2\gamma)\lambda(T) \\ &= \Lambda(T-1)(1+wc)^{\Lambda(T)}(\rho+2\gamma) + (\rho+2\gamma)\lambda(T) \\ &\leq (\Lambda(T-1)+\lambda(T))(1+wc)^{\Lambda(T)}(\rho+2\gamma) \\ &\leq \Lambda(T)(1+wc)^{\Lambda(T)}(\rho+2\gamma).\end{aligned}$$

And this finishes the proof. □

Remark 2 (How does sparsity help robustness?)**.** *In our analysis of the effect of sparsity on the certified radius, we first proved Lemma 1 to show that the effect of poisoning at each iteration is bounded by $\rho + 2\gamma$. Note that if we just use the identity aggregation (which is not sparse), we would get a better bound of ρ for each iteration. Then, how are we getting a better final bound with sparsity? We emphasize that the goal of sparsity is to bound the "propagation error" during the entire training. The improved bound is achieved because of the fact that sparsification removes most of the noise that that poisoning can cause on the updates of benign parties. As we see in Table 2, our approach can actually reduce the final distance between adversarial and benign models which verifies our theory and shows the importance of considering the propagation error.*

A.1.1 SparseFed is a sparse protocol

The definition of sparsity requires that the aggregation protocol to only update k coordinates. The top_k operator, by definition, only updates k operators. The only thing that remains is to show that SparseFed can achieve a small *gamma* as well. Here, we bound the *gamma* for SparseFed, given a certain loss rate that is a known a priory.

Definition 5. *[loss rate ω_k for top-k operator] Let ω_k be the fraction of l_1 mass of information lost via top_k, where $top_k(u)$ recovers a $1-\omega_k$ fraction of the l_1 mass of u. For any model M, any $i \in [T]$ and update vector $(u^1, \ldots, u^n)$ calculated by all parties (including benign and adversarial gradients), and memory W, we have:*

$$|top_k(u^t + W^t)|_1 \geq (1-\omega_k)|u^t + W^t|_1. \tag{5}$$

When clear from the context, we use ω instead of ω_k.

We first show that the size of memroy vector W is bounded.

Lemma 2. *Let W_t and W_t^b be the memory vector at round t for the benign and poisoned protocol respectively. After each iteration we have*

$$|W^i| \leq L\sqrt{d} \cdot \frac{\omega}{1-\omega}$$

and

$$|W_b^i| \leq L\sqrt{d} \cdot \frac{\omega}{1-\omega}$$

where L is the ℓ_2 clipping threshold.

Proof. We prove this by induction on i. The proof is similar for W_i and W_b^i so we only prove it for W^i. For $i = 0$ the induction hypothesis is correct. Now assume the hypothesis is correct for round $i-1$, namely

$$|W^{i-1}| \leq L\sqrt{d} \cdot \frac{\omega}{1-\omega}.$$

For round i we have

$$|W^i| = |W_{i-1} + u_{i-1} - top_k(W_{i-1} + u_{i-1})| \leq \omega(|W_{i-1} + u_{i-1}|) \leq \omega(L\sqrt{d} \cdot \frac{\omega}{1-\omega} + L\sqrt{d}) = L\sqrt{d} \cdot \frac{\omega}{1-\omega}$$

which finishes the proof. □

Now we show that after applying top_k and memory, we do not deviate much from the original gradient (i.e. γ is small).

Lemma 3. *Let* $\gamma = 2L\sqrt{d}\frac{\omega}{1-\omega}$, *we have*

$$|top_k(u_t + W) - u_t|_1 \leq \gamma.$$

Proof. Given that the loss rate of the top_k is ω, we have

$$|top_k(u_t + W) - u_t - W|_1 \leq \omega|u_t + W| \leq \omega(|u_t| + |W|) \leq L\sqrt{d}\frac{\omega}{1-\omega}.$$

Therefore, we have

$$|top_k(u_t + W) - u_t|_1 \leq |top_k(u_t + W) - u_t - W|_1 + |W|_1 \leq 2L\sqrt{d}\frac{\omega}{1-\omega}.$$

□

A.2 Convergence analysis of `SparseFed`

We first restate the convergence of Error Feedback SGD (EF-SGD) of (Karimireddy et al., 2019) and then analyze `SparseFed` under this framework.

A.2.1 Analysis of Error Feedback SGD

Algorithm 4 EF-SGD

Input: learning rate γ, compressor $\mathcal{C}(\cdot), x_0 \in \mathcal{R}^d$
$e_0 = 0 \in \mathcal{R}^d$
for $t = 0, \cdots, T-1$ **do**
$g_t :=$ stochasticGradient(x_t)
$p_t := \gamma g_t + e_t$
$\delta_t := \mathcal{C}(p_t$
$x_{t+1} := x_t - \delta_t$
$e_{t+1} := p_t - \delta_t$
end for

Assumption 3 (Compressor)**.** *An operator* $\mathcal{C} : \mathcal{R}^d \to \mathcal{R}$ *is a* δ*-approximate compressor over* $\mathcal{Q}$ *for* $\delta \in [0, 1]$ *if*

$$\|\mathcal{C}(x) - x\|_2^2 \leq (1 - \delta)\|x\|_2^2, \forall x \in \mathcal{Q}$$

Assumption 4 (Smoothness)**.** *A function* $f : \mathcal{R}^d \to \mathcal{R}$ *is L-smooth if for all* $x, y \in \mathcal{R}^d$ *the following holds:*

$$|f(x) - (f(x) + \langle \nabla f(x), y - x\rangle)| \leq \frac{L}{2}\|y - x\|_2^2$$

Assumption 5 (Moment Bound)**.** *For any x, our query for a stochastic gradient returns g such that*

$$\mathbb{E}[g] = \nabla f(x) and \mathbb{E}\|g\|_2^2 \leq \sigma^2$$

Theorem 5 (Non-convex convergence of EF-SGD)**.** *Let* $x_{t t\geq 0}$ *denote the iterates of Algorithm 4 for any step-size* $\gamma > 0$*. Under Assumptions 3, 4, 5,*

$$\min_{t \in T} \mathbb{E}[\|\nabla f(x_t)\|^2] \leq \frac{2(f(x_0) - f^*)}{\gamma(T+1)} + \frac{\gamma L\sigma^2}{2} + \frac{4\gamma^2 L^2\sigma^2(1-\delta)}{\delta^2}$$

A.3 Analysis of `SparseFed`

To prove the convergence of `SparseFed`, we simply use Theorem 5 and prove that the necessary assumptions are satisfied. That is, we prove that `SparseFed` fits into the theoretical framework of (Karimireddy et al., 2019).

We know already that the top-k operator we use is a δ-approximate compressor (Karimireddy et al., 2019), which satisfies the first assumption. The second and third assumptions, we directly reproduce for the gradient oracle that represents the individual device gradients.

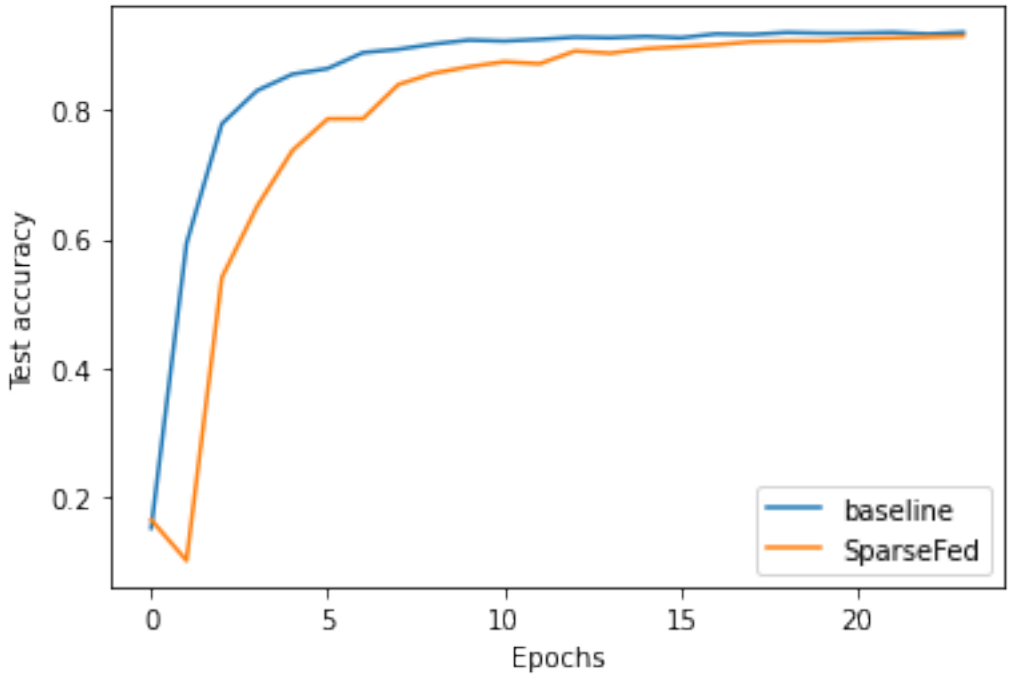


Figure 4: `SparseFed` converges at the same rate as the baseline (FedAvg) on CIFAR10 in the cross-device setting

Assumption 6 (Smoothness). *$\mathcal{L}$ is ℓ-smooth if* $\forall x, y \in \mathcal{R}^d \ |\mathcal{L}(x) - (\mathcal{L}(y) + \langle \nabla \mathcal{L}(x), x - y \rangle)| \leq \frac{\ell}{2} \|x - y\|_2^2$

Assumption 7 (Moment Bound). *For any x, our oracle returns $\boldsymbol{g}$ s.t. $\mathbb{E}[\boldsymbol{g}] = \nabla \theta(x)$ and $\mathbb{E} \|\boldsymbol{g}\|_2^2 \leq \sigma^2$*

Because `SparseFed` is essentially EF-SGD for federated learning, it only remains to show that the federated setting does not complicate our analysis. The federated setting comes with the complications of LocalSGD, namely multiple local epochs, and the non-i.i.d. distribution of data across devices.

As per the statement of Theorem 3, we only prove guarantees for $\tau = 1$; that is, we only do a single local epoch. Prior work has evidenced the challenges of analyzing convergence of LocalSGD in the presence of non-i.i.d. data (Li et al., 2019), and we find empirically that multiple local epochs are unfavorable for both convergence and robustness in the cross-device setting that is the focus of our work.

Therefore, `SparseFed` directly fits into the theoretical framework of (Karimireddy et al., 2019) and Theorem 5 proves the convergence of `SparseFed`.

