Robust Aggregation for Federated Learning

Krishna Pillutla Sham M. Kakade Zaid Harchaoui
University of Washington

Abstract

We present a robust aggregation approach to make federated learning robust to settings when a fraction of the devices may be sending corrupted updates to the server. The proposed approach relies on a robust secure aggregation oracle based on the geometric median, which returns a robust aggregate using a constant number of calls to a regular non-robust secure average oracle. The robust aggregation oracle is privacy-preserving, similar to the secure average oracle it builds upon. We provide experimental results of the proposed approach with linear models and deep networks for two tasks in computer vision and natural language processing. The robust aggregation approach is agnostic to the level of corruption; it outperforms the classical aggregation approach in terms of robustness when the level of corruption is high, while being competitive in regime of low corruption.

1 Introduction

The explosion of rich, decentralized user data from the growing use of smartphones, wearables and edge devices has led to a massive resurgence of interest in the classical field of distributed optimization [10]. While this data has the potential to power the next generation of machine learning and data analytics, it also comes with a responsibility to safeguard the privacy of sensitive user data.

Federated learning (FL) [48] has emerged as a leading paradigm in this setting where a large number of devices with privacy-sensitive data collaboratively optimize a machine learning model under the orchestration of a central server, while keeping the data fully decentralized and private. The privacy of the data is further bolstered by *secure aggregation*, which requires that the server cannot inspect individual contributions of devices but can only access an aggregate. Its applications range from mobile apps [4, 64] to healthcare [28, 55].

In existing approaches, a *secure average oracle* [12] is used to aggregate individual device updates as an element-wise mean, which is susceptible to corrupted updates [47]. Robustness to corrupted updates is a desirable property in general for distributed optimization and in particular for federated optimization. Corrupted updates could be caused, among other reasons, by a malfunction in low-cost hardware on mobile devices, corrupted data or adversaries. However, we set aside here the problem of inferring the causes of corrupted updates. The design of an aggregation oracle which is more robust than the regular one is further complicated by privacy constraints, which require that individual updates cannot be inspected. Therefore, we investigate secure robust aggregation oracles which can be easily implemented using calls to a regular (i.e., non-robust) secure average oracle, relying upon its built-in privacy guarantees.

The approach we present is more attractive than designing from scratch a new secure aggregation oracle which is robust because (a) the latter, as a nonlinear aggregate, would incur a much higher communication cost since secure multiparty computation primitives, upon which secure aggregation is built, are most efficient in general for linear functions such as the weighted mean [e.g., 23], and, (b) a practical implementation of a new secure aggregation oracle at scale would require an immense engineering effort [e.g., 13] while the proposed approach can leverage existing infrastructure with little engineering overhead. Thus, the proposed approach provides robustness to static and adaptive data corruption, as well as update corruption where the devices participate faithfully in the aggregation loop.

Contributions. The main take-away message of this work is:

Federated learning can be made robust to corrupted updates by replacing the weighted arithmetic mean aggregation with an approximate geometric median at thrice the communication cost.

To this end, we make the following concrete contributions.

(a) *Robust Aggregation:* We design a novel robust aggregation oracle based on the classical geometric median. We analyze the convergence of the resulting FL algorithm for i.i.d. least-squares estimation and show that the proposed method is robust to update corruption in up to half the devices.

(b) *Algorithmic Implementation:* We show how to implement this robust aggregation oracle in a practical and privacy-preserving manner using a small number of calls to a secure average oracle. This relies on an alternating minimization algorithm which empirically exhibits rapid convergence. This algorithm can be interpreted as a numerically stable version of the classical algorithm of Weiszfeld [63], thus shedding new light on it.

(c) *Numerical Simulations and Open Source Implementation:* We demonstrate the effectiveness of our framework for data corruption and parameter update corruption, on FL tasks from computer vision and natural language processing, with linear models as well as convolutional and recurrent neural networks. In particular, our results show that the proposed robust aggregation oracle leads to a FL algorithm which, (a) outperforms FedAvg [48] in settings of high corruption level and (b) matches the performance of the federated averaging algorithm in *thrice the communication cost* in low corruption level. Moreover, the aggregation algorithm is completely agnostic to the actual level of corruption in the problem instance.

The Python codes and scripts used to generate experimental results are available online [1]. Furthermore, the proposed approach is available in TensorFlow Federated [2].

Related Work. We now survey some related work.

Federated Learning. FL was introduced in [48] as a distributed optimization approach to handle on-device machine learning, with a secure average oracle given in [12]. Extensions were proposed in [35, 51, 56, 59] — see [33] for a survey. We address robustness, which is broadly applicable in these settings.

Distributed Optimization. Distributed optimization has a long history [see e.g., 10]. Recent work includes a primal-dual framework called COCOA [44, 60] and its decentralized version [25], as well as asynchronous incremental algorithms [38] and distributed optimization in networks [14, 52, 57].

Robust Estimation. Robust estimation was pioneered by Huber [29, 30]. Robust median-of-means approaches were introduced in [53], with more recent work in [27, 39, 42, 43, 49]. Robust mean estimation, in particular, received much attention [18, 21, 50]. These works consider the statistics of robust estimation in the i.i.d. case, while we focus on distributed optimization with privacy-preservation.

Geometric Median Algorithms. The classical algorithm of Weiszfeld [63] has received much attention [7, 34, 36, 62]. However, all these variants are *not* numerically stable, while our variant is (cf. Remark 8). A landmark theoretical construction led to a nearly-linear time algorithm for the geometric median [20], but its practical applicability is unclear.

Privacy. Differential privacy [22] is a popular framework to guarantee privacy of user data. It is an orthogonal direction to ours, and could be used in conjunction. An alternate approach is homomorphic encryption [24] - see [12, 33] for a discussion on privacy and FL.

Byzantine Robustness. Byzantine robustness, resilience to arbitrary, even adversarial behavior of some devices [37], has been studied in in gradient based updates [3, 11, 16, 17, 65]. In this work, we consider a more nuanced and less adversarial corruption model because cryptographic protocols which make up secure aggregation require faithful participation of the devices and thus, Byzantine robustness is a priori not possible without additional assumptions. In addition, our setting requires faithful participation of devices in the aggregation loop — see Sec. 2 for examples of its practical relevance. Further, it is unclear how to securely

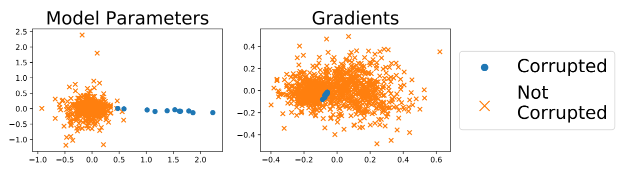


Figure 1: Aggregating model parameters versus gradients: PCA projection on $\mathbb{R}^2$ of model parameters and gradients for a linear model on the EMNIST dataset under static corruption of 1% of the data — see Sec. B.4 of the supplement for details.

Algorithm 1 FL Meta-Algorithm

```
Input: $F$ from (1), devices per round $m$
1: for $t = 1, 2, \cdots$ do                    ▷ Run on the server
2:     $S_t \leftarrow$ random set of $m$ devices
3:     Broadcast $w^{(t)}$ to each $k \in S_t$
4:     for each device $k \in S_t$ in parallel do
5:         $w_k^{(t)} \leftarrow \mathit{LocalUpdate}(k, w^{(t)})$
6:     $w^{(t+1)} \leftarrow \mathit{SecureAggregate}(\{w_k^{(t)}\}_{k \in S_t})$
```

implement the nonlinear aggregation algorithms of these works. Lastly, we note that the use of, e.g., secure enclaves [61] in conjunction with the approach proposed here could guarantee Byzantine robustness in FL.

Moreover, we aggregate *model parameters* in a robust manner, which is irreconcilably different from the approach of aggregating *gradients* in all of these related works; see Fig. 1 for an illustration. We take this approach because (a) in the FL setting, aggregating parameters allows us to make more progress at the same communication cost by increasing local computation on each device, and, (b) assumptions on the distributions of parameters can be easier to interpret than assumptions on distributions of gradients.

Overview. Sec. 2 describes the problem formulation, Sec. 3 proposes a robust aggregation oracle and presents a convergence analysis of the resulting robust FL algorithm, while Sec. 4 gives a concrete implementation of the robust aggregation oracle using only a small number of calls to the secure average oracle. Finally, in Sec. 5, we present comprehensive numerical simulations comparing the proposed federated learning algorithm to FedAvg [48] and distributed stochastic gradient descent (SGD).

2 Problem Setup and Corruption Model

Consider the optimization problem

$$\min_{w \in \mathbb{R}^d} \left[F(w) := \sum_{k=1}^{K} \alpha_k \, \mathbb{E}_{\xi \sim \mathcal{D}_k} \left[f(w; \xi) \right] \right] , \tag{1}$$

where $f : \mathbb{R}^d \times \Xi \to \mathbb{R}$ is given and for each $k \in [K]$, $\alpha_k > 0$ is a weight and and $\mathcal{D}_k$ is a probability distribution supported on Ξ. We assume that the each expectation above is well-defined and finite, and that $f(\cdot, \xi)$ is continuously differentiable for each $\xi \in \Xi$.

In federated learning, each $k \in [K]$ represents a node or a compute device, e.g., a mobile phone. A server or fusion center, which does not contain any data, orchestrates the optimization by communicating directly with each of the devices. The goal of FL is to solve the optimization problem (1), i.e., find the optimal parameters $w^\star$ in as little communication between the devices and the server as possible, while the cost of local computation on each device is negligible in comparison.

Of special interest is supervised machine learning. Here, $\xi = (x, y)$ is an input-output pair and $f(w; \xi) = \ell(y, \varphi(x; w))$, where ℓ is a loss function such as the square loss, and φ maps the input x to a prediction using model parameters w. For instance, $\varphi(x, w) = \phi(x)^\top w$ is a linear predictor with a fixed basis $\phi(x) \in \mathbb{R}^d$.

Privacy Constraints. Owing to the privacy-sensitive nature of the data $\xi \sim \mathcal{D}_k$, each device k is not allowed to directly share any data $\xi \sim \mathcal{D}_k$ with the server. It can perform some local computation based on the data and only share model parameters $w_k \in \mathbb{R}^d$. Furthermore, the server can only access an aggregate of parameters sent by the devices, while individual w_k is private. We now make this precise.

Definition 1. Given m devices with each device k containing $w_k \in \mathbb{R}^d$ and $\beta_k > 0$, a *secure average oracle* computes the average $\sum_{k=1}^m \beta_k w_k / \sum_{k=1}^m \beta_k$ at a total communication of $\mathcal{O}(md + m^2)$ bits such that no w_k or β_k are revealed to either the server or any other device. Any function $A : (\mathbb{R}^d)^m \to \mathbb{R}^d$ which can be computed by finitely many calls to a secure average oracle is said to be an *iterative secure aggregate*.

An example secure average oracle is the cryptographic protocol of Bonawitz et al. [12].

FL Meta-Algorithm. We now make precise the general FL meta-algorithm as a distributed algorithm, which runs in synchronized rounds of communication between the server and the devices, consisting in two steps: (a) each device receives the server's model parameters, performs some local computation and sends its update, and, (b) the server updates its parameters based on an iterative secure aggregate of the parameter updates sent by the devices. These are denoted respectively as *LocalUpdate* and *SecureAggregate* in Algo. 1.

(a) *LocalUpdate*: Formally, we write the local computation Φ_k on device k as $\Phi_k(w, H) := \Phi(w, H; \mathcal{D}_k) \in \mathbb{R}^d$, where $w \in \mathbb{R}^d$ is the parameter sent by the server, H represents other information sent by the server such as the learning rate. The procedure Φ is only allowed to draw i.i.d. samples from $\mathcal{D}_k$. In FedAvg [48], Φ is N steps of SGD based on samples $\xi_1, \cdots, \xi_N \sim \mathcal{D}_k$ for a given N.

(b) *SecureAggregate*: In each step t, the server weighs the parameters sent by the selected device $k \in S_t$ with α_k to compute an iterative secure aggregate. Note that FedAvg [48] aggregates using one call to a secure average oracle as $w^{(t+1)} = \sum_{k \in S_t} \alpha_k \Phi_k(w^{(t)}, H_k) / \sum_{k \in S_t} \alpha_k$.

Model of Corrupted Updates. We allow a set $\mathcal{C} \subseteq [K]$ of "corrupted" devices to, unbeknownst to the server, send corrupted parameter updates. Formally, we modify the local computation Φ_k to allow for corrupted updates as

$$\Phi_k(w, H) = \begin{cases} \Phi(w, H; \mathcal{D}_k)\,, & k \notin \mathcal{C} \\ \Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) & k \in \mathcal{C}\,, \end{cases} \tag{2}$$

where Ψ is an arbitrary $\mathbb{R}^d$-valued function which is allowed to depend on the distributions $\mathcal{D} = (\mathcal{D}_1, \cdots, \mathcal{D}_K)$ of all devices, as well as the server state $\mathcal{S}$. We define the corruption level ρ as $\rho := \sum_{k \in \mathcal{C}} \alpha_k / \sum_{k \in [K]} \alpha_k$.

Next, we consider some examples of update corruption — see [33] for a comprehensive treatment. Corrupted updates could be non-adversarial in nature. Potential causes include sensor malfunctions or hardware bugs in unreliable and heterogeneous devices (e.g., mobile phones) which are outside the control of the orchestrating server.

On the other hand, corrupted updates could be adversarial in nature, where the corrupted devices $\mathcal{C}$ work together to alter the behavior of the system. These could be of several types:

(a) *Static data poisoning:* The corrupted devices $\mathcal{C}$ are allowed to modify their training data prior to the start of the FL algorithm, and the data is fixed thereafter. All devices faithfully execute the *LocalUpdate* procedure in this setting. Formally, we can write for $k \in \mathcal{C}$ that $\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = \Phi(w, H; \mathcal{D}'_k)$, where the distribution $\mathcal{D}'_k$ has been modified from $\mathcal{D}_k$.

(b) *Adaptive data poisoning:* The corrupted devices $\mathcal{C}$ are allowed to modify their training data in each round of the FL algorithm depending on the current model w. Again, all devices faithfully execute the *LocalUpdate* procedure. Formally, we have, $\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = \Phi(w, H; \mathcal{D}''_k(w))$, where the distribution $\mathcal{D}''_k(w)$ depends on w.

(c) *Update Poisoning:* The corrupted devices can run an arbitrary procedure in place of *LocalUpdate*, as described by (2) in its full generality. This setting subsumes all previous examples as special cases.

Table 1: Examples corruptions and capability of an adversary they require, as measured along the following axes: **Data write**, where a device $k \in \mathcal{C}$ can replace its local dataset by any arbitrary data; **Model read**, where a device $k \in \mathcal{C}$ can read the server model w and can execute arbitrary code based on w prior to the execution of *LocalUpdate*, while *LocalUpdate* itself is executed faithfully; **Model write**, where a device $k \in \mathcal{C}$ can execute arbitrary code in place of *LocalUpdate*; and, **Aggregation**, where a device $k \in \mathcal{C}$ can execute arbitrary code in place of *SecureAggregate*.

Corruption Type	**Data write**	**Model read**	**Model write**	**Aggregation**	**RFA applicable?**
Non-adversarial	-	-	-	-	✓
Static data poisoning	Yes	-	-	-	✓
Adaptive data poisoning	Yes	Yes	-	-	✓
Update poisoning	Yes	Yes	Yes	-	✓
Byzantine	Yes	Yes	Yes	Yes	N/A

Note that the corruption model in (2) precludes the *Byzantine setting*, which refers to the worst-case model where a corrupted client device $k \in \mathcal{C}$ can behave arbitrarily during both the *LocalUpdate* procedure, as well as the *SecureAggregate* procedure. In this setting, it provably impossible to design a Byzantine-robust iterative secure aggregate (in the sense of Def. 1). The examples listed above highlight the importance of robustness to the corruption model under consideration.

We compare the various corruptions in terms of the capability of an adversary required to induce a desired corrupted update in Table 1.

3 Robust Aggregation Oracle and the RFA Algorithm

In this section, we design a robust aggregation oracle and analyze the convergence of the resulting federated learning algorithm.

Robust Aggregation with the Geometric Median. The geometric median (GM) of $w_1, \cdots, w_m \in \mathbb{R}^d$ with weights $\alpha_1, \cdots, \alpha_m > 0$ is the minimizer of

$$g(z) := \sum_{k=1}^{m} \alpha_k \|z - w_k\| , \tag{3}$$

where $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm. As a robust aggregation oracle, we use an ϵ-approximate minimizer $\widehat{z}$ of g which satisfies $g(\widehat{z}) - \min_z g(z) \le \epsilon$. We denoted it by $\widehat{z} = \mathrm{GM}\left((w_k)_{k=1}^m, (\alpha_k)_{k=1}^m, \epsilon\right)$. When $\epsilon = 0$, we omit it in the expression above. Further, we denote the exact GM when $\alpha_k = 1/m$ as $\mathrm{GM}\left((w_k)_{k=1}^m\right)$.

The GM is known to be robust to an arbitrary corruption of points with at most half the total weight [41]. We defer to Sec. 4 the actual implementation of robust aggregation oracle using only a secure average oracle. We assume that $w_1, \cdots w_k$ are non-collinear, which is reasonable in the FL setting. Then, g admits a unique minimizer $z^\star$. Further, we suppose that $\sum_k \alpha_k = 1$ w.l.o.g.[1]

Robust Federated Aggregation: The RFA Algorithm. We now present RFA in Algo. 2 as an instantiation of the FL meta-algorithm (Algo. 1) using this GM-based robust aggregation oracle. In particular, the local computation *LocalUpdate* is same as in the case of FedAvg, i.e., a few steps of SGD, while the aggregate is an approximate GM. Note that RFA is completely agnostic to the actual level of corruption in the problem. The robust aggregation step enjoys the following robustness guarantee.

[1]One could apply the results to $\widetilde{g}(z) := g(z)/\sum_{k=1}^m \alpha_k$ and translate them back to g.

Proposition 2. *Consider the setting of Algo. 2 with tolerance $\epsilon \geq 0$. Suppose F from Eq. (1) is L-smooth*[2]*. For any $w^\star \in \arg\min_w F(w)$ and any $S' \subseteq S$ such that $\theta := \sum_{k \in S'} \alpha_k < 1/2$, the output $\widehat{z}$ of the SecureAggregate procedure from Algo. 2 satisfies,*

$$F(\widehat{z}) - F(w^\star) \leq \frac{L}{(1-2\theta)^2} \left(4 \max_{k \in S \setminus S'} \|w_k - w^\star\|^2 + \epsilon^2 \right) .$$

Proof. The classical robustness property of an ϵ-approximate GM, $\widehat{z}$ [41, Thm. 2.2], says that

$$\|\widehat{z} - z\| \leq 2 \left(\frac{1-\theta}{1-2\theta} \right) \max_{k \notin S} \|w_k - z\| + \frac{\epsilon}{1-2\theta} ,$$

for any $z \in \mathbb{R}^d$. The proof is completed by taking $z = w^\star$, invoking smoothness of F as $F(w^{(t+1)}) - F(w^\star) \leq (L/2)\|w^{(t+1)} - w^\star\|^2$ [e.g., 54, Thm. 2.1.5] and lastly, using $(a+b)^2 \leq 2(a^2 + b^2)$ for reals a, b. □

By letting S' denote the set of corrupted devices, we see that an approximate GM is insensitive to corrupted updates. Note that Prop. 2 does not require any convexity assumptions on F.

Convergence of RFA. We now present a convergence analysis of RFA. For this purpose, we make two general assumptions. First, we suppose that the data are homogeneously distributed across devices, i.e., $\mathcal{D}_k = \mathcal{D}_{k'}$ for any pair of devices $k, k' \in [K]$. Consequently, each device is weighted by $\alpha_k = 1/K$.

Second, we focus on least-squares fitting of additive models. Let y be a random variable with $\mathbb{E}[y] = 0$ and $\mathbb{E}[y^2] < \infty$ and x be a $\mathcal{X}$-valued random variable with distribution $\mathbb{P}_X$ for some measurable space $\mathcal{X}$. The goal is to estimate the regression function $\overline{x} \mapsto \mathbb{E}[y | x = \overline{x}]$ from a training sequence of independent copies of (x, y) in each device. Denoting by $\mathbb{P}$ the joint distribution of (x, y) in each device, we define,

$$F(w) = \frac{1}{K} \sum_{k=1}^{K} F_k(w) \ , \quad \text{where} \quad F_k(w) = \mathbb{E}_{(x,y) \sim \mathbb{P}} \ell(y, w^\top \phi(x)) \text{ for all } k \in [K] \, , \tag{4}$$

where $\ell(y, f) = (1/2)(y - f)^2$ is the square loss and $\phi(x) = (\phi_1(x), \ldots, \phi_d(x))$ where $\phi_1, \ldots, \phi_d : \mathcal{X} \to \mathbb{R}$ are a fixed basis of measurable, centered functions. The basis functions may be nonlinear, thus encompassing random feature approximations of kernel feature maps and pre-trained deep network feature representations. We consider stochastic gradient algorithms equipped with "tail-averaging" [32], i.e., averaging over the latter half of the trajectory of iterates.

We state our results under the following assumptions: (a) the feature maps are bounded as $\|\phi(x)\| \leq R$ with probability one under $\mathbb{P}_X$; (b) F is μ-strongly convex and L-smooth; (c) the additive model is well-specified with noise variance σ^2. The results could be extended to broader settings under weaker assumptions. The setting of least-squares regression, and the above assumptions in particular allow us to leverage sharp convergence rates [5, 31, 32], and focus the analysis on the effect of aggregation as opposed to technicalities in the analyses of stochastic gradient algorithms. These assumptions are made in all statements in Sec. 3.

We now show with high probability (with respect to the sampling of the devices by the server and the sampling of data $(x, y) \sim \mathbb{P}$ within devices) that RFA converges when the corruption level $\rho < 1/2$ for a large enough m. Note in this setting that ρ is the fraction of devices sending corrupted updates in this setting.

Theorem 3. *Consider F defined in (4) and the corruption level satisfies $\rho < 1/2$. Consider Algo. 2 run for T outer iterations where (a)* LocalUpdate *is tail-averaged SGD with learning rate $\gamma = 1/(2R^2)$ run for N_t steps in outer iteration t, and (b)* SecureAggregate *returns the* exact *geometric median (i.e., $\epsilon = 0$). Fix a*

[2] A function f is L-smooth if it is continuously differentiable and its gradient ∇f is Lipschitz w.r.t. $\|\cdot\|$.

Algorithm 2 The RFA algorithm

```
Input: Tolerance $\epsilon$
1: Instantiate Algo. 1 with
2: function SecureAggregate($S, \{w_k\}_{k \in S}$)
3:     $\alpha_k \leftarrow n_k / \sum_{j \in S} n_j$ for $k \in S$
4:     $\widehat{z} \leftarrow \mathrm{GM}\left((w_k)_{k\in S}, (\alpha_k)_{k \in S}, \epsilon\right)$ using Algo. 3
5:     return $\widehat{z}$
```

Algorithm 3 Smoothed Weiszfeld's Algorithm

```
Input: $w_1, \cdots, w_m \in \mathbb{R}^d$ with $w_k$ on device $k$, $\alpha_1, \cdots, \alpha_m > 0$, $\nu > 0$, budget $T$, $z^{(0)} \in \mathbb{R}^d$, secure average oracle $\mathcal{A}$
1: for $t = 0, 1, \cdots, T-1$ do
2:     Server broadcasts $z^{(t)}$ to devices $1, \cdots, m$
3:     Device $k$ computes $\beta_k^{(t)} \leftarrow \frac{\alpha_k}{\nu \vee \|z^{(t)} - w_k\|}$
4:     $z^{(t+1)} \leftarrow \frac{\sum_{k=1}^m \beta_k^{(t)} w_k}{\sum_{k=1}^m \beta_k^{(t)}}$ using $\mathcal{A}$
   return $z^{(T)}$
```

$\delta > 0$ and let the number of outer iterations, T, be known in advance. Let $0 < q < 1/2$ and the number of devices per iteration, m, be such that

$$\theta := q + \rho(1-q) + (2-q)\sqrt{\frac{1}{2m}\log\left(3T/\delta\right)} < 1/2\,. \tag{5}$$

Define $C_\theta := (1-\theta)/\sqrt{1-2\theta}$, $w^\star = \arg\min F$, $\Delta_0 := \|w^{(0)} - w^\star\|^2$ and $\kappa := R^2/\mu$. Let $N \geq 4\kappa \log\left(4C_\theta^2 \kappa / q\right)$. If we choose the number N_t of tail-averaged SGD steps in iteration t as $N_t = 2^t N$, then Algo. 2 converges with probability at least $1-\delta$ as

$$F(w^{(T)}) - F(w^\star) \leq \frac{L\Delta_0}{2^{T+1}} + \frac{8C_\theta^2}{q}\frac{\kappa d \sigma^2}{N}\frac{T}{2^T}\,.$$

If, on the other hand, we choose $N_t = N$, then, with probability at least $1-\delta$,

$$F(w^{(T)}) - F(w^\star) \leq \frac{L\Delta_0}{2^{T+1}} + \frac{8C_\theta^2}{q}\frac{\kappa d \sigma^2}{N}\,.$$

Remark 4. *(a) For practical settings where the corruption level ρ is bounded away from $1/2$, then C_θ and q are constants. For instance, $C_\theta = 2$ for $\theta \approx 0.464$ and $C_\theta = 10$ for $\theta \approx 0.499$. Furthermore, for a corruption level $\rho = 1/4$, and $q = 10^{-2}$, $T = 3.3 \times 10^6$, $\delta = 10^{-2}$, $\theta = 0.49$, we require $m \approx 700$.*

(b) The contribution of the noise term $\kappa d\sigma^2$ stays constant for $N_t = N$, while convergence requires that we increase the number of tail-averaged SGD steps in each iteration, e.g., as $N_t = 2^t N$. This is feasible since the cost of local computation is relatively cheap when compared to the communication.

(c) In the case of no corrupted updates, the noise term $\kappa d \sigma^2$ can be improved to $d\sigma^2$ for a bound in expectation (instead of high probability) by noting that $\mathrm{GM}(w_1, \cdots, w_m) \in \mathrm{conv}\{w_1, \cdots, w_m\}$. This also holds for an approximate GM which lies in $\mathrm{conv}\{w_1, \cdots, w_m\}$.

(d) The final bounds in Theorem 3 do not depend on m. We can obtain bounds which to improve with m by using the "median-of-means" approach which partitions the selected devices S into disjoint groups $S_1, \cdots, S_r$ and returns $\mathrm{GM}\big((\sum_{k \in S_j} w_k / |S_j|)_{j \in [r]}\big)$. However, this comes at the cost of requiring more devices m per iteration. See, e.g., [49] for details.

To prove the theorem, we use two results: the convergence of tail-averaged SGD [31, Thm. 1], [32, Cor. 2] and the "confidence-boosting" property of the GM in the presence of outliers [49, Thm. 3.1].