In Fig 4 we empirically validate the speed of convergence of `SparseFed` and find that it converges at the same rate as `FedAvg`, even in the presence of attackers.

A.3.1 Training a single layer neural network with SGD

Example 2 (Training a single layer neural network with SGD). *In this example, we compute the coordinatewise Lipschitz constant of the SGD protocol for a single layer neural network defined as $\sigma(\theta x)$, where σ is the softmax function and $\theta \in \mathcal{R}^d$ are the network parameters. For cross-entropy loss-based training using dataset D, we show that the constant $c = \frac{1}{4}$. Formally,*

$$\sup_{D \in Z, \theta_1, \theta_2 \in \mathcal{M}} |\mathcal{G}(\theta_1, D)[i] - \mathcal{G}(\theta_2, D)[i]|_1 \leq \frac{1}{4} |\theta_1 - \theta_2|_1 \text{ for any coordinate index } i \in [d]$$

Without loss of generality, we assume that dataset D is comprised of samples of the form (x, y), where $x \in [0, 1]^m$, and $y \in \{0, 1\}^C$ is the one-hot encoded representation of any of the C classes. For the single layer neural network, the model parameters are denoted by $\theta \in \mathcal{R}^{C \times m}$, and the softmax layer by the function $\sigma(\cdot)$. The neural network can thus be represented as $\Phi(x, \theta) = \sigma(\theta x)$.

We define $g(\theta, x) = \frac{\partial \mathcal{L}(\Phi(x,\theta),y)}{\partial \theta}$ where $\mathcal{L}$ is the softmax cross entropy loss function. For the SGD protocol, $\mathcal{A}(u) = u$, and $\mathcal{G}(\theta, D) = g(\theta, x)$. Our goal is to find a Lipschitz constant L such that, for all indices $i \in [C]$ and $j \in [m]$,

$$\sup_{x \in D, \theta_1, \theta_2} \frac{|g(\theta_1, x)_{ij} - g(\theta_2, x)_{ij}|_1}{|\theta_1 - \theta_2|_1} \leq L \tag{6}$$

We define intermediate variable $z = \theta x$ and the neural network output distribution $p = \sigma(z)$, such that both $p, z \in \mathbb{R}^C$. Note, for a given target class t, the cross entropy loss function $\mathcal{L}(p, y) = -\log(p_t)$ where $p_t = \frac{e^{z_t}}{\sum_j e^{z_j}}$. Thus,

$$g(\theta, x)_{ij} = \frac{\partial \mathcal{L}}{\partial \theta_{ij}} = \sum_{c=1}^{C} \frac{\partial \mathcal{L}}{\partial z_c} \frac{\partial z_c}{\partial \theta_{ij}}. \tag{7}$$

Computing the terms of (7), we have $\frac{\partial \mathcal{L}}{\partial z_c} = p_t - 1$ for $c = t$; and $\frac{\partial \mathcal{L}}{\partial z_c} = p_c$ otherwise; and $\frac{\partial z_c}{\partial \theta_{ij}} = x_j$. Thus,

$$\begin{aligned} g(x, \theta)_{ij} &= x_j(p_t - 1) \quad \text{for i = t} \\ &= x_j p_i \quad \text{for } i \neq t \end{aligned} \tag{8}$$

We compute the Hessian of $g(x, \theta)_{ij}$ as:

$$\begin{aligned} \frac{\partial g(x, \theta)_{ij}}{\partial \theta_{kl}} &= x_j p_t (1 - p_t) x_l \quad \text{for } k = t \\ &= x_j p_k (1 - p_k) x_l \quad \text{for } k \neq t \end{aligned} \tag{9}$$

where $k \in [C], l \in [m]$. The maximum value of the Hessian in (9), occurs at $x_j = x_l = 1$, and $p_t = p_k = \frac{1}{2}$. Thus,

$$\begin{aligned} \max_{i,j,k,l} \frac{\partial g(x, \theta)_{ij}}{\partial \theta_{kl}} &\leq \frac{1}{4} \quad \text{for } k = t \\ &\leq \frac{1}{4} \quad \text{for } k \neq t \end{aligned} \tag{10}$$

To obtain the Lipschitz constant, we first define the function

$$h(t) = g((1-t)\theta_1 + t\theta_2, x)_{ij} \text{ where } t \in [0, 1]$$

Thus, $h(0) = g(\theta_1, x)_{ij}$ and $h(1) = g(\theta_2, x)_{ij}$. Since, the function $h(t)$ is differentiable everywhere in $(0, 1)$, using Mean Value Theorem Rudin (1976), we know that there exists a point $t^* \in (0, 1)$ such that:

$$h(1) - h(0) \leq h'(t^*) \text{ where } h'(t) = (\theta_2 - \theta_1) g'((1-t)\theta_1 + t\theta_2, x)_{ijkl}. \tag{11}$$

Rewriting (6), we get

$$\begin{aligned} & \sup_{x \in D, \theta_1, \theta_2} |g(\theta_1, x) - g(\theta_2, x)|_1 \\ \leq & \sup_{x \in D, \theta_1, \theta_2} |\max_{i,j} \{g(\theta_1, x)_{ij} - g(\theta_2, x)_{ij}\}|_1 \end{aligned}$$

Let i^*, j^* correspond to the indices where the maximum in the above equation occurs. Combining (10) and (11), we get:

$$\sup_{x \in D, \theta_1, \theta_2} |g(\theta_1, x)_{i^* j^*} - g(\theta_2, x)_{i^* j^*}|_1 \leq \frac{1}{4} |\theta_1 - \theta_2|_1 \tag{12}$$

Comparing (12) with (6) we get $c = \frac{1}{4}$.

A.3.2 Computing the certified radius

Algorithm 5 calculates the maximum distance between the poisoned and benign models, based on the number of attackers, protocol parameters c, λ defined in Definition 3, number of iterations T, clipping parameter L, the dimension of the model d and sparsification parameter k. The correctness of this procedure follows from the proof of Theorem 2.

Algorithm 5 Radius calculation

Input: poisoning parameter ρ, number of model weights to update each round k, number of timesteps T, decay function λ, model parameters θ, test dataset $(x, y)_{i=1}^{m}$, Lipschitzness c, error γ
$r = 0$
$\beta = \epsilon + \gamma$
for $t = 1, 2, \cdots T$ **do**
$\alpha = 1 + 2\lambda(t)ck$
$r = r * \alpha + \lambda(t)\beta$
end for
Output: radius r

B Methods and Metrics

B.1 Methods

In this section we give a detailed treatment of the methods we compare. We run all experiments on commercially available NVIDIA Pascal GPUs. With this in mind, all implementations are optimized to run on a single GPU and all our experiments can be reproduced within a few hours (`SparseFed`) or days (Byzantine-robust aggregation methods).

B.1.1 FedAvg

We use the standard implementation of federated averaging McMahan et al. (2017), described in Algorithm 6, as the baseline for all defenses in this work. The first major departure is the use of ℓ_2 clipping, which is in place whenever we refer to the ℓ_2 clipping defense. When we refer to an "undefended" system, we do not make use of ℓ_2 clipping. As an implementation detail, we average updates and not individual models because we employ norm clipping in all defenses and clipping model parameters wholesale is more difficult than clipping updates. The second major departure is the use of server-side momentum, which has empirically been shown to improve convergence (Rothchild et al., 2020).

Local epochs make outlier detection difficult: From an adversarial perspective, `FedAvg` has a key vulnerability: the use of multiple local epochs τ, which is a design choice to amortize communication costs. As the number of local epochs $\tau \to \infty$, individual updates from benign devices become further apart in ℓ_2 space. This makes it difficult for Byzantine-robust aggregation rules such as Bulyan and Krum to identify outliers, because both attacker updates and benign updates are very far apart. Therefore, when benign devices do multiple local epochs, attackers are more likely to remain undetected by outlier detection methods. To ensure we are comparing against the strongest versions of the Byzantine-robust aggregation rules possible, we use $\tau = 1$.

Local epochs amplify existing vulnerabilities: Even when the number of local epochs $\tau = 1$, `FedAvg` with ℓ_2 clipping does not reduce to distributed SGD because devices scale their updates by the learning rate before doing norm clipping. This presents an opportunity for the attacker: when the global learning rate is very small, such as towards the end of training when using a typical decaying learning rate schedule, the updates of most benign devices will have ℓ_2 norm close to 0. Here, the attacker can simply project their update to the perimeter of the ℓ_2 norm constraint and essentially have an update which is hundreds of times larger than the rest of the benign devices, which enables them to perform model replacement. In Appendix B.2 we propose and evaluate a method to mitigate this vulnerability.

Model replacement: Model replacement has already been proposed as an attack strategy in prior work (Bagdasaryan et al., 2020) because state of the art models often converge to a stationary point towards the end of training. This vulnerability is simply amplified in federated learning, because all federated learning deployments today make use of multiple local epochs, as update communication is the system bottleneck.

Uncompressed FL is more robust than `FedAvg`: In Table 7 we show that using distributed SGD as the backbone algorithm rather than `FedAvg` has a marked impact on the attack accuracy. We refer to this regime as "uncompressed FL" because we are not compressing communication costs, and note that this regime is strictly unrealistic. Even in the uncompressed regime, the attack still functions via model replacement, because the benign objective reaches a stationary point and the gradients from benign devices are very small. We note that while the attack does not reach 100% accuracy against the ℓ_2 defense in this setting, when we incorporate minor

Algorithm 6 `SparseFed`

Input: learning rate λ, number of timesteps T, local batch size b, number of devices selected per round n, norm clipping parameter L, local epochs τ, local learning rate γ
Initialize model θ_0 using the same random seed on the devices and aggregator
Initialize momentum vector $\mathrm{R}^t = 0$
for $t = 1, 2, \cdots T$ **do**
  Randomly select n devices $d_1, \ldots d_n$
  loop {In parallel on devices $\{d_i\}_{i=1}^n$}
    Download new model weights $\theta_t = \theta$
    for $m \in \tau$ **do**
      Compute gradient $\mathrm{g}_t^i = \frac{1}{b}\sum_{j=1}^{l} \nabla_\theta \mathcal{L}(\theta^t, \mathrm{D}_j)$
      Accumulate gradient $\theta_t = \theta_t - \gamma(t,m)\mathrm{g}_t^i$
    end for
    Compute update $\mathrm{u}_t^i = \theta_t - \theta$
    Clip update $\mathrm{u}_t^i = \mathrm{u}_t^i \cdots \min(1, \frac{L}{|\mathrm{u}_i^t|_2})$
  end loop
  Aggregate gradients $\mathrm{u}_t = \frac{1}{n}\sum_{i=1}^{n} \mathrm{u}_t^i$
  Momentum $\mathrm{R}^t = 0.9\mathrm{R}^t + \mathrm{u}_t$
  Update $\theta_{t+1} = \theta_t - \lambda(t)\mathrm{R}_t$
end for
Output: $\{\theta^t\}_{t=1}^T$

Table 7: Attack accuracy decrease for ℓ_2 norm clipping and `SparseFed` when doing uncompressed FL (SGD) *as compared to using* ***`FedAvg`***. CIFAR10, 1e4 clients, 200 attackers.