Theorem 5 ([31, 32])**.** *Consider F defined in Eq. (4). Then, defining $\kappa := R^2/\mu$, the output $\overline{z}_N$ of tail-averaged SGD, with initial iterate $z_0 \in \mathbb{R}^d$, learning rate $\gamma = 1/(2R^2)$ and number of steps N satisfies*

$$\mathbb{E}\|\overline{z}_N - w^\star\|^2 \leq 2\kappa \exp\left(-\frac{N}{4\kappa}\right)\|z_0 - w^\star\|^2 + \frac{8d\sigma^2}{\mu N}\,.$$

Lemma 6 ([49]). *Let $z_1, \cdots, z_m$ be independent estimators of $z^\star \in \mathbb{R}^d$ and let $\theta < 1/2$ and $\epsilon > 0$ be fixed. Suppose $\mathbb{P}(\|z_k - z^\star\| > \epsilon) \le q$ holds for $k \in S \subseteq [m]$ and that $\theta > \tau + q - \tau q$ for $\tau = 1 - |S|/m$. Then, with $C_\theta := (1-\theta)/\sqrt{1-2\theta}$, the geometric median $\widehat{z} = \mathrm{GM}((z_k)_{k=1}^m)$ satisfies*

$$\mathbb{P}(\|\widehat{z} - z^\star\| > C_\theta \epsilon) \le \exp\left(-2m(1-\tau)\left(\frac{\theta - \tau}{1-\tau} - q\right)^2\right).$$

Proof of Thm. 3. The proof proceeds as follows. For one iteration, we set up low probability bounds for *LocalUpdate* using Thm. 5, and boost them to high probability for the GM using Lemma 6. We track the failure probability over T iterations using the union bound.

Notation. For each t, let $\mathcal{F}_t$ denote the sigma algebra generated by $w^{(t)}$. Let S_t denote the set of selected devices of outer iteration t, and define $\tau_t := |S_t \cap \mathcal{C}|/|S_t|$ as the fraction of corrupted devices selected in iteration t. Define $\tau := \rho + \sqrt{(1/2m)\log(3T/\delta)}$. and define $\mathcal{E} := \{\tau_t \le \tau \text{ for } t = 0, \cdots, T-1\}$ to be the event that at most a τ-fraction of corrupted devices have been chosen in any round. Note that $\tau < \theta < 1/2$.

Further, define $\mathcal{E}_t$ to denote the event that the following holds:

$$\|w^{(t)} - w^\star\|^2 \le \frac{1}{2}\|w^{(t-1)} - w^\star\|^2 + \frac{8C_\theta^2 d\sigma^2}{q\mu N_t}. \tag{6}$$

Computing the Rate. Suppose events $\mathcal{E}, \mathcal{E}_1, \ldots, \mathcal{E}_T$ simultaneously hold. Unrolling (6) from $t = T, \cdots, 1$,

$$\|w^{(T)} - w^\star\|^2 \le 2^{-T}\|w^{(0)} - w^\star\|^2 + \frac{8C_\theta^2 d\sigma^2}{q\mu}\sum_{t=1}^{T}\frac{1}{2^{T-t}N_{t-1}}.$$

To obtain the bound, we sum the series in the second term and invoke smoothness as $F(w^{(T)}) - F(w^\star) \le (L/2)\|w^{(T)} - w^\star\|^2$. When $N_t = 2^t N$, the series sums to $2^{-(T-1)}T/N$, while for $N_t = N$, the series is upper bounded by $2/N$. It remains to compute the probability that events $\mathcal{E}, \mathcal{E}_1, \ldots, \mathcal{E}_T$ simultaneously hold.

Setting up Probability Bound. Consider *LocalUpdate* on a not-corrupted device $k \in S_t \setminus \mathcal{C}$, starting from $w^{(t)}$. Thm. 5 gives, upon using $N_t \ge N \ge 4\kappa\log(4C_\theta^2\kappa/q)$,

$$\mathbb{E}\left[\|w_k^{(t)} - w^\star\|^2 \,\middle|\, \mathcal{E}, \mathcal{F}_t\right] \le \frac{q}{2C_\theta^2}\|w^{(t)} - w^\star\|^2 + \frac{8d\sigma^2}{\mu N_t}.$$

From Markov's inequality,

$$\mathbb{P}\left(\|w_k^{(t)} - w^\star\|^2 > \frac{1}{2C_\theta^2}\|w^{(t)} - w^\star\|^2 + \frac{8d\sigma^2}{q\mu N_t} \,\middle|\, \mathcal{E}, \mathcal{F}_t\right) \le q.$$

Noting that $\theta > \tau + q - \tau q$, we invoke Lemma 6 to get

$$\mathbb{P}(\overline{\mathcal{E}}_{t+1} | \mathcal{E}, \mathcal{F}_t) \le \exp\left(-2m(1-\tau)\left(\frac{\theta - \tau}{1-\tau} - q\right)^2\right). \tag{7}$$

Bounding Failure Probability. Here, we bound the probability of failure $\mathbb{P}(\overline{\mathcal{E}\mathcal{E}_{:T}})$, where we use shorthand $\mathcal{E}_{:T} = \mathcal{E}_1 \cdots \mathcal{E}_T$. From (7) and the union bound, we get,

$$\mathbb{P}\left(\overline{\mathcal{E}_{:T}} \,|\, \mathcal{E}\right) \le T \exp\left(-\frac{2m}{1-\tau}(\theta - \tau - q + q\tau)^2\right) \le \delta/3\,,$$

where we plugged in the values of θ, τ and used $1 - \tau \le 1$. Next, Hoeffding's inequality gives $\mathbb{P}(\tau_t > \tau) \le \exp(-2m(\tau - \rho)^2) = \delta/(3T)$, from our choice of τ. A union bound over $t = 0, \cdots, T-1$ now gives $\mathbb{P}(\overline{\mathcal{E}}) \le \delta/3$. Finally, from the union bound and the law of total probability, we get,

$$\begin{aligned}\mathbb{P}\left(\overline{\mathcal{E}\mathcal{E}_{:T}}\right) &\le \mathbb{P}\left(\overline{\mathcal{E}}\right) + \mathbb{P}\left(\overline{\mathcal{E}}_{:T}\right) \\ &= \mathbb{P}\left(\overline{\mathcal{E}}\right) + \mathbb{P}\left(\overline{\mathcal{E}_{:T}} \,\middle|\, \overline{\mathcal{E}}\right)\mathbb{P}\left(\overline{\mathcal{E}}\right) + \mathbb{P}\left(\overline{\mathcal{E}_{:T}} \,|\, \mathcal{E}\right)\mathbb{P}\left(\mathcal{E}\right) \\ &\le 2\mathbb{P}\left(\overline{\mathcal{E}}\right) + \mathbb{P}\left(\overline{\mathcal{E}_{:T}} \,|\, \mathcal{E}\right) \le \delta\,.\end{aligned}$$

□

4 Implementing a Robust Aggregation Oracle

While the GM is a natural robust aggregation oracle, the key challenge in the federated setting is to design an algorithm to compute it with a small number of calls to a secure average oracle only. We now consider an alternating minimization approach which satisfies this criteria and analyze its convergence.

Smoothing and Surrogate. Since the GM objective g is non-smooth, we consider, for $\nu > 0$, the smoothing

$$g_\nu(z) := \sum_{k=1}^m \alpha_k \|z - w_k\|_{(\nu)}\,, \quad \text{where} \quad \|z\|_{(\nu)} := \begin{cases} \frac{1}{2\nu}\|z\|^2 + \frac{\nu}{2}\,, & \|z\| \le \nu \\ \|z\|\,, & \|z\| > \nu \end{cases}. \tag{8}$$

It is known that g_ν is a $(1/\nu)$-smooth approximation to g and that g_ν approximates g to $\nu/2$ [8]. We minimize g_ν with a surrogate $G : \mathbb{R}^d \times \mathbb{R}^m_{++} \to \mathbb{R}$ defined using $\eta = (\eta_1, \cdots, \eta_m) \in \mathbb{R}^m$ as

$$G(z, \eta) := \frac{1}{2}\sum_{k=1}^m \alpha_k \left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k\right).$$

It is easy to see that G is jointly convex in (z, η) and that the following holds:

$$\min_{(z,\eta)\in\mathbb{R}^d\times\mathcal{E}_\nu} G(z,\eta) = \min_{z\in\mathbb{R}^d} g_\nu(z)\,, \quad \text{where} \quad \mathcal{E}_\nu := \{\eta \in \mathbb{R}^m \,:\, \eta_1, \cdots, \eta_m \ge \nu\}\,.$$

Algorithm and Convergence. We consider an alternating minimization algorithm in G over z, η:

$$\eta^{(t)} = \arg\min_{\eta\in\mathcal{E}_\nu} G(z^{(t)}, \eta)\,, \quad \text{and,} \quad z^{(t+1)} = \arg\min_{z\in\mathbb{R}^d} G(z, \eta^{(t)})\,.$$

Both updates can be computed in closed form — see Algo. 3. Each iteration requires one call to the secure average oracle. We call it the smoothed Weiszfeld algorithm for its resemblance to Weiszfeld [63]'s classical algorithm, which is a special case of Algo. 3 with $\nu = 0$ when $z \ne w_k$ for all k. Recall from our corruption model that each device k, including $k \in \mathcal{C}$ correctly computes $\beta_k^{(t)}$ and w_k remains fixed throughout.

The algorithm enjoys the following rate of convergence.

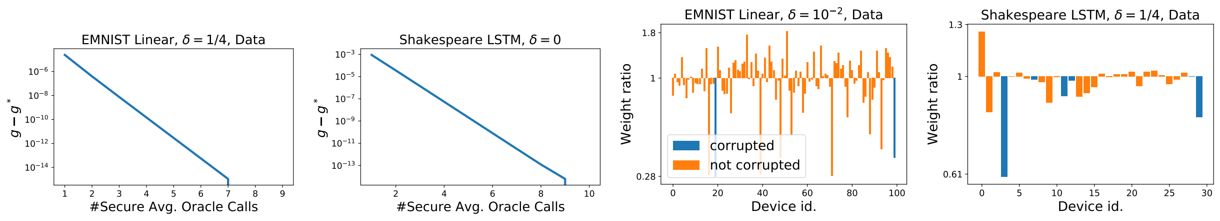


Figure 2: Left: Convergence of the smoothed Weiszfeld algorithm. Right: Visualization of the re-weighting β_k/α_k, where β_k is the weight of w_k in $\mathrm{GM}((w_k), (\alpha_k)) = \sum_k \beta_k w_k$. See Sec B.5 of the supplement for details.

Proposition 7. *The iterate $z^{(t)}$ of Algo. 3 with input $z^{(0)} \in \mathrm{conv}\{w_1, \cdots, w_m\}$ and $\nu > 0$ satisfies*

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\overline{\nu} t} + \frac{\nu}{2},$$

where $z^\star = \arg\min g$ and $\overline{\nu} = \min_{t' \in [t], k \in [m]} \nu \vee \|z^{(t'-1)} - w_k\| \ge \nu$. Furthermore, if $z^\star$ does not coincide with any w_k, and $\nu \le \min_{k=1,\cdots,m} \|z^\star - w_k\|$, then it holds that $g(z^{(t)}) - g(z^\star) \le 2\|z^{(0)} - z^\star\|^2/\overline{\nu} t$.

For a ϵ-approximate GM, we set $\nu = \mathcal{O}(\epsilon)$ to get a $\mathcal{O}(1/\epsilon^2)$ rate. However, if the GM $z^\star$ is not too close to any w_k, then the same algorithm automatically enjoys a faster $\mathcal{O}(1/\epsilon)$ rate.

Remark 8. (a) *Weiszfeld's original algorithm and its variants are numerically unstable due to division by $\|z - w_k\|$ when it is small. This is combated in practice by heuristically using $\nu \vee \|z - w_k\|$ for some small ν, which results exactly in Algo. 3. Here, we use the smooth version g_ν to rigorously and directly analyze the algorithm used in practice.*

(b) *While Prop. 7 proves a global sublinear rate, it is known that the Weiszfeld algorithm exhibits locally linear convergence [34]. Indeed, we find in Fig. 2 that Algo. 3 displays rapid convergence, giving a high quality solution in* 3-5 iterations.

(c) *This rapid empirical convergence obviates the need for acceleration, which is possible in theory [7, 45].*

The proof relies on the following key lemma, which shows descent and contraction. Full details of the proofs below are in Sec. A of the supplement.

Lemma 9. *The sequence $(z^{(t)})$ produced by the Algo. 3 satisfies*

(a) $g_\nu(z^{(t+1)}) \le g_\nu(z^{(t)}) - 1/(2L^{(t)})\|\nabla g_\nu(z^{(t)})\|^2$, *where* $L^{(t)} := \sum_{k=1}^m \alpha_k/\eta_k^{(t)}$,

(b) *if* $g_\nu(z^{(t+1)}) \ge g_\nu(z^\star)$, *then* $\|z^{(t+1)} - z^\star\| \le \|z^{(t)} - z^\star\|$.