Defense	Test acc	Attack acc (decrease)	Attack acc
ℓ_2	84.07 ±0.7	34.0 ±6	66.0 ±6
`SparseFed`	81.72 ±0.9	20.0 ±5	5.6 ±1

adjustments to the attack (Appendix B.8) we can reach 100% accuracy; `SparseFed` still functions well as a defense.

In Appendix B.9 we introduce a communication-efficient variant of `SparseFed` which can drop the use of multiple local epochs altogether, and therefore obtains improved robustness empirically.

Momentum is necessary for convergence: As an implementation detail, we employ momentum factor masking (Rothchild et al., 2020) in `SparseFed`. This entails maintaining a momentum buffer which we zero out similar to the error feedback vector. We provide the momentum enabled algorithm in Algorithm 7 We do not analyze the role of momentum in robustness, but it is empirically useful.

In Table 8 we see that without the use of momentum, neither the model nor the attack converge when using just `FedAvg` with ℓ_2 clipping. This is what `SparseFed` reduces to as $k \to d$, because at every iteration we zero out the entire momentum buffer.

B.1.2 The Attack

In Algorithm 8 we provide the model poisoning attack that we use throughout this work. This attack is similar to the PGD attack proposed in prior work (Sun et al., 2019), with the addition of the attacker batch size parameter which enables us to poison models with larger auxiliary datasets. In Appendix B.8 we provide detailed analysis on how we choose the attacker batch size and number of PGD epochs. The attackers sample data from the

Table 8: Test/Attack accuracy decrease for ℓ_2 norm clipping when not using momentum. CIFAR10, 1e4 clients, 200 attackers.

Defense	Test Acc (decrease)	Test acc	Attack acc (decrease)	Attack acc
ℓ_2	31.08 ±0.7	53.14 ±1.7	61.4 ±6	4.6 ±1

Algorithm 7 `SparseFed`

Input: number of coordinates to update each round k, learning rate λ, number of timesteps T, local batch size b, number of devices selected per round n, norm clipping parameter L, local epochs τ, local learning rate γ
Initialize model θ_0 using the same random seed on the devices and aggregator
Initialize memory vector $\mathrm{W}_t = 0$, momentum vector $\mathrm{R}^t = 0$
for $t = 1, 2, \cdots T$ **do**
  Randomly select n devices $d_1, \ldots d_n$
  loop {In parallel on devices $\{d_i\}_{i=1}^n$}
    Download new model weights $\theta_t = \theta$
    for $m \in \tau$ **do**
      Compute gradient $\mathrm{g}_t^i = \frac{1}{b}\sum_{j=1}^{l} \nabla_\theta \mathcal{L}(\theta^t, \mathrm{D}_j)$
      Accumulate gradient $\theta_t = \theta_t - \gamma(t,m)\mathrm{g}_t^i$
    end for
    Compute update $\mathrm{u}_t^i = \theta_t - \theta$
    Clip update $\mathrm{u}_t^i = \mathrm{u}_t^i \cdots \min(1, \frac{L}{|\mathrm{u}_i^t|_2})$
  end loop
  Aggregate gradients $\mathrm{u}_t = \frac{1}{n}\sum_{i=1}^{n} \mathrm{u}_t^i$
  Momentum: $\mathrm{R}^t = 0.9 \cdot R^{t-1} + u^t$
  Error feedback: $\mathrm{W}_t = \mathrm{R}_t + \mathrm{W}_t$
  Extract top_k: $\Delta_t = top_k(\mathrm{W}_t)$
  Error accumulation: $\mathrm{W}_{t+1} = \mathrm{W}_t - \Delta_t$
  Update $\theta_{t+1} = \theta_t - \lambda(t)\Delta_t$
end for
Output: $\{\theta^t\}_{t=1}^T$

"auxiliary dataset", a dataset which is composed of datapoints with their labels flipped that the attacker uses as a proxy to formulate the poisoned gradient.

Algorithm 8 Attack

Input: learning rate η, local batch size ℓ, norm clipping parameter L, number of local epochs e
1: This procedure is used by all attackers in a round to ensure that they upload the same update
2: **for** number of PGD epochs $e_i \in e$ **do**
3: Compute stochastic gradient g_i^t on batch B_i of size ℓ: $\mathrm{g}_i^t = \frac{1}{\ell}\sum_{j=1}^{l} \nabla_{\mathrm{M}} \mathcal{L}(\mathrm{M}_{e_i}^t, \mathrm{D}_j)$
4: Update local model $\widehat{M}_{e_{i+1}}^t = \mathrm{M}_{e_i}^t - \eta \mathrm{g}_i^t$
5: Project accumulated update onto the perimeter of the ℓ_2 constraint $\mathrm{M}_{e_{i+1}}^t = M_0^t - CLIP(\widehat{M}_{e_{i+1}}^t - M_0^t)$
6: **end for**
Output: M_e^t

B.1.3 Byzantine-resilient defenses

Every algorithm we describe in this section is implemented via replacing line 15 in Algorithm 6. This introduces additional computational complexity into the aggregation step, which is the bottleneck in federated learning. This complexity can be minor (trimmed mean) or it can be massive (Bulyan). Our experiments with Bulyan take approximately 20× longer to run than our experiments with `SparseFed`; because these experiments are so computationally infeasible, where possible we omit Bulyan from comparisons in the rest of the Appendix. These defenses as initially proposed do not make use of ℓ_2 norm clipping, but because we use ℓ_2 clipping in the baseline defense, and because it benefits all defenses (Appendix B.2), the input gradients to all the aggregation rules are already clipped.

Trimmed mean: In Algorithm 9 we see that trimmed mean iteratively rejects outliers at each coordinate until it has eliminated $2f$ coordinates. If the attacker's updates have extremely small or large values, then trimmed mean will mitigate the attack. However, if most of the attacker's updates are close to 0 at many coordinates, then trimmed mean will not mitigate the attack. This is the phenomena observed in (Bhagoji et al., 2019); the attacker's updates are far sparser than benign updates, which in turn means that most coordinate values are 0 and thus trimmed mean is ineffective.

Coordinate median: Coordinate median is simply implemented by returning the coordinatewise median instead of the mean. This does not converge because of the gap between median and mean (Chen et al., 2020;

Algorithm 9 Trimmed mean

Input: number of compromised devices f, set of individual updates $U = \left\{u^t\right\}_{i=1}^{n}$
for number of compromised devices f **do**
  for each coordinate $\{c\}_{j=1}^{d}$ **do**
    $U_c \leftarrow U_c \setminus \min U_c$
    $U_c \leftarrow U_c \setminus \max U_c$
  end for
end for
Aggregate remaining updates $\mathrm{u}^t = \frac{1}{n-2f} \sum_{i=1}^{n-2f} \mathrm{u}_i^t$
Output: u^t

Muñoz-González et al., 2019; Yin et al., 2019).

Algorithm 10 Krum

Input: number of compromised devices f, set of individual updates $U = \left\{u^t\right\}_{i=1}^{n}$
for each update u_i^t **do**
  $U_i = U$
  for f+2 **do**
    $U_i = U_i \setminus \arg\max_{u_j^t \in U_i} \left\|u_j^t - u_i^t\right\|$
  end for
  $S_i = \sum_{u_j \in U_i} \left\|u_j^t - u_i^t\right\|$
end for

Output: $\mathrm{u}^t = \arg\min_{u \in U} S$

Krum: Algorithm 10 implements Krum, which attempts a Byzantine-resilient variant of the barycentric aggregation rule (Blanchard et al., 2017a). Krum selects a single update from the aggregated set to update the global model. In the cross-device federated setting, this will never converge. Essentially, we will be using SGD instead of minibatch SGD, and it will take us 100× longer to do one pass over the entire dataset. Because Bulyan uses Krum and trimmed mean, we do not analyze Krum in isolation in depth.

Algorithm 11 Bulyan

Input: number of compromised devices f, set of individual updates $U = \left\{u^t\right\}_{i=1}^{n}$
$\Theta = n - 2f$
$S = \emptyset$
while $|S| < \Theta$ **do**
  $p = \texttt{KRUM}(U, f)$
  $U \leftarrow U \setminus p$
  $S \leftarrow S \cup p$
end while
Output: $\mathrm{u}^t = \texttt{TRIMMEAN}(S, f)$

Bulyan: Algorithm 11 describes Bulyan (Mhamdi et al., 2018) implemented with Krum as the base aggregation rule. Bulyan builds a set by iteratively applying Krum onto the set of aggregated updates, and then returns the trimmed mean of this set. If Krum selects the attacker, we already know that trimmed mean is not likely to reject the attacker. Thus, it remains to intuit why Krum will select at least one attacker. In the non-i.i.d. setting, benign update vectors are sufficiently far away that a very small number of colluding attackers at each iteration can minimize their distance to all other vectors by sending the same update, which ensures that they have a distance of 0 from each other. Thus, Krum selects at least one attacker, and Bulyan fails, as we show in our experiments.

It is readily apparent that for large values of n, Bulyan is fairly computationally inefficient even when implemented efficiently. Although the asymptotic complexity of Bulyan is the same as that of Krum, the constant factor is quite large ($n = 100$).

B.1.4 SparseFed

In the main body we include the algorithm for SparseFed implemented with true top-k. As an implementation detail, the algorithm presented in the main body is in the uncompressed regime, where we do not perform any local epochs and the learning rate is multiplied after the top-k coordinates are extracted.

Algorithm 12 SparseFed implemented with FetchSGD instead of global top-k

Input: number of model weights to update each round k
Input: learning rate η
Input: norm clipping parameter L
Input: number of timesteps T
Input: momentum parameter ρ, local batch size ℓ
Input: Number of clients selected per round W
Input: Sketching and unsketching functions $\mathcal{S}, \mathcal{U}$
Initialize S_u^0 and S_e^0 to zero sketches
Initialize model θ_0 using the same random seed on the devices and aggregator
for $t = 1, 2, \cdots T$ **do**
  Randomly select n devices $d_1, \dots d_n$
  loop {In parallel on devices $\{d_i\}_{i=1}^n$}
    Download new model weights $\theta_t = \theta$
    Compute gradient $\mathrm{g}_t^i = \frac{1}{b}\sum_{j=1}^{l} \nabla_\theta \mathcal{L}(\theta^t, \mathrm{D}_j)$
    Clip g_i^t according to L: $\mathrm{g}_i^t = \mathrm{g}_i^t * \min(1, \frac{L}{|\mathrm{g}_i^t|_2})$
    Sketch g_i^t: $\mathrm{S}_i^t = \mathcal{S}(\mathrm{g}_i^t)$ and send it to the Aggregator
  end loop
  Aggregate sketches $\mathrm{S}^t = \frac{1}{W}\sum_{i=1}^{W} \mathrm{S}_i^t$
  Momentum: $\mathrm{S}_u^t = \rho \mathrm{S}_u^{t-1} + \mathrm{S}^t$
  Error feedback: $\mathrm{S}_e^t = \eta \mathrm{S}_u^t + \mathrm{S}_e^t$
  Unsketch: $\Delta^t = \text{Top-k}(\mathcal{U}(\mathrm{S}_e^t))$
  Error accumulation: $\mathrm{S}_e^{t+1} = \mathrm{S}_e^t - S(\Delta^t)$
  Update $\theta^{t+1} = \theta^t - \Delta^t$
end for
Output: $\{\mathrm{w}^t\}_{t=1}^T$

FetchSGD: Algorithm 12 is the FetchSGD algorithm (Rothchild et al., 2020) combined with ℓ_2 clipping. FetchSGD approximates true top-k and has been empirically shown to be communication efficient; in Appendix B.9 we validate the robustness of SparseFed implemented with FetchSGD. Because SparseFed implemented with FetchSGD can achieve communication efficiency without the use of multiple local epochs, it has improved robustness over SparseFed implemented with true top-k, which still requires multiple local epochs for communication efficiency.

B.1.5 Adaptively choosing k in SparseFed

The hyperparameter k is critical for the convergence of SparseFed. In Algorithm 13 we provide an adaptive algorithm for selecting k. The algorithm requires as input the maximum information loss tolerance due to sparsification, and essentially just performs binary search over a range of reasonable values of k until finding the smallest k that does not lose "too much" information.

B.1.6 Metrics

In the main body, we mainly use the attack accuracy metric for the fixed cross-silo and cross-device settings. However, in the rest of the Appendix we do not always use this setting when it does not illustrate the full breadth of a trend, and we note that attack accuracy is not a perfect metric. For example, when trying to poison 1 point, the attackers can trivially obtain 100% attack accuracy, but this is not the case when they are trying to poison 100 points. Similarly, 100 attackers will have an easier time poisoning 1 point than 1 attacker will. To address these shortcomings, we introduce a new metric.

Outsized Impact Factor (OIF) We first define some notation. Let S be the set of agents participating in federated learning, and S_b the set of benign agents so that $I = \frac{|S \setminus S_b|}{|S|}$ is the influence of the attacker on the system, represented as the fraction of agents which are compromised. We propose that the baseline for any model poisoning attack should be for the attackers to be able to poison datapoints (e.g. flip the label on that datapoint)

Algorithm 13 Selecting k

Input: model θ, maximum information loss ω, number of model parameters d, number of iterations in an epoch r, number of gradients to sample n (more samples gives a better estimate of ω)

set initial k $k = \frac{d}{r}$
set initial realized information loss $\delta = \infty$
while $\delta > \omega$ **do**
  compute n sample minibatch gradients $\{g\}_{j=1}^{n} | g_j = \nabla_\theta \mathcal{L}(\theta, z_j)$
  extract top-k $\{u\}_{j=0}^{n} | u_j = top_k(g_j)$
  calculate average l_1 mass lost $\delta^* = \frac{1}{n} \sum_{j=1}^{n} |g_j - u_j|_1$
  update $\delta = \min(\delta, \delta^*)$
  if $\delta > \omega$ **then**
    $k = k + \frac{d}{r}$
  end if
end while

Output: k

Defense	Attack Accuracy (without)	Attack Accuracy (with)
Trimmed mean	100	81.4
Bulyan	100	81.8

Table 9: In the cross-device setting of CIFAR10, trimmed mean and Bulyan benefit greatly from the use of adaptive clipping.

$\widehat{X}_m$ proportional to their influence I. Therefore, if $|\widehat{X}_m|$ is the number of datapoints successfully poisoned and n is the total number of datapoints controlled by all agents in the system, we define $\frac{|\widehat{X}_m|}{I \cdot n}$, which is the ratio of datapoints successfully poisoned relative to the influence of the attacker, normalized by the size of the dataset, as the *outsized impact factor* (OIF). This quantity determines the extent to which the attacker is able to 'punch above its weight' in terms of impacting the final model to a larger extent than its influence would already allow.

Our standard for a successful attack is an OIF of 1. This means that the attacker can poison the same fraction of the dataset as of the client population they control. By using the OIF metric as a heuristic for attack success, we can easily compare the efficacy of attacks across parameter settings when different numbers of attackers are present.

B.2 Norm Clipping

Adaptive clipping to mitigate the vulnerability of `FedAvg` As we note in Appendix B.1.1, the key vulnerability of `FedAvg` is that benign devices multiply their gradients by a small learning rate that can vary over the course of training, which can make their gradients smaller than the specified ℓ_2 norm clipping bound when the learning rate is small (e.g. when warming up the learning rate schedule at the start of training). However, the attack is under no such compulsion, and this can present an easy vulnerability for the attacker. To mitigate this, we propose the use of an adaptive ℓ_2 clipping schedule which simply mirrors the learning rate schedule. At each iteration, before we clip the device gradient to the specified norm L, we scale L by the learning rate $L := L \cdot \lambda(t)$. In Table 9 we ablate the effectiveness of this on trimmed mean and Bulyan.

Sparsification needs norm clipping

We perform ablations of the central idea of the paper, sparsification as a defense against model poisoning attacks, with and without the use of ℓ_2 norm clipping.

In Fig. 5 we compare the efficacy of the combination of the distributed poisoning attack and the PGD attack against the top_k defense, with and without ℓ_2 clipping with parameter 3. We observe that when ℓ_2 clipping is in place, sparsification completely mitigates the attack. However, without any clipping the attacker is able to successfully flip the labels of their entire auxiliary dataset. This is because without any constraint on the norm of its update, the attacker can massively magnify its update and ensure that all the coordinates in the top_k are in the direction of the adversarial optimum.

Byzantine-Robust Aggregation Benefits from Norm Clipping The prior defenses we consider (Krum,

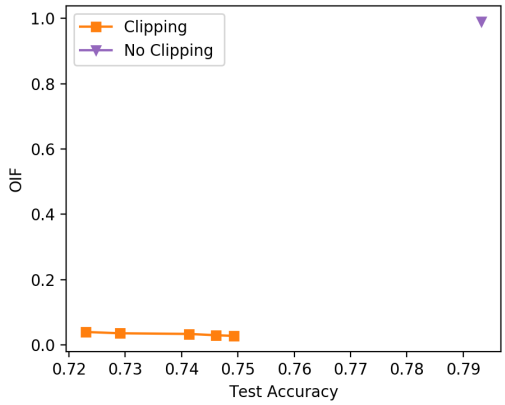


Figure 5: Pareto frontier of the combination of distributed poisoning and PGD attacks against `SparseFed` defenses with and without ℓ_2 clipping. Without ℓ_2 clipping, sparsification is entirely unable to mitigate the attack. CIFAR10, 10000 devices, 100 attackers.

Table 10: Implementing norm clipping greatly mitigates the effectiveness of the attack against Bulyan and trimmed mean when no colluding attackers are present. CIFAR10, 1e4 devices, 100 attackers.

Defense	**Test acc**	**Attack acc**
Bulyan (ℓ_2)	83.64	10.0
Bulyan	84.94	38.6
Trimmed Mean (ℓ_2)	77.42	71.6
Trimmed	81.99	100.0

Bulyan, trimmed mean, coordinate median) do not require norm clipping as part of the implementation. Norm clipping will either help the defense by limiting the impact of the attacker, in which case the server will enforce norm clipping, or it will hurt the defense by making the attack more stealthy, in which case the attacker will use norm clipping. In Table 10 we compare the changes in test and attack accuracy for Bulyan and trimmed mean when implementing norm clipping (Krum and coordinate median do not converge). As expected, norm clipping limits the impact of the attacker and helps Bulyan mitigate the attack when no colluding attackers are present.

B.2.1 Robustness in the DP defense costs accuracy

Prior work proposed combining ℓ_2 norm clipping and adding Gaussian noise to ensure robustness, similar to the process adopted in DP-SGD. In this work, we assume that practitioners will not be willing to adopt defenses which negatively impact the test accuracy of their models in scenarios where attackers are not present. We note that this is distinct from the accuracy degradation incurred from using a communication-efficient algorithm such as `FetchSGD` as a defense, or deploying DP-SGD to ensure differential privacy. In these cases, adversarial robustness can be seen as an additional benefit that 'comes for free'. However, the parameters that we find allow for some adversarial robustness at the cost of test accuracy for the DP defense do not actually enable any differential privacy. As a result, we do not use these parameters for most of our experiments because *we do not believe practitioners will adopt a defense which significantly negatively impacts their model performance.*

In Fig. 6 we examine the effect of adding noise $n \sim \mathcal{N}(0, \sigma^2 = 0.001)$. This noise parameter is identical to the one chosen in [1] As mentioned above, this amount of noise is entirely insufficient to ensure any differential privacy guarantees. We show the pareto frontier of the combination of distributed poisoning and PGD against the ℓ_2 defense with a parameter of 5, with and without noise addition. We find that when no attackers are present, adding noise reduces the test accuracy by a minimum of 12%, whereas not adding noise does not reduce the test accuracy at all. Therefore, while adding noise can make the model more robust, it is also guaranteed to significantly

[1] Sun et. al. 2019: https://arxiv.org/abs/1911.07963

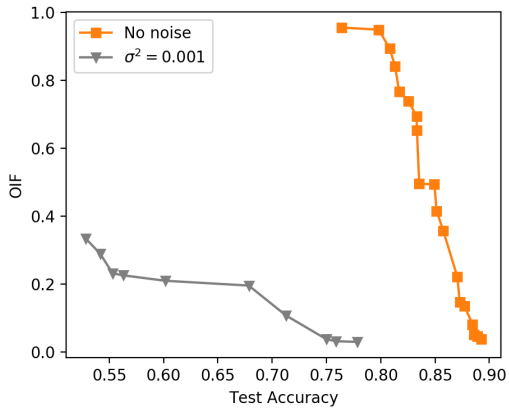


Figure 6: Pareto frontier of the ℓ_2 defense with clipping parameter 5, with and without noise addition, against the attack. Although noise addition can improve the robustness of the model to attackers, it also degrades test accuracy. In situations where no attackers are present, adding enough noise to mitigate any possible attackers will reduce the test accuracy by $> 10\%$. Because we do not expect practitioners will adopt any defense which is guaranteed to reduce the performance of their models by such a nontrivial amount, we do not use noise addition. (points with low OIF either do not make use of PGD or have too small batch sizes) CIFAR10, 10000 devices, 100 attackers.

degrade model performance. In keeping with the aforementioned systemic assumption that practitioners will not use defenses which damage model performance, we do not use noise addition in most experiments.