Proof. A simple computation shows that $\nabla g_\nu(z^{(t)}) = L^{(t)}\left(z^{(t)} - z^{(t+1)}\right)$, and,

$$g_\nu(z) \le G(z, \eta^{(t)}) = g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z - z^{(t)}\|^2.$$

We arrive at part (a) by using $z = z^{(t+1)}$ and plugging in $z^{(t)} - z^{(t+1)} = (1/L^{(t)})\nabla g_\nu(z^{(t)})$. For part (b), we use $z = z^{(t+1)}$ and convexity as $g_\nu(z^\star) \ge g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z^\star - z^{(t)}\right)$ coupled with the expression above for $\nabla g_\nu(z^{(t)})$, we get,

$$\begin{aligned} g_\nu(z^{(t+1)}) &= g_\nu(z^\star) + L^{(t)}\left((z^{(t)} - z^{(t+1)})^\top (z^{(t+1)} - z^\star) + \frac{1}{2}\|z^{(t+1)} - z^{(t)}\|^2 \right) \\ &= g_\nu(z^\star) + \frac{L^{(t)}}{2}\left(\|z^{(t)} - z^\star\|^2 - \|z^{(t+1)} - z^\star\|^2 \right), \end{aligned}$$

where the last step used the cosine law from trigonometry. The lemma follows from noting that $L^{(t)} > 0$. □

Table 2: Dataset description and statistics.

Dataset	Task	#Classes	#Train	#Test	#Devices	#Train per Device		
						Median	Max	Min
EMNIST	Image Classification	62	$204K$	$23K$	1000	160	418	92
Shakespeare	Character-level Language Modeling	53	$2.2M$	$0.25M$	628	1170	70600	90

Proof of Prop. 7. With the previous lemma in place, the classical proof technique of gradient descent [e.g., 54, Theorem 2.1.13] gives $g(z^{(t)}) - g(z^\star) \le 2\|z^{(0)} - z^\star\|^2/\overline{\nu}t$ if $g_\nu(z^{(t)}) - g_\nu(z^\star) > 0$. Else, we have, $g_\nu(z^{(t)}) - g_\nu(z^\star) \le 0$, we get from the properties of the smoothing that $g(z^{(t)}) - g(z^\star) \le \nu/2$. $\square$

5 Numerical Simulations

We now conduct simulations to compare RFA with other federated learning algorithms. The simulations were run using TensorFlow and the data was preprocessed using LEAF [15]. The full details from this section and more simulation results are given in the Sec. B of the supplement.

Datasets, Tasks and Models. We consider two machine learning tasks. The datasets are described in Table 2. Since we only have a finite sample $\xi_1, \cdots, \xi_{n_k} \sim \mathcal{D}_k$ for device k, we take its weight $\alpha_k \propto n_k$.

(a) *Character Recognition*: We use the EMNIST dataset [19], where the input x is a 28×28 grayscale image of a handwritten character and the output y is its identification (0-9, a-z, A-Z). Here, each device is a writer of the handwritten character x. We use two models for φ — a linear model $\varphi(x; w) = w^\top x$ and a convolutional neural network (ConvNet). The loss function ℓ used is the multinomial logistic loss and performance is evaluated using the classification accuracy.

(b) *Character-Level Language Modeling*: We learn a character-level language model over the Complete Works of Shakespeare [58]. We formulate it as a multiclass classification problem, where the input x is a window of 20 characters, the output y is the next (i.e., 21st) character. Each device is a role from a play (e.g., Brutus from The Tragedy of Julius Caesar). We use a long-short term memory model (LSTM) [26] for φ, and the loss function ℓ is the multinomial logistic loss. The performance is evaluated with the classification accuracy of next-character prediction.

Corruption Models. We consider two corruption models in *LocalUpdate* of corrupted devices $k \in \mathcal{C}$.

(a) *Data Corruption*: This is an instance of static data poisoning, where the distribution $\mathcal{D}_k$ on a device $k \in \mathcal{C}$ is replaced by some $\mathcal{D}'_k$. For the EMNIST dataset, we take the negative of an image so that $\mathrm{d}\mathcal{D}'_k(x, y) = \mathrm{d}\mathcal{D}_k(1 - x, y)$. For the Shakespeare dataset, we reverse the text so that $\mathrm{d}\mathcal{D}'_k(c_1 \cdots c_{20}, c_{21}) = \mathrm{d}\mathcal{D}_k(c_{21} \cdots c_2, c_1)$. In both cases, the labels are unchanged.

(b) *Omniscient corruption*: This is an example of the update poisoning, where the parameters w_k returned by devices $k \in \mathcal{C}$ are modified so that the weighted arithmetic mean $\sum_{k \in S} p_k w_k$ over the selected devices S is set to the negative of what it would to have been without the corruption. This is designed to hurt the weighted arithmetic mean aggregation.

Hyperparameters. The hyperparameters of FedAvg and RFA are chosen similar to the defaults of [48]. A learning rate and its decay were tuned on a validation set for FedAvg without any noise and the same values were used for all settings. Further, the aggregation step of RFA is implemented using the smoothed Weiszfeld's algorithm with a budget of 3 calls to the secure average oracle, thanks to its rapid empirical

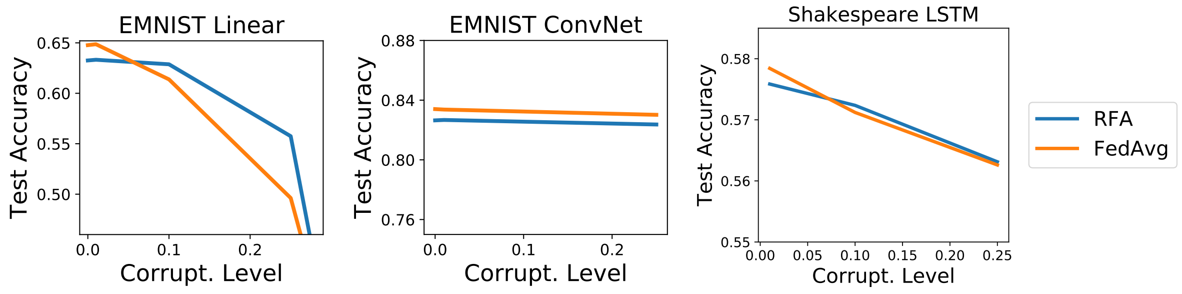


Figure 3: Comparison of robustness of RFA and FedAvg under data corruption.

convergence (cf. Fig. 2), and $\nu = 10^{-6}$ for numerical stability (cf. Remark 8). Each simulation was repeated 5 times and the shaded area denotes the minimum and maximum over these runs. Sec. B of the supplement contains detailed hyperparameter choices, an experimental study of the effect of varying hyperparameters such as the number of chosen devices and the smoothed Weiszfeld communication budget.

Robustness. Fig. 3 compares the maximum test accuracy of RFA and FedAvg. We observe that RFA gives us improved robustness in case of the linear model. On the ConvNet and LSTM models, both FedAvg and RFA perform similarly. We note that the behavior of the training of a neural network when the data is corrupted is not well-understood in general [e.g., 66].

Performance Across Iterations. Next, we plot in Fig. 4 the performance of competing methods versus the number of rounds of communication as measured by the number of calls to the secure average oracle.

We note that in the low corruption regime of $\rho = 0$ or $\rho = 10^{-2}$ under data corruption, FedAvg is faster when measured in the number of calls to the secure average oracle. However, it matches the performance of RFA when measured in terms of the number of outer iterations (cf. Fig. 9-11 in Sec. B of the supplement).

Further, both FedAvg and SGD diverge under the omniscient corruption. An exception, however, is that FedAvg with the LSTM model does not diverge under the omniscient corruption at $\rho = 10^{-2}$, where the probability of encountering an omniscient corruption in any iteration is $\sim 5\%$. Note that RFA still converges with no change in accuracy. Furthermore, we observe that RFA is qualitatively more stable than FedAvg in its behavior over time in the high data corruption setting of $\rho = 1/4$.

Acknowledgments The authors would like to thank Zachary Garrett, Peter Kairouz, Jakub Konečný, Brendan McMahan, Krzysztof Ostrowski and Keith Rush for fruitful discussions, as well as help with the implementation of RFA on Tensorflow Federated. This work was first presented at the Workshop on Federated Learning and Analytics in June 2019. This work was supported by NSF CCF-1740551, NSF DMS-1839371, the Washington Research Foundation for innovation in Data-intensive Discovery, the program "Learning in Machines and Brains", and faculty research awards.

This research was funded in part by JPMorgan Chase & Co. Any views or opinions expressed herein are solely those of the authors listed, and may differ from the views and opinions expressed by JPMorgan Chase & Co. or its affiliates. This material is not a product of the Research Department of J.P. Morgan Securities LLC. This material should not be construed as an individual recommendation for any particular client and is not intended as a recommendation of particular securities, financial instruments or strategies for a particular client. This material does not constitute a solicitation or offer in any jurisdiction.

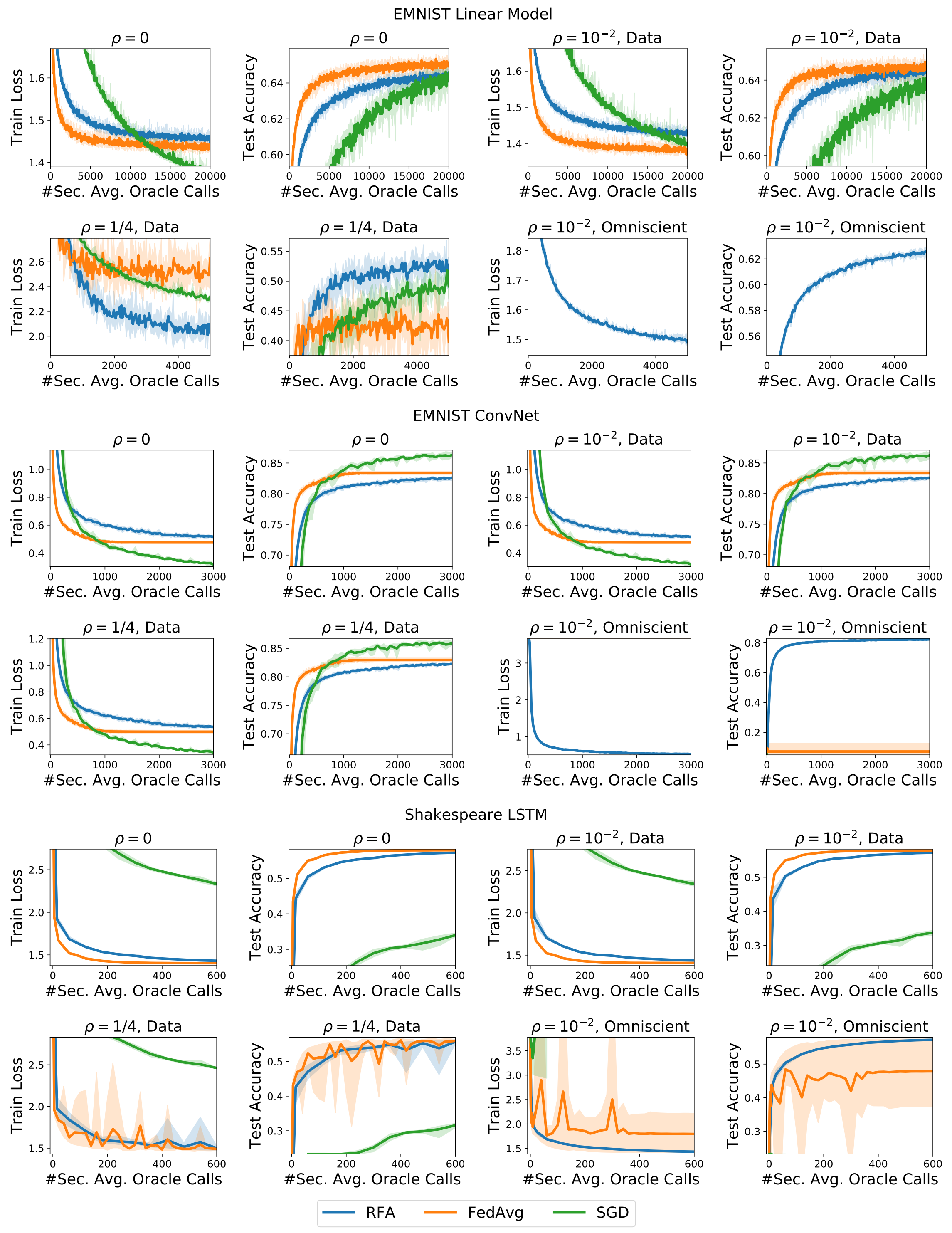


Figure 4: Comparison of methods plotted against number of calls to the secure average oracle for different corruption settings. For the case of omniscient corruption, FedAvg and SGD are not shown in the plot if they diverge.

References

[1] https://github.com/krishnap25/rfa, 2019.

[2] https://github.com/tensorflow/federated/tree/master/tensorflow_federated/python/research/robust_aggregation, 2019.

[3] D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 31*, pages 4618–4628, 2018.

[4] M. Ammad-ud din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System. *arXiv Preprint*, 2019.

[5] F. Bach and E. Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In *Advances in Neural Information Processing Systems*, pages 773–781, 2013.

[6] A. Beck. On the Convergence of Alternating Minimization for Convex Programming with Applications to Iteratively Reweighted Least Squares and Decomposition Schemes. *SIAM Journal on Optimization*, 25(1): 185–209, 2015.

[7] A. Beck and S. Sabach. Weiszfeld's Method: Old and New Results. *J. Optimization Theory and Applications*, 164 (1):1–40, 2015.

[8] A. Beck and M. Teboulle. Smoothing and First Order Methods: A Unified Framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[9] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[10] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[11] P. Blanchard, R. Guerraoui, E. M. El Mhamdi, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30*, pages 119–129, 2017.

[12] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[13] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, and H. B. McMahan. Towards Federated Learning at Scale: System Design. *arXiv Preprint*, 2019.

[14] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

[15] S. Caldas, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. LEAF: A benchmark for federated settings. *arXiv Preprint*, 2018.

[16] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos. DRACO: Byzantine-resilient Distributed Training via Redundant Gradients. In *International Conference on Machine Learning*, pages 902–911, 2018.