We nevertheless perform a comparison of the DP defense with ℓ_2 parameter 5 and noise addition with $\sigma^2 = 0.001$, against `SparseFed` in Fig. 7. Our findings reinforce our prior conclusions. While adding noise with strict clipping is sufficient to mostly mitigate the attack, it comes at the cost of an egregious 20% test accuracy drop. By comparison, `SparseFed` suffers little accuracy degradation and mitigates the attack even better.

B.3 Hyperparameter Tuning

B.3.1 Dataset Parameters

CIFAR Parameters: In all experiments we train for 24 epochs, with 1% of clients participating each round, for 2400 total iterations. We use the standard train/test split of 50000/10000. We split the dataset into 10000 clients, each of which has 5 points from a single target class. In each round we have 100 clients participating, inducing a batch size of 500 (this is of course increased when an adversary participates). We use standard data augmentation techniques: random crops, random horizontal flips, and the images are normalized according to the mean and standard deviation during training and testing. We do not use batch normalization in any of our experiments, because batch normalization does not work well on batches of 5 (batch normalization has to be conducted at a per-client level). We use a triangular learning rate schedule which peaks at 0.2. We use a momentum constant of 0.9. These training procedures and the ResNet9 architecture are drawn from Page [2].

FEMNIST Parameters: The FEMNIST dataset is composed of 805,263 28×28 pixel grayscale images which are distributed unevenly across 3,550 classes/users. Per user, there are an average of 226.83 datapoints, with a standard deviation of 88.94. To preprocess the data, we use the script in the LEAF repository with the command: `./preprocess.sh -s niid -sf 1.0 -k 0 -t sample`. After discarding some datapoints, we end with a dataset of 706,057 training samples and 80,182 validation samples across 3,500 clients ala Leaf [3].

The model architecture we use is a 40M-parameter ResNet101, but we replace the batch norm with layer norm because batch norm does not work well with small batch sizes. The average batch size is ≈ 600 but it can vary based on the clients that are sampled. We again use the standard data augmentations of random cropping and

[2] https://myrtle.ai/learn/how-to-train-your-resnet/
[3] https://tinyurl.com/u2w3twe

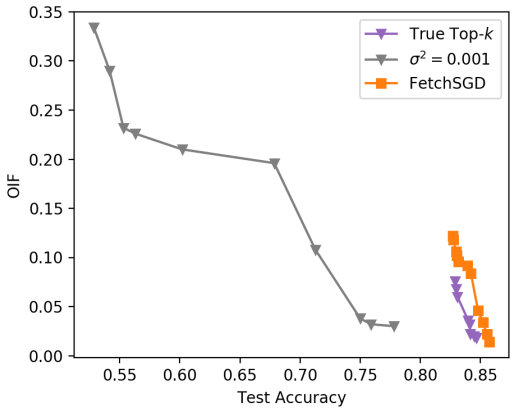


Figure 7: Pareto frontier of the ℓ_2 defense with noise addition and clip parameter 3, and `SparseFed` implemented with top_k and `FetchSGD` with clip parameter 3, against the combination of distributed poisoning and PGD. The attack was given the same grid search against all 3 defenses: $[50, 100, 200, 400] \times [5, 7, 9]$. Although noise addition is able to mitigate the attack, it suffers dramatically reduced test accuracy when compared to `SparseFed`; `SparseFed` achieves lower OIF with 10% higher test accuracy. CIFAR10, 10000 devices, 100 attackers.

Table 11: `FedAvg` convergence does not benefit from doing multiple local epochs. We use local learning rate=0.9, but even for a small number of local epochs convergence does not benefit, and at these small number of local epochs a smaller local learning rate would not have much impact because the exponential decay factor is not large. CIFAR10, 10000 devices, no attackers.

Num. epochs	**Test acc decrease**	**Test acc**
1	0	90
2	0.41	89.59
5	80	10

flips, and a triangular learning rate schedule. We train for only 1 epoch; this mimics the federated setup where we expect to only use each client once. We increase the learning rate from 0 to 0.01 over $\frac{1}{5}$th of the dataset, and then decrease the learning rate back to 0.

`FedAvg` **Parameters:** As we discuss in Appendix B.1, we use a standard implementation of `FedAvg` where there are three algorithmic hyperparameters: the number of local epochs, the local batch size, and the local learning rate decay. Prior work has already shown that the use of multiple local epochs does not improve convergence in the regime of small and non-i.i.d. datasets (Rothchild et al., 2020), and multiple algorithmic variants have been proposed to address this (Li et al., 2020) which are out of the scope of this work. Furthermore, the prior defenses considered in this work rely on approximating some consensus mechanism between benign devices based on the closeness or agreement of benign updates (Mhamdi et al., 2018). As the number of local epochs increases, this consensus falls apart, and so for the sake of fairness we do not evaluate defenses with more than one local epoch. In Table 11 we do our own experiments to validate that `FedAvg` convergence does not benefit from multiple local epochs.

B.3.2 Defense parameters

Norm clipping parameter: For the ℓ_2 defense, we tune the value of the clipping parameter. We test values for the ℓ_2 defense of $(1, 3, 5, 10)$. Where possible, we do a grid search over as many parameters as possible to find the limit of the attacker's ability.

First we validate the ℓ_2 defense against the baseline attack empirically in Table 12, which shows that by appropriately choosing the ℓ_2 parameter, the OIF is reduced significantly. There is a clear tradeoff: using stricter ℓ_2 norm clipping mitigates the attack further, but at the cost of reduced test accuracy.

Table 12: The appropriate choice of the norm clipping parameter greatly mitigates the effectiveness of the baseline attack on CIFAR with auxiliary set of size 500. CIFAR10, 10000 devices, 100 attackers.

Clipping param.	Test acc	Attack acc
10	0.7972	1
5	0.83	0.136
1	0.691	0.014

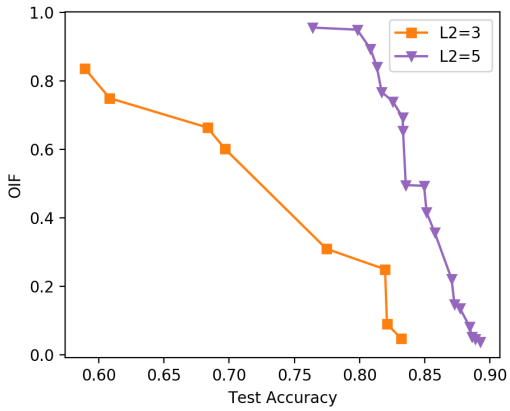


Figure 8: Pareto frontier of the ℓ_2 defense, comparing clipping parameters of 3 and 5. Although using a stricter norm clipping parameter can reduce OIF, it comes at the cost of test accuracy degradation. We find that when no attackers are present, using a norm clipping parameter of 5 does not sacrifice any test accuracy, whereas using a norm clipping parameter of 3 sacrifices $> 5\%$ test accuracy. Because we do not expect practitioners will adopt any defense which is guaranteed to reduce the performance of their models by such a nontrivial amount, we use a clipping parameter of 5. CIFAR10, 10000 devices, 100 attackers.

In Fig. 8 we examine the effect of using stricter clipping in the ℓ_2 defense. We show the pareto frontier of our attack against the ℓ_2 defense with two choices of the ℓ_2 parameter: 3 and 5. We find that when no attackers are present, using a parameter of 3 admits a minimum of 5% test accuracy degradation, while using a parameter of 5 does not reduce test accuracy at all in the same scenario. Therefore, while using a smaller norm clipping parameter can make the model more robust, it is also guaranteed to always reduce test accuracy. In keeping with the aforementioned systemic assumption that practitioners will not use defenses which damage model performance, we use the parameter of 5 in most experiments. For all further experiments, we show 5 as the parameter for the ℓ_2 defense, balancing test accuracy and adversarial robustness.

`SparseFed` parameters: For `SparseFed` we tune the value of k, the number of coordinates which are updated at each iteration. We test values of $[1, 5, 10, 50, 100, 200, 400] \times 10^3$ and report most experiments using the value of 5×10^3 on CIFAR10/CIFAR100/FMNIST, and use the value of 400×10^3. In the main body, we include graphs for the tradeoffs revolving around k.

In Fig. 9 we show the tradeoff between k, test accuracy, and attack accuracy for the uncompressed setting. In the main body, `FedAvg` is the baseline and as noted in (Bagdasaryan et al., 2020), the attacker can simply perform model replacement at the last iteration because the learning rate is nearly 0. However, in the uncompressed setting this is not possible, so we do not see the same trend as in the main body. In Appendix B.9 we showcase an algorithm which can realistically be implemented without using `FedAvg` to compress communication costs.

B.4 Impact of Defenses on Test Accuracy

Practitioners in federated learning prioritize the convergence of their models, and attempt to optimize tradeoffs of convergence with communication efficiency, security, and privacy. In Table 13 we show the decrease in test accuracy when no attackers are present for each defense evaluated in this work. We train each model for exactly

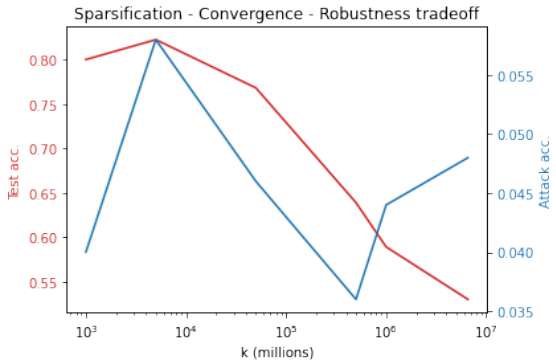


Figure 9: Tradeoff between sparsification parameter k (x axis, in logscale from 1000 to $k = d = 6568640$), test accuracy when attackers are present (left axis, blue), and attack accuracy (right axis, red) for *uncompressed* FL. In the uncompressed setting, no choice of k allows the attack to succeed, because as $k \to d$ no momentum is present and neither the attack nor the model converge. CIFAR10, 10000 devices, 200 attackers.

Table 13: Comparing the impact on test accuracy of the defenses. CIFAR10, 10000 devices, no attackers (averaged over 3 runs).

Defense	**Test Acc. decrease**	**Test Acc**
No defense	0 ±0	90.0 ±0.1
ℓ_2	2.0 ±0.1	88.0 ±0.1
Krum	80.0 ±0	10.0 ±0
Median	80.0 ±0	10.0 ±0
Trimmed mean ($f = 5$)	12.58 ±0.8	77.42 ±0.8
Bulyan ($f = 5$)	18.88 ±0.79	71.12 ±0.79
Bulyan ($f = 10$)	66.48	23.52
`SparseFed` ($k = 5e3$)	6.82 ±0.7	83.18 ±0.7
`SparseFed` ($k = 5e4$)	3.0 ±0.01	87.0 ±0.01

2400 iterations using the same triangular learning rate schedule. Because the Byzantine-resilient aggregation rules rely on outlier detection, they must necessarily throw away information even when attackers are not present. We set the robustness parameter $f = 5$ to give an idea of the tradeoff for these algorithms, because including a full curve is computationally infeasible. Bulyan drops more test accuracy than trimmed mean, because Bulyan throws away $4f + 2$ updates at each coordinate whereas trimmed mean only throws away $2f$ updates at each coordinate. As we explain in the main body of the work, Krum and coordinate median do not converge in this setting.

B.5 Stealth of Attack

Successful attacks are stealthy attacks: A necessary component of a successful attack is relative stealth. If an attacker can only successfully poison the model by overwriting all of the model's parameters that are necessary to achieve good performance on benign data, we do not consider this a viable attack. In any practical deployment, the entity coordinating federated learning would simply discard a model with such low accuracy after running the model on a private test set. We draw points for the auxiliary dataset from the test set. This can force the test accuracy to drop by as much as 5% when the attacker poisons the model with perfect accuracy over an auxiliary set of size 500 out of a test set of total size 10000. In Table 14 we include the decrease in test accuracy on the validation set **not including the auxiliary set** of size 500 , and confirm that the attack has an element of stealth. For the attacks on CIFAR10, CIFAR100, and FMNIST, the auxiliary dataset is drawn randomly from all classes and the decrease in test accuracy is also evenly distributed across the classes.

Throughout the Appendix we show the tradeoff between benign accuracy and attack accuracy/OIF in tables and pareto curves in graphs, and leave the task of evaluating risk to practitioners. We note that for the semantic backdoor task, the attack is not stealthy by definition.

Table 14: Attack accuracy and decrease in test accuracy on CIFAR10, 10000 devices, 200 attackers.

Name	Test acc decrease	Attack acc
Trimmed Mean	4.78	100
Bulyan	7.35	92.6
Clipping	7.1	100
SparseFed (Ours)	6.61	**25.6**

Table 15: Increasing the size of the auxiliary set can typically result in higher OIF with greatly reduced stealth when using the baseline attack against undefended model on CIFAR.

Aux. set size	Test acc	OIF
0	0.9001	0
500	0.7796	1
1000	0.6981	2
5000	0.3107	6.258

B.6 Strength of attack

In this section, we perform a thorough evaluation of the attack described in Algorithm 2 for both defended and undefended systems. We show that the attack is more powerful than previously considered, collusion can break existing defenses, poisoning attacks can also be used to induce Byzantine failures, evaluate model replacement attacks, and consider the possibility of an adaptive attack against `SparseFed`.

B.6.1 The outsized impact of model poisoning attacks on undefended systems

In this section, we evaluate the effectiveness of the baseline model poisoning attack against undefended federated learning systems. We find that the attack achieves a high OIF across a number of attack scenarios, much higher than considered by previous work.

In Tables 15 and 16, the baseline attack achieves higher OIF on CIFAR10 than any previous work has been able to attain. Here, 100 attackers corresponds to a frequency of 1, meaning 1 attacker is selected in every round, and 1 attacker corresponds to a frequency of 0.01, meaning 1 attacker is selected every 100 rounds. Given a total dataset size of 50000 and a client population of 10000, each attacker "should" only be to flip the labels of 5 datapoints, because that is the amount of data controlled by any agent in the system. Therefore, when 10 attackers are able to flip the labels of 500 datapoints with high accuracy, they achieve a remarkably high OIF.

This validates a hypothesis: *model poisoning attacks are orders of magnitude more powerful than has been shown.*

B.6.2 Colluding attackers break the norm clipping defense

We systematically evaluate the ℓ_2 norm clipping defense proposed in Sun et al. (Sun et al., 2019) against the strong attack with the ability for attackers to collude.

Table 16: The attack is effective against an undefended model on CIFAR, across a broad range of attacker population sizes. Prior work has not achieved high OIF values, ranging from 0.0063 to 0.126 (Bagdasaryan et al., 2020).

Aux. set size	No. attackers	Test acc	OIF
500	100	0.7796	1
500	10	0.8332	9.74
50	10	0.8842	1
50	1	0.887	1.1
5	1	0.8927	1

Table 17: The attack on the FEMNIST dataset far outperforms prior benchmarks which achieve at most an OIF of 0.03 (Sun et al., 2019).

Attacker batch size	Test acc	OIF
0	0.8198	0
300	0.7517	1.461
600	0.7618	1.456
1200	0.7614	1.405
3000	0.7969	0.0146

CIFAR10: In Fig. 10 we vary the attacker batch size and number of PGD epochs (more details in Appendix B.8) and obtain an attack which recovers an OIF close to 1. Against a defended system with a moderate stealth threshold of 5%, the attack can achieve an OIF of 0.5 which is significantly higher than any prior work claims (Sun et al., 2019). Therefore, *colluding attackers can break the ℓ_2 defense.*

FEMNIST: We now compare directly with (Sun et al., 2019) and examine the OIF they obtain in their paper. We keep the percentage of attackers the same and use the same attack and defense (PGD and ℓ_2 clipping), and observe that when we scale up the setting and the size of the auxiliary set, the defense does not scale. In Table 17, we achieve an OIF > 1. This corresponds to flipping the labels of nearly every datapoint from the considered task, which indicates that the peak OIF could be higher if considering different tasks. This OIF is about $50\times$ that of (Sun et al., 2019). Crucially, appropriate auxiliary set minibatching is required for success at this scale with low frequency. If the attacker batch size is too large, the adversary does not make enough progress on the iterations where it is present and the benign agents quickly revert the model on subsequent non-adversarial iterations.

Scaling up from (Sun et al., 2019): Our goal when doing experiments on FEMNIST is to evaluate a dataset where each device is only chosen to participate once. Furthermore, we want each iteration to include a somewhat realistic number of devices $(10 - 100)$ without exceeding the optimal batch size for our residual architecture $(500 - 600)$. Under these constraints, we split each device into $9 - 10$ devices so that we can sample $10s$ of devices at each iteration and maintain a good batch size. We are able to train a model to convergence in one pass over the dataset, sampling each device only once, and we believe that this is an important experimental setting for federated experiments.

Note on the attack: When attempting to modify the behavior of the benign model on a large number of datapoints, every additional point of OIF requires giving up more stealth, because every "misflipped" point reduces stealth but doesn't increase OIF. Further, increasing the number of PGD epochs, or the attacker batch size, moves along the OIF-stealth tradeoff. This suggests the following strategy for an attacker with an OIF goal in mind: while the goal is not met, increase the batch size as much as possible while maintaining convergence, then only increase the number of PGD epochs.

B.6.3 Byzantine attacks

Prior work has evaluated model poisoning attacks with the objective of inducing Byzantine failure against the same cadre of Byzantine-resilient aggregation rules (Fang et al., 2020). Using a better architecture, larger number of devices, smaller number of attackers, more severe non-i.i.d. partitioning, and smaller participation rate, we compare the Byzantine failure rate induced by our attack to previous work in Table 2. We modify the attack to return arbitrary gradients projected onto the perimeter of the ℓ_2 norm ball. We find that the scale of our evaluation reveals a much higher rate of Byzantine failure.

B.6.4 Model replacement attack against `SparseFed`

We replicate the experimental setting of the model replacement attack of (Bagdasaryan et al., 2020). In particular, we are attempting to insert a semantic backdoor on the Reddit dataset. The LSTM model architecture, dataset details, and all other experimental parameters are identical to those in the experiments of (Bagdasaryan et al., 2020). The semantic backdoor that we insert is again drawn from their experiments: "people in athens are rude". When we compare this to the backdoors inserted for computer vision datasets, it is easy to see that this backdoor makes up a relatively small portion of the training dataset in comparison. Therefore, the attack itself is much

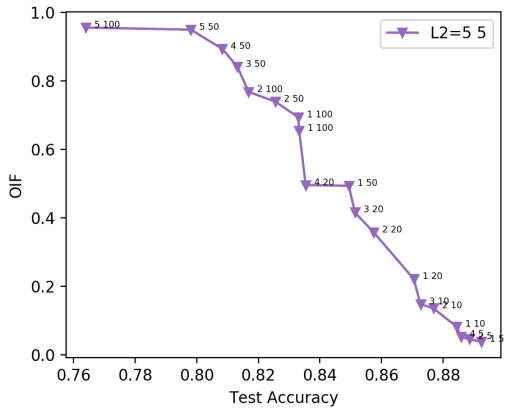


Figure 10: Pareto frontier of the attack against the ℓ_2 defense on CIFAR, for fixed auxiliary dataset size of 500, norm clipping parameter of 5, 10000 clients and 100 workers with 100 attackers. This attack achieves an OIF 5× higher than the baseline attack against the ℓ_2 defense.

stronger, and this has been observed by (Wang et al., 2020a) as it is an "edge-case" attack rather than the model poisoning attack we focus on in the main body which is very much a "base case" attack. In Figure 11 we evaluate the model replacement attack against the baseline ℓ_2 clipping defense with a threshold of 3, the minimum value that we find does not degrade test accuracy, and `SparseFed` with the same ℓ_2 clipping parameter. In the model replacement attack, the attacker compromises a small percentage of devices for a short period of time, and the goal is to enable the backdoor to persist for as long as possible while the benign devices continue training the model. We defer a detailed study of the staying power of the model replacement attack to future work, and only compare the speed that the baseline defense and `SparseFed` erase the backdoor. We do not tune the hyperparameter k for the new LSTM architecture and use the same value of k as in our computer vision experiments, that does not degrade convergence. The attack is inserted slightly faster when `SparseFed` is implemented, which is expected because we have not optimized the convergence-robustness tradeoff. Even unoptimized, `SparseFed` reduces the attack accuracy of the backdoored model significantly faster than the baseline defense.