[17] Y. Chen, L. Su, and J. Xu. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

[18] Y. Cheng, I. Diakonikolas, and R. Ge. High-Dimensional Robust Mean Estimation in Nearly-Linear Time. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 2755–2771, 2019.

[19] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv Preprint*, 2017.

[20] M. B. Cohen, Y. T. Lee, G. L. Miller, J. Pachocki, and A. Sidford. Geometric Median in Nearly Linear Time. In *Symposium on Theory of Computing*, pages 9–21, 2016.

[21] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart. Robust Estimators in High Dimensions without the Computational Intractability. In *Symposium on Foundations of Computer Science*, pages 655–664, 2016.

[22] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284, 2006.

[23] D. Evans, V. Kolesnikov, M. Rosulek, et al. A Pragmatic Introduction to Secure Multi-Party Computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[24] C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.

[25] L. He, A. Bian, and M. Jaggi. COLA: Decentralized Linear Learning. In *Advances in Neural Information Processing Systems 31*, pages 4541–4551, 2018.

[26] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] D. J. Hsu and S. Sabato. Loss Minimization and Parameter Estimation with Heavy Tails. *Journal of Machine Learning Research*, 17:18:1–18:40, 2016.

[28] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu. Patient Clustering Improves Efficiency of Federated Machine Learning to Predict Mortality and Hospital stay time using Distributed Electronic Medical Records. *Journal of Biomedical Informatics*, 99:103291, 2019.

[29] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 03 1964.

[30] P. J. Huber. *Robust Statistics*. Springer, 2011.

[31] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, V. K. Pillutla, and A. Sidford. A Markov Chain Theory Approach to Characterizing the Minimax Optimality of Stochastic Gradient Descent (for Least Squares). In *Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 2:1–2:10, 2017.

[32] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18:223:1–223:42, 2017.

[33] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gasc/'on, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *arXiv Preprint*, 2019.

[34] I. N. Katz. Local convergence in Fermat's problem. *Mathematical Programming*, 6(1):89–104, 1974.

[35] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv Preprint*, 2016.

[36] H. W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4(1):98–107, Dec 1973.

[37] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[38] R. Leblond, F. Pedregosa, and S. Lacoste-Julien. Improved Asynchronous Parallel Optimization Analysis for Stochastic Incremental Methods. *Journal of Machine Learning Research*, 19, 2018.

[39] G. Lecué and M. Lerasle. Robust machine learning by median-of-means: theory and practice. *arXiv Preprint*, 2017.

[40] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[41] H. P. Lopuhaa and P. J. Rousseeuw. Breakdown points of affine equivariant estimators of multivariate location and covariance matrices. *Annals of Statistics*, 19(1):229–248, 03 1991.

[42] G. Lugosi and S. Mendelson. Risk minimization by median-of-means tournaments. *arXiv Preprint*, 2016.

[43] G. Lugosi and S. Mendelson. Regularization, sparse recovery, and median-of-means tournaments. *arXiv Preprint*, 2017.

[44] C. Ma, J. Konečný, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takác. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

[45] J. Mairal. Optimization with First-Order Surrogate Functions. In *International Conference on Machine Learning*, pages 783–791, 2013.

[46] J. Mairal. Incremental Majorization-Minimization Optimization with Application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[47] R. Maronna, D. Martin, and V. Yohai. *Robust Statistics: Theory and Methods*. Wiley, 2006.

[48] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[49] S. Minsker. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

[50] S. Minsker. Uniform Bounds for Robust Mean Estimators. *arXiv Preprint*, 2018.

[51] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic Federated Learning. *arXiv Preprint*, 2019.

[52] A. Nedic and A. Ozdaglar. Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[53] A. S. Nemirovski and D. B. Yudin. Problem Complexity and Method Efficiency in Optimization. 1983.

[54] Y. Nesterov. *Introductory Lectures on Convex Optimization Vol. I: Basic course*, volume 87. Springer Science & Business Media, 2013.

[55] A. Pantelopoulos and N. G. Bourbakis. A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, 2009.

[56] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the Convergence of Federated Optimization in Heterogeneous Networks. *arXiv Preprint*, 2018.

[57] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié. Optimal Convergence Rates for Convex Distributed Optimization in Networks. *Journal of Machine Learning Research*, 20(159):1–31, 2019.

[58] W. Shakespeare. The Complete Works of William Shakespeare. URL `https://www.gutenberg.org/ebooks/100.`

[59] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems 30*, pages 4424–4434, 2017.

[60] V. Smith, S. Forte, M. Chenxin, M. Takáč, M. I. Jordan, and M. Jaggi. COCOA: A General Framework for Communication-Efficient Distributed Optimization. *Journal of Machine Learning Research*, 18:230, 2018.

[61] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia. A Formal Foundation for Secure Remote Execution of Enclaves. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450, 2017.

[62] Y. Vardi and C.-H. Zhang. A modified Weiszfeld algorithm for the Fermat-Weber location problem. *Mathematical Programming*, 90(3):559–566, 2001.

[63] E. Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[64] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. Applied Federated Learning: Improving Google Keyboard Query Suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[65] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5636–5645, 2018.

[66] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Supplementary Material:
Robust Aggregation for Federated Learning

Table of Contents

A The Smoothed Weiszfeld Algorithm

In this section, we elaborate on the smoothed Weiszfeld's algorithm.

A.1 Setup

We are given distinct points $w_1, \cdots, w_m \in \mathbb{R}^d$ and scalars $\alpha_1, \cdots, \alpha_m > 0$ such that $\sum_{k=1}^m \alpha_k = 1$. We make the following non-degenerateness assumption, which is assumed to hold throughout this work. It is reasonable in the federated learning setting we consider.

Assumption 10. The points $w_1, \cdots, w_k$ are not collinear.

The geometric median is defined as any minimizer of

$$g(z) := \sum_{k=1}^m \alpha_k \|z - w_k\| \,. \tag{9}$$

Under Assumption 10, g is known to have a unique minimizer - we denote it by $z^\star$.

Given a smoothing parameter $\nu > 0$, its smoothed variant g_ν is

$$g_\nu(z) := \sum_{k=1}^m \alpha_k \|z - w_k\|_{(\nu)} \,, \tag{10}$$

where

$$\|z\|_{(\nu)} := \max_{u^\top u \le 1} \left\{ u^\top z - \tfrac{\nu}{2} u^\top u \right\} + \tfrac{\nu}{2} = \begin{cases} \frac{1}{2\nu}\|z\|^2 + \frac{\nu}{2}\,, & \|z\| \le \nu \\ \|z\|\,, & \|z\| > \nu \end{cases} . \tag{11}$$

In case $\nu = 0$, we define $g_0 \equiv g$. Beck and Teboulle [8] show that $\|\cdot\|_{(\nu)}$ is $(1/\nu)$-smooth and that

$$0 \le \|\cdot\|_{(\nu)} - \|\cdot\| \le \nu/2 \tag{12}$$

Under Assumption 10, g_ν has a unique minimizer as well, denoted by $z_\nu^\star$. We call $z_\nu^\star$ as the ν-smoothed geometric median.

We let R denote the diameter of the convex hull of $\{w_1, \cdots, w_m\}$, i.e.,

$$R := \mathrm{diam}(\mathrm{conv}\{w_1, \cdots, w_m\}) = \max_{z, z' \in \mathrm{conv}\{w_1, \cdots, w_m\}} \|z - z'\| \,. \tag{13}$$

We also assume that $\nu < R$, since for all $\nu \ge R$, the function g_ν is simply a quadratic for all $z \in \mathrm{conv}\{w_1, \cdots, w_m\}$.

A.2 Weiszfeld's Algorithm: Review

The Weiszfeld [63] algorithm performs the iterations

$$z^{(t+1)} = \begin{cases} \left(\sum_{k=1}^m \beta_k^{(t)} w_k\right) / \left(\sum_{k=1}^m \beta_k^{(t)}\right)\,, & \text{if } z^{(t)} \notin \{w_1, \cdots, w_k\}\,, \\ w_k\,, & \text{if } z^{(t)} = w_k \text{ for some } k\,, \end{cases} \tag{14}$$

where $\beta_k^{(t)} = \alpha_k / \|z^{(t)} - w_k\|$. Kuhn [36, Thm. 3.4] showed that the sequence $\left(z^{(t)}\right)_{t=0}^\infty$ converges to the minimizer of g from (9), provided no iterate coincides with one of the w_k's. We modify Weiszfeld's algorithm to find the smoothed geometric median by considering

$$z^{(t+1)} = \frac{\sum_{k=1}^m \beta_k^{(t)} w_k}{\sum_{k=1}^m \beta_k^{(t)}}\,, \quad \text{where,} \quad \beta_k^{(t)} = \frac{\alpha_k}{\max\left\{\nu, \|z^{(t)} - w_k\|\right\}} \,. \tag{15}$$

This is also stated in Algo. 4. Since each iteration of Weiszfeld's algorithm or its smoothed variant consists in taking a weighted average of the w_k's, the time complexity is $\mathcal{O}(md)$ floating point operations per iteration.

Algorithm 4 The Smoothed Weiszfeld's Algorithm

```
Input: $w_1, \cdots, w_m \in \mathbb{R}^d$, $\alpha_1, \cdots, \alpha_m > 0$ with $\sum_{k=1}^m \alpha_k = 1$, $\nu > 0$, Number of iterations $T$, $z^{(0)} \in \mathrm{conv}\{w_1, \cdots, w_m\}$.
1: for $t = 0, 1, \cdots, T-1$ do
2:     Set $\eta_k^{(t)} = \max\left\{\nu, \|z^{(t)} - w_k\|\right\}$ and $\beta_k^{(t)} = \alpha_k / \eta_k^{(t)}$ for $k = 1, \cdots, m$.
3:     Set $z^{(t+1)} = \left(\sum_{k=1}^m \beta_k^{(t)} w_k\right) / \left(\sum_{k=1}^m \beta_k^{(t)}\right)$.
Output: $z^{(T)}$.
```

A.3 Derivation

We now derive Weiszfeld's algorithm with smoothing as as an alternating minimization algorithm or as an iterative minimization of a majorizing objective.

Surrogate Definition. Consider $\eta = (\eta_1, \cdots, \eta_m) \in \mathbb{R}^m$ and define $G : \mathbb{R}^d \times \mathbb{R}^m_{++} \to \mathbb{R}$ as

$$G(z, \eta) = \frac{1}{2} \sum_{k=1}^m \alpha_k \left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k \right) . \tag{16}$$

Note firstly that G is jointly convex in z, η over its domain.

The first claim shows how to recover g and g_ν from G.

Claim 11. *Consider g, g_ν and G defined in Equations (9), (10) and (16), and fix $\nu > 0$. Then we have the following:*

$$g(z) = \inf_{\eta_1, \cdots, \eta_k > 0} G(z, \eta) , \quad \textit{and,} \tag{17}$$

$$g_\nu(z) = \min_{\eta_1, \cdots, \eta_k \geq \nu} G(z, \eta) . \tag{18}$$

Proof. Define $G_k : \mathbb{R}^d \times \mathbb{R}_{++} \to \mathbb{R}$ by

$$G_k(z, \eta_k) := \frac{1}{2} \left(\frac{\|z - w_k\|^2}{\eta_k} + \eta_k \right) ,$$

so that $G(z, \eta) = \sum_{k=1}^m \alpha_k G_k(z, \eta_k)$.

Since $\eta_k > 0$, the arithmetic-geometric mean inequality implies that $G_k(z, \eta_k) \geq \|z - w_k\|$ for each k. When $\|z - w_k\| > 0$, the inequality above holds with equality when $\|z - w_k\|^2 / \eta_k = \eta_k$, or equivalently, $\eta_k = \|z - w_k\|$. On the other hand, when $\|z - w_k\| = 0$, let $\eta_k \to 0$ to conclude that

$$\inf_{\eta_k > 0} G_k(z, \eta_k) = \|z - w_k\| .$$

For the second part, we note that if $\|z - w_k\| \geq \nu$, then $\eta_k = \|z - w_k\| \geq \nu$ minimizes $G_k(z, \eta_k)$, so that $\min_{\eta_k \geq \nu} G_k(z, \eta_k) = \|z - w_k\|$. On the other hand, when $\|z - w_k\| < \nu$, we note that $G_k(z, \cdot)$ is minimized over $[\nu, \infty)$ at $\eta_k = \nu$, in which case we get $G_k(z, \eta) = \|z - w_k\|^2 / (2\nu) + \nu/2$. From (11), we conclude that

$$\min_{\eta_k \geq \nu} G_k(z, \eta_k) = \|z - w_k\|_{(\nu)} .$$

The proof is complete since $G(z, \eta) = \sum_{k=1}^m \alpha_k G_k(z, \eta_k)$. □

Claim 11 now allows us to consider the following problem in lieu of minimizing g_ν from (10).

$$\min_{\substack{z \in \mathbb{R}^d, \\ \eta_1, \cdots, \eta_m \geq \nu}} G(z, \eta) . \tag{19}$$

Alternating Minimization. Next, we consider an alternating minimization algorithm to minimize G in z, η. The classical technique of alternating minimization method, known also as the block-coordinate method [see, e.g., 9], minimizes a function $f : X \times Y \to \mathbb{R}$ using the updates

$$x^{(t+1)} = \arg\min_{x \in X} f(x, y^{(t)}) \quad \text{and, } y^{(t+1)} = \arg\min_{y \in Y} f(x^{(t+1)}, y) \,.$$

Application of this method to Problem (19) yields the updates

$$\begin{aligned} \eta^{(t)} &= \arg\min_{\eta_1, \cdots, \eta_m \geq \nu} G(z^{(t)}, \eta) = \left(\arg\min_{\eta_k \geq \nu} \left\{ \frac{\|z^{(t)} - w_k\|^2}{\eta_k} + \eta_k \right\} \right)_{k=1}^{m} , \\ z^{(t+1)} &= \arg\min_{z \in \mathbb{R}^d} G(z, \eta^{(t)}) = \arg\min_{z \in \mathbb{R}^d} \sum_{k=1}^{m} \frac{\alpha_k}{\eta_k^{(t)}} \|z - w_k\|^2 \,. \end{aligned} \tag{20}$$

These updates can be written in closed form as

$$\begin{aligned} \eta_k^{(t)} &= \max\{\nu, \|z^{(t)} - w_k\|\} \,, \\ z^{(t+1)} &= \left(\sum_{k=1}^{m} \frac{\alpha_k}{\eta_k^{(t)}} w_k \right) \Big/ \left(\sum_{k=1}^{m} \frac{\alpha_k}{\eta_k^{(t)}} \right) . \end{aligned} \tag{21}$$

This gives the smoothed Weiszfeld's algorithm, as pointed out by the following claim.