B.6.5 Adaptive attack against `SparseFed`

We recognize that in order to produce attacks which can overcome strong defenses such as ℓ_2 norm clipping and Bulyan, we need to make use of adaptive attacks which incorporate knowledge of the defense into their attack strategy. Specifically, to beat ℓ_2 clipping the attacker should use PGD, and to beat Bulyan (or other Byzantine-resilient aggregation rules such as trimmed mean) the attackers should collude. For an adaptive attack against `SparseFed`, the attacker would need information about the top-k that will be updated, which is unrealistic in practice. Nevertheless, here we introduce an adaptive attacker with this information available to them to test the effectiveness of `SparseFed`.

Algorithm 14 Adaptive attack against `SparseFed`

Input: learning rate η, local batch size ℓ, norm clipping parameter L, number of local epochs e
Input: true top-k coordinates to be updated at this iteration K
1: This procedure is used by all attackers in a round to ensure that they upload the same update
2: **for** number of PGD epochs $e_i \in e$ **do**
3: Compute stochastic gradient g_i^t on batch B_i of size ℓ: $\mathrm{g}_i^t = \frac{1}{\ell}\sum_{j=1}^{l} \nabla_{\mathrm{M}} \mathcal{L}(\mathrm{M}_{e_i}^t, \mathrm{D}_j)$
4: Update local model $\widehat{M}_{e_{i+1}}^t = \mathrm{M}_{e_i}^t - \eta \mathrm{g}_i^t$
5: Project accumulated update onto the true top-k coordinates
6: Project accumulated update onto the perimeter of the ℓ_2 constraint $\mathrm{M}_{e_{i+1}}^t = M_0^t - CLIP(\widehat{M}_{e_{i+1}}^t - M_0^t)$
7: **end for**
Output: M_e^t

The main idea is to use PGD under the coordinate-wise constraint, with the assumption that the attacker has perfect knowledge of all gradients at the current timestep and is able to project their update onto the top-k

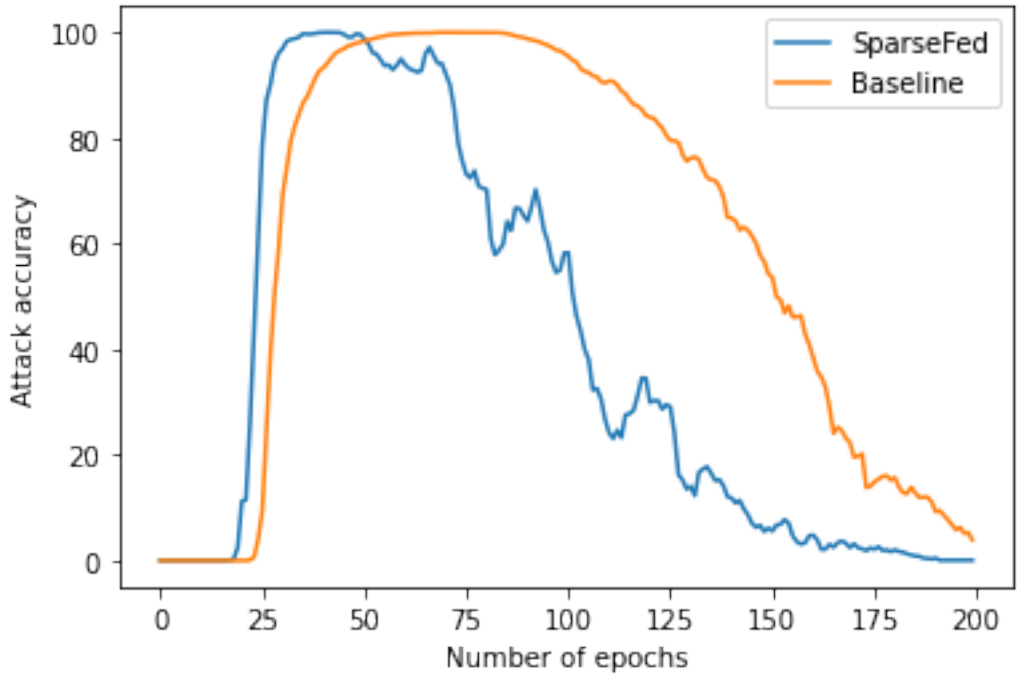


Figure 11: Model replacement attack on the Reddit dataset. `SparseFed` quickly returns the model to the benign optimum.

Table 18: The adaptive attack against `SparseFed` performs similarly to the base attack.

Attack	**Attack acc**	**Test acc**
Baseline	3.8	76.57
Adaptive	4.4	77.02

coordinates which will be updated. This attack will only succeed if the attacker has any signal in the true top-k coordinates; otherwise, the attacker will simply keep updating their local model with noise and no progress will be made. In Table 18 we evaluate the adaptive attack against `SparseFed` and find that the adaptive attack only obtains a negligible improvement over the baseline attack, indicating the strength of the `SparseFed` defense. The combination of clipping and small number of possible coordinates to update represent a fundamental barrier for the attacker.

B.7 Range proofs for `SparseFed`

We do not go in depth on a proposed implementation of range proofs in a federated learning system for three main reasons.

First, prior work on defenses do not make any claims about the computational or communication efficiency of their proposed robust aggregation mechanisms, including the methods that we compare to in this work (Bulyan, Krum, etc.) This includes the works which initially proposed L2 norm clipping as a defense (Sun et. al. 2019). Given this, we did not feel that there is a precedent for defense papers which utilize L2 norm clipping and its variants to propose an efficient range proof that is compatible with existing systems, as this would fall more in the realm of an applied-cryptography/systems-security paper.

Second, through our industry experience, we know that not all existing deployments make use of secure aggregation due to its costly overhead and inefficiency at scaling up to larger numbers of clients. Because this is the case, a federated learning system which does not use secure aggregation can implement L2 norm clipping at the server very efficiently.

Third, to the best of our knowledge, all existing defenses against model poisoning attacks all need some degree of verification of client's gradient updates whether it is L2 norm clipping or checking the sign of the gradient. SparseFed, unlike schemes which require consensus such as Bulyan or sign aggregation, does not require any additional secure computation beyond L2 norm clipping because there is no need to establish consensus between clients. In this regard, it is the most suited for deployment in a setting which requires secure aggregation assuming that a secure multiparty computation for L2 norm clipping has already been deployed.

Despite the above qualifications, we will now address the issue of how to implement range proofs for L2 norm clipping efficiently, using an informal description of how such a range proof can be achieved. While we do not provide details here, we believe that the method presented can lead to an actual proof in future work. The parties in a federated learning system are one server and one or more clients. The server will play the role of the verifier and the clients will be provers. Because our proposed protocol does not require any coordination between clients, without loss of generality we can simplify the system to one prover and one verifier. In the first step of the protocol, the prover generates a commitment to their update vector over the floating point domain. Next, the prover computes the sum of squares via a zkSNARK circuit (zero knowledge succinct non interactive argument of knowledge). Assuming that a custom SNARK is constructed for this application and the prover is using a standard multi-CPU chip found in the latest smartphones, the proving time would be less than thirty seconds (citation 1). This is minimal compared to the existing overhead in secure aggregation, which can take many minutes when accounting for multiple rounds of dropped users. If we want to be very conservative about how much information is leaked, we can treat the sum of squares as a secret committed value and use a bulletproof to ensure that it falls within the range of (0, L**2) where L is the L2 norm clipping constraint. Bulletproofs are fairly small and scale logarithmically in the number of commitments; we can validate all 100 L2 norms in one bulletproof for just 1MB in space, and all of this can be verified in 2ms by the verifier's hardware. If we can accept leaking the sum of squares, then we can just make it public and have the verifier check it outside the circuit. In either case, only provers who pass the verification will have their update vectors aggregated. This protocol sketch can be implemented without significantly increasing either the communication complexity (which is already quite large given that we have to at minimum upload gradients of deep networks) or the computation complexity (again, quite large because the device already has to compute gradients on local data).

B.8 Tuning Attack Parameters

CIFAR attack parameters: We consider various numbers of attackers: $[100, 200, 400, 1000]$ but most experiments are conducted with $100 - 200$ attackers which corresponds to having $1 - 2$ attackers present in every round. We consider this to be in line with a real world threat model. Typical federated learning training cycles take place over the course of a few days, and in order to use data from as many agents as possible, each round must draw data from many agents. Agents are called on to participate when they fulfill a number of criteria, and an attacker can forge these criteria in order to control when they are selected. Therefore, it should be straightforward for a small number of attackers to ensure that they are selected in every round. All auxiliary datapoints are drawn from the CIFAR validation set. Each point is randomly given a label from one of the 9 classes which it does not belong to. There are a number of unique attack hyperparameters which we search over. For the boosting factor, we search over [1,4,6,8,10,20] and find that a boosting factor of 20 works well for our experiments to ensure that PGD projects the update onto the perimeter of the ℓ_2 constraint. However, tuning the boosting factor does not make an impact whenever the ℓ_2 defense is in place with a sufficiently small clipping threshold. We tune the attacker's local batch size when they are doing PGD. We use values of $[N/10, 2N/10, 4N/10, 8N/10]$ where N is the size of the auxiliary set. We tune the number of epochs when the attacker is using the PGD attack. We use values of $[1, 3, 5, 7, 9, 11]$.

B.8.1 Hyperparameter Tuning in Attacks

The hyperparameters we consider are the attacker's **local batch size**, and the **number of local epochs for PGD**.

In Fig. 12 we consider the impact of changing the attacker batch size across two different auxiliary set sizes: 500 and 5000, against the ℓ_2 defense with parameter 5. We find that varying the attacker batch size for the smaller auxiliary set size reveals a smooth pareto frontier which enables the attacker to double its attack efficacy against the ℓ_2 for a moderate stealth budget when compared to the baseline attack. Increasing the attacker batch size up to a certain point increases the efficacy of the attack at the expense of stealth; further increasing the attacker

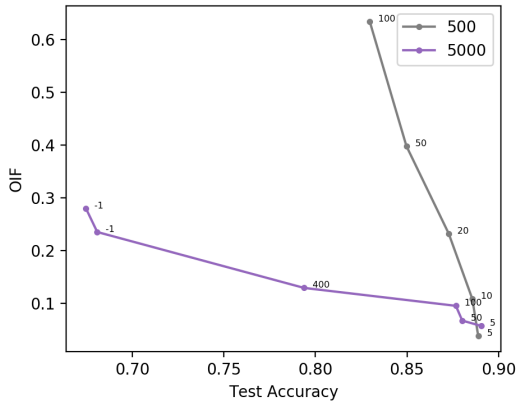


Figure 12: Pareto frontier of the attack when varying the batch size against the ℓ_2 defense with a parameter of 5, using auxiliary set sizes of 500 and 5000. While tuning the batch size does not achieve an OIF of 1, it does improve the pareto frontier for the attacker. We find that varying the attacker batch size moves along the OIF-stealth tradeoff; larger backdoors correspond to better OIF, at the expense of stealth.

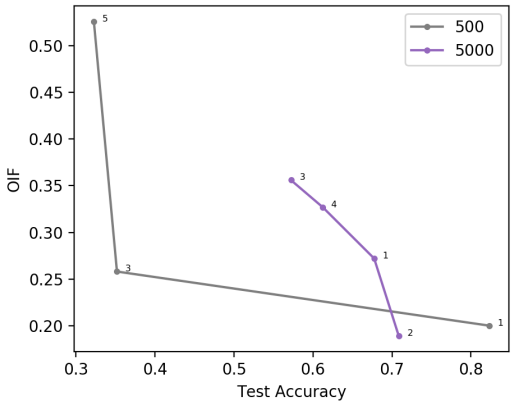


Figure 13: Pareto frontier of the PGD attack against the ℓ_2 defense with a parameter of 5, using auxiliary set sizes of 500 and 5000. Increasing the number of epochs improves the OIF at the expense of stealth.

batch size does not continue moving along the pareto frontier. This is because, as shown in our initial validation of the ℓ_2 defense, attempting to backdoor the entire auxiliary set at every iteration for the smaller auxiliary set results in a very small OIF.

In Fig. 13 we tune the number of PGD epochs against the ℓ_2 defense with parameter 5 at two different auxiliary set sizes, 500 and 5000. Performing a larger number of gradient descent iterations over the auxiliary set overfits the gradient significantly, which enables the attacker to insert a backdoor with higher OIF at the expense of a considerable degree of stealth.

B.8.2 Additional Results

In Fig. 14 we vary the size of the auxiliary set to observe how successful a more "ambitious" attacker can be. Generally, increasing the auxiliary set size enables the baseline attack to achieve a higher OIF at the expense of considerable stealth. These results are summarized in the main body in Table 1.

In Fig. 15, we use the attack to insert a large number of backdoors against an undefended system on the FEMNIST dataset. As mentioned in the main body, the OIF we obtain is notably $\approx 50\times$ that of the attack benchmarked in prior work. This is because we consider attackers that use a subset of the auxiliary set by minibatching, which

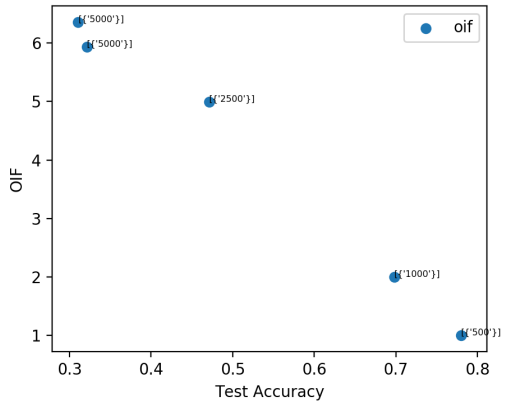


Figure 14: Pareto frontier of the baseline attack against the undefended system on CIFAR10 with 10000 clients and 100 workers. Annotation is the size of the auxiliary set.

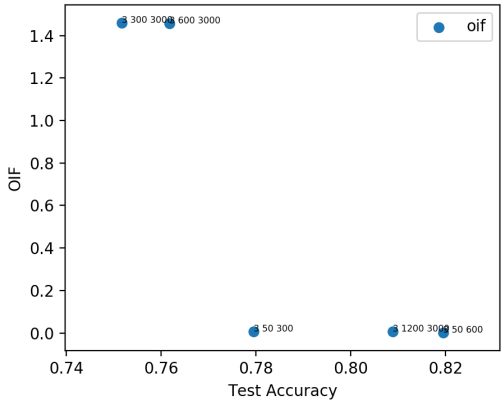


Figure 15: Pareto frontier of the attack against the undefended system on FEMNIST. Annotation is the attacker batch size, and the size of the auxiliary set. Using a larger auxiliary set with an appropriately tuned batch size allows for much higher OIF.

enables us to use a much larger overall auxiliary set size in the attack. These results are summarized in Table 17.

In Fig. 16 we show the baseline attack against a system on CIFAR100 with 50000 clients, each client possessing 1 datapoint, 500 workers and 100 attackers. When the system is undefended, the small number of attackers are able to insert an attack with OIF 1. However, enforcing the ℓ_2 defense with parameter 5 successfully mitigates this attack. In the main body, we show results for the adaptive attack, where the attack reaches 100% accuracy against the ℓ_2 defense.

In Figure 17 we vary the number of attackers against various defenses. We conclude that the defense which has the absolute highest robustness is: uncompressed `SparseFed` with $k = d$, which is equivalent to uncompressed ℓ_2 clipping without momentum. However, the test accuracy of this approach is low (44%). Overall, `SparseFed` dominates the other defenses significantly, especially for a smaller number of attackers.

In Table 19 we vary the nature of the semantic backdoor when attacking FEMNIST. Instead of targeting the pair of digits 1 and 7, we target 4 and 9. We find that both semantic backdoors perform similarly.

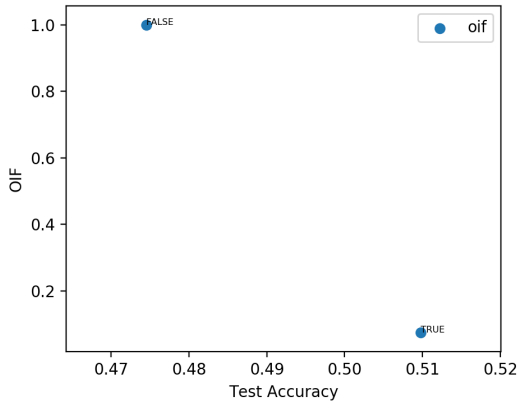


Figure 16: Baseline attack against CIFAR100 systems, with and without a DP-based ℓ_2 defense in place.

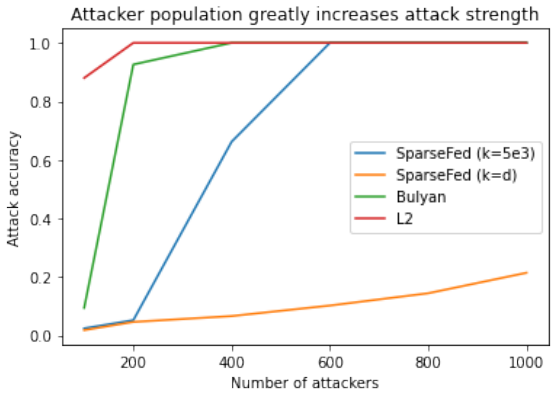


Figure 17: Attack against various defenses on CIFAR10 with varying number of attackers.

B.9 `FetchSGD`

B.9.1 The Case for Sparsification

In Fig. 18 we evaluate our provable defense using two implementations of `SparseFed`: top-k and FetchSGD sparsification. As an implementation detail, here we use top-k and the ℓ_2 defense in the uncompressed setting, and FetchSGD is in the "uncompressed" setting where the overall communication cost is reduced by a factor of 10. In all experiments, we update only $k = 5e4$ gradient parameters at every iteration. We see that for a defended system with a moderate stealth threshold of 5%, the attack achieves 0.05 OIF. Thus our `SparseFed` defense outperforms the ℓ_2 defense by a factor of 10× (recall that the ℓ_2 defense incurs an OIF of 0.5 under comparable constraints in Figure 10). Both implementations mitigate the attack, and using FetchSGD for robustness simultaneously achieves communication efficiency and enables us to operate in the uncompressed setting where we gain further robustness.

Table 19: Varying the semantic backdoor does not have a significant impact on the success of the attack against FEMNIST.

Defense	Attack acc (1/7)	Attack acc (4/9)
ℓ_2	100	100
`SparseFed`	1.95	6.72

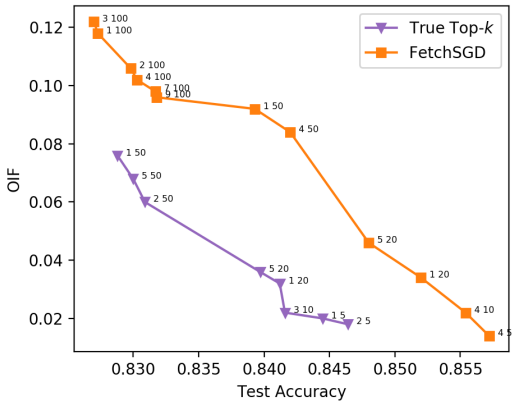


Figure 18: Pareto frontier of `SparseFed` using top_k and `FetchSGD` with ℓ_2 clipping using parameter 5, against varying hyperparameters of the colluding PGD attack, with a fixed auxiliary dataset of size 500. This is the best that the strongest available attack can perform against our defense, and we achieve a factor of $5-10\times$ improvement over the ℓ_2 defense.

C Limitations and societal impact

Limitations: The empirical limitation of our work is that we are forced to make imperfect simulations of cross-device federated settings because we do not have access to real federated datasets at the scale of tens of thousands of devices. For CIFAR10, CIFAR100, and FMNIST, lacking any natural non-iid partitioning, our simulation strategy is to simulate each device only drawing samples from the distribution of one class than multiple classes, but this may not necessarily be true in the real world. We encourage the federated learning community to contribute real-world and large-scale datasets to overcome such limitations in the future.

Security considerations: We recognize that our analysis of existing Byzantine resilient defenses reveals that colluding attackers can successfully attack systems which may use these defenses today. To mitigate these attacks, we urge stakeholders in these deployed systems to inspect their vulnerabilities using the same powerful attacker we use in our work. The field of federated learning has seen a great deal of research interest lately. Federated learning systems today utilize data from millions of users and serve millions more, so adversarial robustness is of paramount importance. Prior work in the field of targeted model poisoning attacks has examined the impact that attacks have in the cross-silo setting, and mostly concluded that In this work, we complement this body of work by demonstrating the outsized impact of model poisoning attacks on systems at scale and showing that existing defenses can be broken by colluding attackers. We also introduce `SparseFed`, and prove practical robustness guarantees for our novel defense. We compare `SparseFed` to existing defenses, and confirm that it outperforms these against our strongest available attacks empirically at large scales. Although future work may introduce attacks which are stronger than we consider, we emphasize that `SparseFed` will maintain provable robustness against any attack. We leave investigation of the tradeoffs between other proposed attacks and defenses to future work.