Claim 12. *For any fixed $\nu > 0$ and starting point $z^{(0)} \in \mathbb{R}^d$, the sequences $\left(z^{(t)}\right)$ produced by (15) and (21), and hence, (20) are identical.*

Proof. Follows from plugging in the expression from $\eta_k^{(t)}$ in the update for $z^{(t+1)}$ in (21). □

Majorization-Minimization. We now instantiate the smoothed Weiszfeld's algorithm as a majorization-minimization scheme. In particular, it is the iterative minimization of a first-order surrogate in the sense of Mairal [45, 46].

Define $g_\nu^{(t)} : \mathbb{R}^d \to \mathbb{R}$ as

$$g_\nu^{(t)}(z) := G(z, \eta^{(t)}) \,, \tag{22}$$

where $\eta^{(t)}$ is as defined in (20). The z-step of (20) simply sets $z^{(t+1)}$ to be the minimizer of $g_\nu^{(t)}$.

We note the following properties of $g_\nu^{(t)}$.

Claim 13. *For $g_\nu^{(t)}$ defined in (22), the following properties hold:*

$$g_\nu^{(t)}(z) \geq g_\nu(z) \,, \quad \textit{for all} z \in \mathbb{R}^d \,, \tag{23}$$

$$g_\nu^{(t)}(z^{(t)}) = g_\nu(z^{(t)}) \,, \quad \textit{and,} \tag{24}$$

$$\nabla g_\nu^{(t)}(z^{(t)}) = \nabla g_\nu(z^{(t)}) \,. \tag{25}$$

Moreover $g^{(t)}$ can also be written as

$$g_\nu^{(t)}(z) = g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z - z^{(t)}\right) + \frac{L^{(t)}}{2} \|z - z^{(t)}\|^2 \,, \tag{26}$$

where

$$L^{(t)} := \sum_{k=1}^{m} \frac{\alpha_k}{\eta_k^{(t)}} \,. \tag{27}$$

Proof. The first part follows because

$$g_\nu(z) = \min_{\eta_1,\cdots,\eta_m} G(z,\eta) \le G(z,\eta^{(t)}) = g_\nu^{(t)}(z)\,.$$

For Eq. (24), note that the inequality above is an equality at $z^{(t)}$ by the definition of $\eta^{(t)}$ from (20). To see (25), note that

$$\nabla g_\nu(z) = \sum_{k=1}^m \alpha_m \frac{z - w_k}{\max\{\nu, \|z-w_k\|\}}\,.$$

Then, by the definition of $\eta^{(t)}$ from (21), we get that

$$\nabla g_\nu(z^{(t)}) = \sum_{k=1}^m \frac{\alpha_m}{\eta_k^{(t)}}(z^{(t)} - w_k) = \nabla g_\nu^{(t)}(z^{(t)})\,.$$

The obtain the expansion (26), we write out the Taylor expansion of the quadratic $g^{(t)}(z)$ around $z^{(t)}$ to get

$$g_\nu^{(t)}(z) = g_\nu^{(t)}(z^{(t)}) + \nabla g_\nu^{(t)}(z^{(t)})^\top \left(z - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z - z^{(t)}\|^2\,,$$

and complete the proof by plugging in (24) and (25). □

Gradient Descent. The next claim rewrites the smoothed Weiszfeld's algorithm as gradient descent on g_ν.

Claim 14. *Equation* (15) *can also be written as*

$$z^{(t+1)} = z^{(t)} - \frac{1}{L^{(t)}}\nabla g_\nu(z^{(t)})\,, \tag{28}$$

where $L^{(t)}$ is as defined in (27).

Proof. Use $z^{(t+1)} = \arg\min_{z\in\mathbb{R}^d} g_\nu^{(t)}(z)$, where $g_\nu^{(t)}$ is written using (26). □

A.4 Properties of Iterates

The first claim reasons about the iterates $z^{(t)}, \eta^{(t)}$.

Claim 15. *Starting from any $z^{(0)} \in \mathrm{conv}\{w_1,\cdots,w_m\}$, the sequences $(\eta^{(t)})$ and $(z^{(t)})$ produced by Algorithm 4 satisfy*

- *$z^{(t)} \in \mathrm{conv}\{w_1,\cdots,w_m\}$ for all $t \ge 0$, and,*
- *$\nu \le \eta_k^{(t)} \le R$ for all $k = 1,\cdots,m$, and $t \ge 1$,*

where $R = \mathrm{diam}(\mathrm{conv}\{w_1,\cdots,w_m\})$. Furthermore, $L^{(t)}$ defined in (27) *satisfies $1/R \le L^{(t)} \le 1/\nu$ for all $t \ge 0$.*

Proof. The first part follows for $t \ge 1$ from the update (15), where Claim 12 shows the equivalence of (15) and (20). Then case of $t = 0$ is assumed. The second part follows from (21) and the first part. The bound on $L^{(t)}$ follows from the second part since $\sum_{k=1}^m \alpha_k = 1$. □

The next result shows that it is a descent algorithm. Note that the non-increasing nature of the sequence $\left(g_\nu(z^{(t)})\right)$ also follows from the majorization-minimization viewpoint [46]. Here, we show that this sequence is strictly decreasing. Recall that $z_\nu^\star$ is the unique minimizer of g_ν.

Lemma 16. *The sequence $(z^{(t)})$ produced by Algorithm 4 satisfies $g_\nu(z^{(t+1)}) < g_\nu(z^{(t)})$ unless $z^{(t)} = z_\nu^\star$.*

Proof. Let $\mathcal{E}_\nu = \{\eta \in \mathbb{R}^m \,:\, \eta_k \geq \nu \text{ for } k = 1, \cdots, m\}$. Starting with (18), we successively deduce,

$$\begin{aligned} g_\nu(z^{(t+1)}) &= \min_{\eta_1,\cdots,\eta_m \geq \nu} G(z^{(t+1)}, \nu) \\ &\leq G(z^{(t+1)}, \eta^{(t)}) \\ &= \min_{z \in \mathbb{R}^d} G(z, \eta^{(t)}) \\ &\leq G(z^{(t)}, \eta^{(t)}) \\ &= \min_{\eta_1,\cdots,\eta_m \geq \nu} G(z^{(t)}, \eta) \\ &= g_\nu(z^{(t)})\,. \end{aligned}$$

Here, we used the fact that $z^{(t+1)}$ minimizes $G(\cdot, \eta^{(t)})$ over $\mathbb{R}^d$ and that $\eta^{(t)}$ minimizes $G(z^{(t)}, \cdot)$ over $\mathcal{E}_\nu$.

Suppose now that $g_\nu(z^{(t+1)}) = g_\nu(z^{(t)})$. In this case, both the inequalities above hold with equality. Since $G(\cdot, \eta^{(t)})$ is $L^{(t)}$-strongly convex where $L^{(t)} \geq 1/R$ (cf. Claim 15), this implies that $z^{(t)} = \arg\min_{z\in\mathbb{R}^d} G(z, \eta^{(t)})$. By definition then, $\eta^{(t+1)} = \eta^{(t)}$ is the unique minimizer of $G(z^{(t)}, \cdot)$ over S, since $G(z^{(t)}, \cdot)$ is strictly convex. The associated first-order optimality conditions are the following:

$$\nabla_z G(z^{(t)}, \eta^{(t)}) = 0\,, \quad \text{and,} \quad \nabla_\eta G(z^{(t)}, \eta^{(t)})^\top (\eta - \eta^{(t)}) \geq 0 \quad \forall \eta \in \mathcal{E}_\nu\,.$$

Putting these together, we find that the pair $(z^{(t)}, \eta^{(t)})$ satisfies the first-order optimality conditions for G over the domain $\mathbb{R}^d \times \mathcal{E}_\nu$. Hence, $z^{(t)} = z^\star_\nu$. □

The next lemma shows that $\|z^{(t)} - z^\star\|$ is non-increasing. This property was shown by Beck and Sabach [7, Corollary 5.1] for the case of Weiszfeld algorithm without smoothing.

Lemma 17. *The sequence $(z^{(t)})$ produced by Algorithm 4 satisfies for all $t \geq 0$,*

$$\|z^{(t+1)} - z^\star_\nu\| \leq \|z^{(t)} - z^\star_\nu\|\,.$$

Furthermore, if $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$, then it holds that

$$\|z^{(t+1)} - z^\star\| \leq \|z^{(t)} - z^\star\|\,.$$

Proof. We adapt the proof of Beck and Sabach [7]. First note from Claim 14 that

$$\nabla g_\nu(z^{(t)}) = L^{(t)}(z^{(t)} - z^{(t+1)})\,, \tag{29}$$

where $L^{(t)}$ is defined in (27). Starting from the results of Claim 13, we observe for any z that,

$$\begin{aligned} g_\nu(z^{(t+1)}) &\overset{(23)}{\leq} g_\nu^{(t)}(z^{(t+1)}) \\ &\overset{(26)}{=} g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\ &\overset{(*)}{\leq} g_\nu(z) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\ &\overset{(29)}{=} g_\nu(z) + L^{(t)}\left(z^{(t)} - z^{(t+1)}\right)^\top \left(z^{(t+1)} - z\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2\,, \end{aligned}$$

where $(*)$ following from the convexity of g_ν as $g_\nu(z) \geq g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top (z - z^{(t)})$. Next, we use the Pythagorean identity: for any $a, b, c \in \mathbb{R}^d$, it holds that

$$\|b - a\|^2 + 2(b-a)^\top (a - c) = \|b - c\|^2 - \|a - c\|^2\,.$$

With this, we get,

$$g_\nu(z^{(t+1)}) \leq g_\nu(z) + \frac{L^{(t)}}{2}\left(\|z^{(t)} - z\|^2 - \|z^{(t+1)} - z\|^2\right)\,.$$

Plugging in $z = z^\star_\nu$, the fact that $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$ implies that $\|z^{(t+1)} - z^\star_\nu\|^2 \leq \|z^{(t)} - z^\star_\nu\|^2$, since $L^{(t)} \geq 1/R$ is strictly positive. Likewise, for $z = z^\star$, the claim holds under the condition that $g_\nu(z^{(t+1)}) \geq g_\nu(z^\star)$. □

A.5 Rate of Convergence

We are now ready to prove the global sublinear rate of convergence of Algorithm 4.

Theorem 18. *The iterate $z^{(t)}$ produced by Algorithm 4 with input $z^{(0)} \in \mathrm{conv}\{w_1, \cdots, w_k\}$ and $\nu > 0$ satisfies*

$$g_\nu(z^{(t)}) - g_\nu(z_\nu^\star) \le \frac{2\|z^{(0)} - z_\nu^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} \le \frac{2\|z^{(0)} - z_\nu^\star\|^2}{\widehat{\nu} t} ,$$

where $L^{(\tau)} = \sum_{k=1}^m \alpha_k / \eta_k^{(\tau)}$ is defined in (27), and

$$\widehat{\nu} = \min_{\tau=0,\cdots,t-1} \min_{k\in[m]} \max\{\nu, \|z^{(\tau)} - w_k\|\} \ge \nu . \tag{30}$$

Furthermore, it holds that

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} + \frac{\nu}{2} \le \frac{2\|z^{(0)} - z^\star\|^2}{\widehat{\nu} t} + \frac{\nu}{2} .$$

Proof. With the descent and contraction properties of Lemmas 16 and 17 respectively, the proof now follows the classical proof technique of gradient descent [e.g., 54, Theorem 2.1.13]. Starting from the results of Claim 13, we observe for any z that,

$$\begin{aligned} g_\nu(z^{(t+1)}) &\overset{(23)}{\le} g_\nu^{(t)}(z^{(t+1)}) \\ &\overset{(26)}{=} g_\nu(z^{(t)}) + \nabla g_\nu(z^{(t)})^\top \left(z^{(t+1)} - z^{(t)}\right) + \frac{L^{(t)}}{2}\|z^{(t+1)} - z^{(t)}\|^2 \\ &\overset{(28)}{=} g_\nu(z^{(t)}) - \frac{1}{2L^{(t)}}\|\nabla g_\nu(z^{(t)})\|^2 . \end{aligned} \tag{31}$$

Convergence on g_ν. For ease of notation, we let $\widetilde{\Delta}_t := g_\nu(z^{(t)}) - g_\nu(z_\nu^\star)$. We assume now that $\widetilde{\Delta}_{t+1}$ is nonzero, and hence, so is $\widetilde{\Delta}_t$ (Lemma 16). If $\widetilde{\Delta}_{t+1}$ were zero, then the theorem would hold trivially at $t+1$.

Now, from convexity of g_ν and the Cauchy-Schwartz inequality, we get that

$$\widetilde{\Delta}_t \le \nabla g_\nu(z^{(t)})^\top \left(z^{(t)} - z_\nu^\star\right) \le \|\nabla g_\nu(z^{(t)})\| \|z^{(t)} - z_\nu^\star\| .$$

Plugging this in, we get,

$$\begin{aligned} \widetilde{\Delta}_{t+1} - \widetilde{\Delta}_t &\le -\frac{1}{2L^{(t)}} \frac{\widetilde{\Delta}_t^2}{\|z^{(t)} - z_\nu^\star\|^2} \\ &\le -\frac{1}{2L^{(t)}} \frac{\widetilde{\Delta}_t^2}{\|z^{(0)} - z_\nu^\star\|^2} , \end{aligned}$$

where we invoked Lemma 17.

Now, we divide by $\widetilde{\Delta}_t \widetilde{\Delta}_{t+1}$, which is nonzero by assumption, and use $\widetilde{\Delta}_t / \widetilde{\Delta}_{t+1} \ge 1$ (Lemma 16) to get

$$\begin{aligned} \frac{1}{\widetilde{\Delta}_t} - \frac{1}{\widetilde{\Delta}_{t+1}} &\le -\frac{1}{2L^{(t)}} \left(\frac{\widetilde{\Delta}_t}{\widetilde{\Delta}_{t+1}}\right) \frac{1}{\|z^{(0)} - z_\nu^\star\|^2} \\ &\le -\frac{1}{2L^{(t)} \|z^{(0)} - z_\nu^\star\|^2} . \end{aligned}$$

Telescoping, we get,

$$\frac{1}{\widetilde{\Delta}_t} \ge \frac{1}{\widetilde{\Delta}_t} - \frac{1}{\widetilde{\Delta}_0} \ge \left(\sum_{\tau=0}^{t-1} \frac{1}{L^{(\tau)}}\right) \frac{1}{2\|z^{(0)} - z_\nu^\star\|^2} .$$

This proves the first inequality to be proved. The second inequality follows from the definition in Eq. (27) since $\sum_{k=1}^m \alpha_k = 1$.

Convergence on g. The proof follows along the same ideas as the previous proof. Define $\Delta_t := g_\nu(z^{(t)}) - g_\nu(z^\star)$. Suppose $\Delta_t > 0$. Then, we proceed as previously for any $\tau < t$ to note by convexity and Cauchy-Schwartz inequality that

$$\Delta_\tau \le \|\nabla g_\nu(z^{(\tau)})\| \|z^{(\tau)} - z^\star\| \,.$$

Again, plugging this into (31), using that $\Delta_\tau / \Delta_{\tau+1} \ge 1$ and invoking Lemma 17 gives (since $\Delta_\tau > 0$)

$$\frac{1}{\Delta_\tau} - \frac{1}{\Delta_{\tau+1}} \le -\frac{1}{2L^{(\tau)}\|z^{(0)} - z^\star\|^2} \,.$$

Telescoping and taking the reciprocal gives

$$g_\nu(z^{(t)}) - g_\nu(z^\star) = \Delta_t \le \frac{2\|z^{(0)} - z^\star\|^2}{\sum_{\tau=0}^{t-1} 1/L^{(\tau)}} \,.$$

Using (12) completes the proof for the case that $\Delta_t > 0$. Note that if $\Delta_t \le 0$, it holds that $\Delta_{t'} \le 0$ for all $t' > t$. In this case, $g_\nu(z^{(t)}) - g_\nu(z^\star) \le 0$. Again, (12) implies that $g(z^{(t)}) - g(z^\star) \le \nu/2$, which is trivially upper bounded by the quantity stated in the theorem statement. This completes the proof. □

A.6 Faster Rate of Convergence

We now make an additional assumption:

Assumption 19. The geometric median $z^\star$ does not coincide with any of $w_1, \cdots, w_m$. In other words,

$$\widetilde{\nu} := \min_{k=1,\cdots,m} \|z^\star - w_k\| > 0 \,. \tag{32}$$

Remark 20. *Beck and Sabach [7, Lemma 8.1] show a lower bound on $\widetilde{\nu}$ in terms of $\alpha_1, \cdots, \alpha_m$ and $w_1, \cdots, w_m$.*

Now, we analyze the condition under which the $z^\star = z^\star_\nu$.

Lemma 21. *Under Assumption 19, we have that $z^\star = z^\star_\nu$ for all $\nu \le \widetilde{\nu}$, where $\widetilde{\nu}$ is defined in (32).*

Proof. By the definition of the smooth norm in (11), we observe that $\|z^\star - w_k\|_{(\nu)} = \|z^\star - w_k\|$ for all $\nu \le \widetilde{\nu}$, and hence, $g_\nu(z^\star) = g(z^\star)$. For any $z \in \mathbb{R}^d$, we have,

$$g_\nu(z) \overset{(12)}{\ge} g(z) \ge g(z^\star) = g_\nu(z^\star) \,,$$

or that $z^\star = z^\star_\nu$. □

In this case, we get a better rate on the non-smooth objective g.

Corollary 22. *Consider the setting of Theorem 18 where Assumption 19 holds and $\nu \le \widetilde{\nu}$. Then, the iterate $z^{(t)}$ produced by Algorithm 4 satisfies,*

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\widehat{\nu} t} \,,$$

where $\widehat{\nu}$ is defined in Eq. (30).

Proof. This follows from Theorem 18's bound on $g_\nu(z^{(t)}) - g_\nu(z^\star_\nu)$ with the observations that $g(z^{(t)}) \overset{(12)}{\le} g_\nu(z^{(t)})$ and $g(z^\star) = g_\nu(z^\star)$ (see the proof of Lemma 21). □

The previous corollary obtains the same rate as Beck and Sabach [7, Theorem 8.2], up to constants upon using the bound on $\widetilde{\nu}$ given by Beck and Sabach [7, Lemma 8.1].

We also get as a corollary a bound on the performance of Weiszfeld's original algorithm without smoothing, although it could be numerically unstable in practice. This bound depends on the actual iterates, so it is not informative about the performance of the algorithm a priori.

Corollary 23. *Consider the setting of Theorem 18. Under Assumption 19, suppose the sequence $(z^{(t)})$ produced by Weiszfeld's algorithm in Eq. (14) satisfies $\|z^{(t)} - w_k\| > 0$ for all t and k, then it also satisfies*

$$g(z^{(t)}) - g(z^\star) \le \frac{2\|z^{(0)} - z^\star\|^2}{\nu^{(t)} t} \,.$$

where $\nu^{(t)}$ is given by

$$\nu^{(t)} = \min\left\{ \widetilde{\nu}, \min_{\tau=0,\cdots,t} \min_{k\in[m]} \|z^{(\tau)} - w_k\| \right\} \,.$$

Proof. Under these conditions, note that the sequence $(z^{(\tau)})_{\tau=0}^t$ produced by the Weiszfeld algorithm without smoothing coincides with the sequence $(z^{(\tau)}_{\nu^{(t)}})_{\tau=0}^t$ produced by the smoothed Weiszfeld's algorithm at level $\nu = \nu^{(t)}$. Now apply Cor. 22. □

A.7 Comparison to Previous Work

We compare the results proved in the preceding section to prior work on the subject.

Comparison to Beck and Sabach [7]. They present multiple different variants of the Weiszfeld algorithm. For a particular choice of initialization, they can guarantee that a rate of the order of $1/\widetilde{\nu}t$. It is not clear how This choice of initialization can be implemented using a secure average oracle since, if at all. This is because it requires the computation of all pairwise distances $\|w_k - w_{k'}\|$. Moreover, a naive implementation of their algorithm could be numerically unstable since it would involve division by small numbers. Guarding against division by small numbers would lead to the smoothed variant considered here. Note that our algorithmic design choices are driven by the federated learning setting.

Comparison to Beck [6]. He studies general alternating minimization algorithms, including the Weiszfeld algorithm as a special case, with a different smoothing than the one considered here. While his algorithm does not suffer from numerical issues arising from division by small numbers, it always suffers a bias from smoothing. On the other hand, the smoothing considered here is more natural in that it reduces to Weiszfeld's original algorithm when $\|z^{(t)} - w_k\| > \nu$, i.e., when we are not at a risk of dividing by small numbers. Furthermore, the bound in Thm. 18 exhibits a better dependence on the initialization $z^{(0)}$.

B Numerical Simulations: Full Details

The section contains a full description of the experimental setup as well as the results.

We start with the dataset and task description in Sec. B.1, hyperparameter choices in Sec. B.2 and evaluation methodology in Sec. B.3. The simulation results are divided into distinct parts: Section B.4 visualizes the principal components of parameter vectors and gradients, Section B.5 compares different algorithms to compute the geometric median, while Section B.6 compares the RFA algorithm to its competitors. Lastly, Section B.7 studies the effect of the hyperparameter choices.

B.1 Datasets and Task Description

We experiment with two tasks, handwritten-letter recognition and character-level language modeling. Since we only have a finite sample $\xi_1, \cdots, \xi_{n_k} \sim \mathcal{D}_k$ for device k, we take its weight $\alpha_k \propto n_k$.

B.1.1 Handwritten-Letter Recognition

The first dataset is the EMNIST dataset [19] for handwritten letter recognition.

Data. Each inpt x is a gray-scale image resized to 28×28. Each output y is categorical variable which takes 62 different values, one per class of letter (0-9, a-z, A-Z).

Formulation. The task of handwritten letter recognition is cast as a multi-class classification problem with 62 classes.

Distribution of Data. The handwritten characters in the images are annotated by the writer of the character as well. We use a non-i.i.d. split of the data grouped by a writer of a given image. We discard devices with less than 100 total input-output pairs (both train and test), leaving a total of 3461 devices. Of these, we sample 1000 devices to use for our simulations, corresponding to about 30% of the data. This selection held constant throughout the simulations. The number of training examples across these devices summarized in the following statistics: median 160, mean 202, standard deviation 77, maximum 418 and minimum 92. This preprocessing was performed using LEAF [15].

Models. For the model φ, we consider two options: a linear model and a convolutional neural network.

- Linear Model: The linear model maintains parameters $w_1, \cdots, w_{62} \in \mathbb{R}^{28\times 28}$. For a given image x, class l is assigned score $\langle w_l, x\rangle$, which is then converted to a probability using a softmax operation as $p_l = \exp(\langle w_l, x\rangle)/\sum_{l'} \exp(\langle w_{l'}, x\rangle)$. For a new input image x, the prediction is made as $\arg\max_l \langle w_l, x\rangle$.
- Convolutional Neural Network (ConvNet): The ConvNet [40] we consider contains two convolutional layers with max-pooling, followed by a fully connected hidden layer, and another fully connected (F.C.) layer with 62 outputs. When given an input image x, the output of this network is assigned as the scores of each of the classes. Probabilities are assigned similar to the linear model with a softmax operation on the scores. The schema of network is given below:

$$\begin{array}{c}\text{Input}\\ 28\times 28\end{array} \longrightarrow \begin{array}{c}\text{Conv2D}\\ \text{filters} = 32\\ \text{kernel} = 5\times 5\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool}\\ \text{kernel} = 2\times 2\\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{Conv2D}\\ \text{filters} = 64\\ \text{kernel} = 5\times 5\end{array} \longrightarrow$$

$$\text{ReLU} \longrightarrow \begin{array}{c}\text{Max Pool}\\ \text{kernel} = 2\times 2\\ \text{stride} = 2\end{array} \longrightarrow \begin{array}{c}\text{F.C.}\\ \text{units} = 2048\end{array} \longrightarrow \text{ReLU} \longrightarrow \begin{array}{c}\text{F.C.}\\ \text{units} = 62\end{array} \longrightarrow \text{score}$$

Loss Function. We use the multinomial logistic loss $\ell(y, p) = -\log p_y$, for probabilities $p = (p_1, \cdots, p_{62})$ and $y \in \{1, \cdots, 62\}$. In the linear model case, it is equivalent to the classical softmax regression.

Evaluation Metric. The model is evaluated based on the classification accuracy on the test set.

B.1.2 Character-Level Language Modeling

The second task is to learn a character-level language model over the Complete Works of Shakespeare [58]. The goal is to read a few characters and predict the next character which appears.

Data. The dataset consists of text from the Complete Works of William Shakespeare as raw text.

Formulation. We formulate the task as a multi-class classification problem with 53 classes (a-z, A-Z, other) as follows. At each point, we consider the previous $H = 20$ characters, and build $x \in \{0, 1\}^{H\times 53}$ as a one-hot encoding of these H characters. The goal is then try to predict the next character, which can belong to 53 classes. In this manner, a text with l total characters gives l input-output pairs.

Distribution of Data. We use a non-i.i.d. split of the data. Each role in a given play (e.g., Brutus from The Tragedy of Julius Caesar) is assigned as a separate device. All devices with less than 100 total examples are discarded, leaving 628 devices. The training set is assigned a random 90% of the input-output pairs, and the other rest are held out for testing. This distribution of training examples is extremely skewed, with the following statistics: median 1170, mean 3579, standard deviation 6367, maximum 70600 and minimum 90. This preprocessing was performed using LEAF [15].

Models. We use a long-short term memory model (LSTM) [26] with 128 hidden units for this purpose. This is followed by a fully connected layer with 53 outputs, the output of which is used as the score for each character. As previously, probabilities are obtained using the softmax operation.

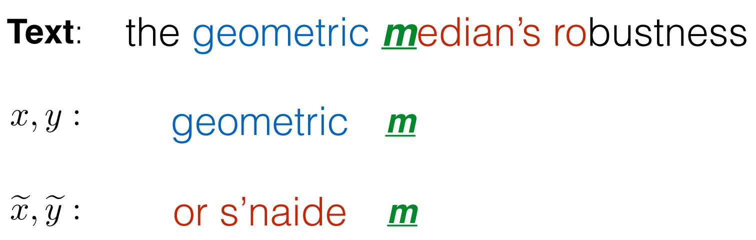

Text: the geometric median's robustness
x, y :
geometric
m
$\widetilde{x}, \widetilde{y}$:
or s'naide
m

Figure 5: Illustration of the data corruption introduced in the Shakespeare dataset. The first line denotes the original text. The second line shows the effective x when predicting the "m" of the word "median". The second line shows the corresponding $\widetilde{x}$ after the introduction of the corruption. Note that $\widetilde{x}$ is the string "edian's ro" reversed.

Loss Function. We use the multinomial logistic loss.

Evaluation Metric. The model is evaluated based on the accuracy of next-character prediction on the test set.

B.2 Methods, Hyperparameters and Variants

We first describe the corruption model, followed by various methods tested.

B.2.1 Corruption Model

Since the goal of this work to test the robustness of federated learning models in the setting of high corruption, we artificially corrupt updates while controlling the level of corruption. We experiment with two different corruption models in the local computation Ψ defined in Sec. 3: data corruption or omniscient corruption.

Data Corruption. : This is an example of static data poisoning. The model training procedure is not modified, but the data fed into the model is modified. In particular, for some modification $\mathcal{D}_k'$ of the local dataset $\mathcal{D}_k$ of client k, we use

$$\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = \Phi(w, H, \mathcal{D}_k') \,.$$

The exact nature of the modification $\mathcal{D}_k'$ depends on the dataset:

- EMNIST: We take the negative of the image x. Mathematically, $\mathrm{d}\mathcal{D}_k'(x, y) = \mathrm{d}\mathcal{D}_k(1 - x, y)$, assuming the pixels of x are normalized to lie in $[0, 1]$.
- Shakespeare: We reverse the original text. Mathematically, $\mathrm{d}\mathcal{D}_k'(c_1 \cdots c_{20}, c_{21}) = \mathrm{d}\mathcal{D}_k(c_{21} \cdots c_2, c_1)$ This is illustrated in Fig. 5.

In both cases, the labels are left unmodified.

Omniscient corruption. This is an example of update poisoning. The data is not modified here but the parameters of a device are directly modified. In particular, $\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S})$ for clients $k \in S_t$ returns

$$\Psi(k, w, H, \mathcal{D}_k, \mathcal{D}, \mathcal{S}) = -\frac{1}{\sum_{k \in S_t \cap \mathcal{C}} \alpha_k} \Bigg(2 \sum_{k \in S_t \setminus \mathcal{C}} \alpha_k \Phi(w^{(t)}, H_k, \mathcal{D}_k) + \sum_{k \in S_t \cap \mathcal{C}} \alpha_k \Phi(w^{(t)}, H_k, \mathcal{D}_k) \Bigg) \,,$$

such that

$$\sum_{k \sim S_t} \alpha_k \Phi_k(w^{(t)}, H_k) = - \sum_{k \sim S_t} \alpha_k \Phi(w^{(t)}, H_k, \mathcal{D}_k) \,.$$

Algorithm 5 The RFA algorithm

```
Input: Function $F$ as in Eq. (1) distributed over $K$ devices, $\epsilon \geq 0$ , fraction $c$, minibatch size $b$, number of
    local epochs $n_e$, learning rate sequence $(\gamma_t)$, initial iterate $w^{(0)}$, secure average oracle $\mathcal{A}$
    Server executes:
 1: for $t = 1, 2, \cdots$ do
 2:     $m \leftarrow \max\{cK, 1\}$
 3:     Let $S_t$ denote a random set of $m$ devices
 4:     Broadcast $w^{(t)}$ to each device $k \in S_t$
 5:     for each device $k \in S_t$ in parallel do
 6:         $w_k^{(t)} \leftarrow \mathit{LocalUpdate}(k, w^{(t)})$
 7:     $\alpha_k^{(t)} \leftarrow n_k / \sum_{j \in S_t} n_j$ for $k \in S_t$
 8:     $w^{(t+1)} \leftarrow \mathrm{GM}\left( (w_k^{(t)})_{k \in S_t}, (\alpha_k^{(t)})_{k \in S_t}, \epsilon \right)$ using Algo. 3

 9: function LocalUpdate$(k, w)$                                        ▷ Run on device $k$
10:     for $i = 1, \cdots, n_k n_e / b$ do
11:         $B_i \leftarrow$ random subset of $\mathcal{D}_k$ of size $b$
12:         $w \leftarrow w - \gamma_t \sum_{f \in B_i} \nabla f(w)/b$
        return $w$
```

In other words, the weighted arithmetic mean of the model parameter is set to be the negative of what it would have other been without the corruption. This corruption model requires full knowledge of the data and server state, and is adversarial in nature.

Implementation details. Given a corruption level ρ, the set of devices which return corrupted updates are selected as follows:

- Start with $\mathcal{C} = \varnothing$.
- Sample device k uniformly without replacement and add to $\mathcal{C}$. Stop when $\sum_{k \in C} \alpha_k$ just exceeds ρ.

Note that we choose $\alpha_k \propto n_k$, where n_k is the number of training samples on device k.

B.2.2 Methods

We compare the following algorithms:

- the FedAvg algorithm [48],
- the RFA algorithm proposed here in Algo. 5,
- the stochastic gradient descent (SGD) algorithm. The local computation Φ_k of device k is simply one gradient step based on a minibatch drawn from $\mathcal{D}_k$, while the aggregation is simply the weighted arithmetic mean.

B.2.3 Hyperparameters

The hyperparameters for each of these algorithms are detailed below.

FedAvg . The FedAvg algorithm requires the following hyperparameters.

- Device Fraction c: This determines the fraction of devices used in each iteration. We use the default $c = 0.1$ for EMNIST and $c = 0.05$ for the Shakespeare dataset, which fall under the recommended range of McMahan et al. [48].
- Batch Size b, and Number of Local Epochs n_e: For the EMNIST dataset, we use $b = 50, n_e = 5$, and for the Shakespeare dataset, we use $b = 10, n_e = 1$. These values were also shown to achieve competitive performance by McMahan et al. [48].
- Learning Rate (γ_t): We use a learning a learning rate scheme $\gamma_t = \gamma_0 C^{\lfloor t/t_0 \rfloor}$, where γ_0 and C were tuned using grid search on validation set (20% held out from the training set) for a fixed time horizon on the uncorrupted data. The

values which gave the highest validation accuracy were used *for all settings* - both corrupted and uncorrupted. The time horizon used was 2000 iterations for the EMNIST linear model, 1000 iterations for the EMNIST ConvNet 200 iterations for Shakespeare LSTM.

- Initial Iterate $w^{(0)}$: Each element of $w^{(0)}$ is initialized to a uniform random variable whose range is determined according to TensorFlow's "glorot_uniform_initializer".

RFA . The RFA algorithm, proposed here in Algo. 5 requires the following hyperparameters which FedAvg also requires.

- Device Fraction c, Batch Size b, Number of Local Epochs n_e, Initial Iterate $w^{(0)}$: We use the same settings as for the FedAvg algorithm. Moreover, Sec. B.7. presents effect of hyperparameters on RFA .
- Learning Rate (γ_t): We use the *same* learning rate and decay scheme found for FedAvg without further tuning.
  In addition, RFA requires the following additional parameters:
- Algorithm: We use the smoothed Weiszfeld's algorithm as default based on the results of Fig. 7a in Sec. B.5 which compares it favorably against mirror-prox.
- Smoothing parameter ν: Based on the interpretation that ν guards against division by small numbers, we simply use $\nu = 10^{-6}$ throughout.
- Robust Aggregation Stopping Criterion: The concerns the stopping criterion used to terminate either the smoothed Weiszfeld's algorithm or mirror-prox. We use two criteria: an iteration budget and a relative improvement condition - we terminate if a given iteration budget has been extinguished, or if the relative improvement in objective value $|g_\nu(z^{(t)}) - g_\nu(z^{(t)})|/g_\nu(z^{(t+1)}) \le 10^{-6}$ is small.

B.3 Evaluation Methodology and Other Details

We specify here the quantities appearing on the x and y axes on the plots, as well as other details.

x Axis. As mentioned in Section 2, the goal of federated learning is to learn the model with as few rounds of communication as possible. Therefore, we evaluate various methods against the number of rounds of communication, which we measure via the number of calls to a secure average oracle.

Note that FedAvg and SGD require one call to the secure average oracle per outer iteration, while RFA could require several. Hence, we also evaluate performance against the number of outer iterations.

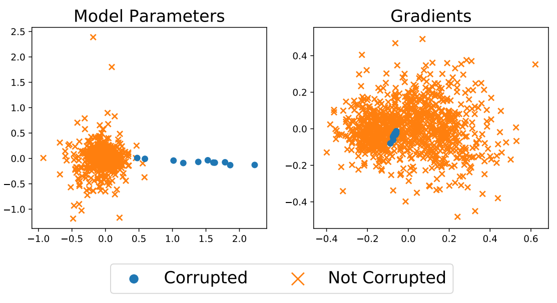
y Axis. We are primarily interested in the test accuracy, which measures the performance on unseen data. We also plot the function value F, which is the quantity our optimization algorithm aims to minimize. We call this the train loss. For completeness, the plots also show the train accuracy as well as the test loss.

Evaluation with Data Corruption. In simulations with data corruption, while the training is performed on corrupted data, we evaluate train and test progress using the corruption-free data.

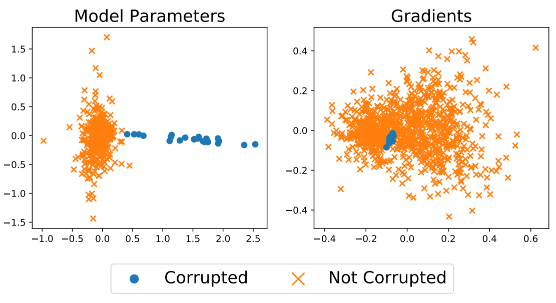
Software. We use the package LEAF [15] to simulate the federated learning setting. The models used are implemented in TensorFlow.

Hardware. Each simulation was run in a simulation as a single process. The EMNIST linear model simulations were run on two workstations with 126GB of memory, with one equipped with Intel i9 processor running at 2.80GHz, and the other with Intel Xeon processors running at 2.40GHz. Simulations involving neural networks were run either on a 1080Ti or a Titan Xp GPU.

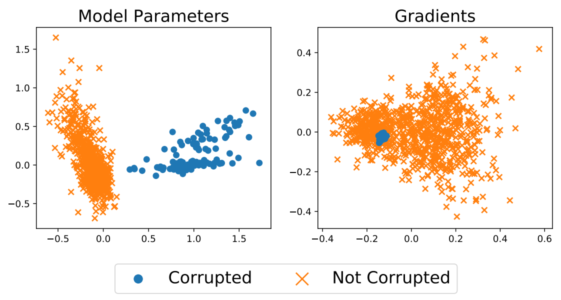
Random runs. Each simulation is repeated 5 times with different random seeds, and the solid lines in the plots here represents the mean over these runs, while the shaded areas show the maximum and minimum values obtained in these runs.



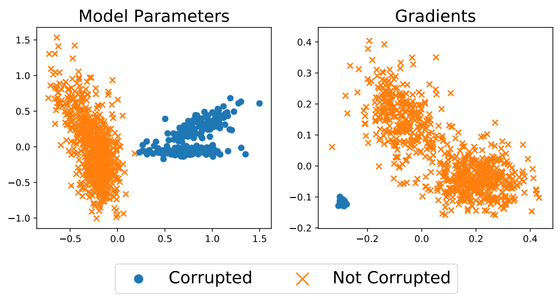
(a) $\rho = 0.01$.



(b) $\rho = 0.02$.



(c) $\rho = 0.1$.



(d) $\rho = 0.25$.

Figure 6: Visualization of the first two principal components of model parameters and gradients for a linear model on the EMNIST dataset under data corruption at different corruption levels ρ. The PCA projection of parameter/gradient vectors of the devices with corrupted data are denoted by blue dots while those of devices with nominal data are denoted by orange 'x'.

B.4 Simulation Results: Visualizing Aggregation of Model Parameters vs. Gradients

In this section we visualize the first two principal components of the model parameters and gradients.

We freeze FedAvg at after 100 iterations on the EMNIST linear model. We run *LocalUpdate* on each of the device and collect their model parameters vectors. We also compute the gradient on 10% of the data on each device and collect all the gradients vectors. We perform principal component analysis (PCA) on each of these collections and visualize the first two principal components in Fig. 6, for different levels of data corruption.

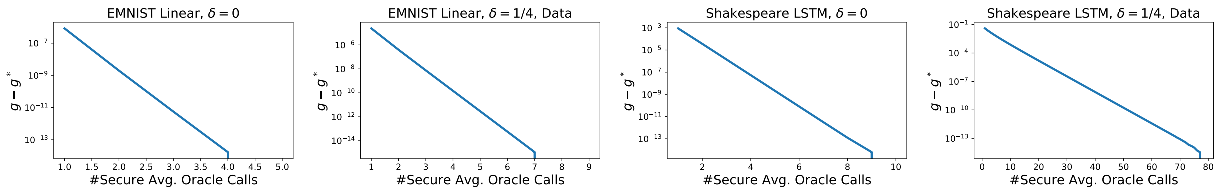
This visualization is useful to note the difference between aggregation of model parameters, the approach taken by this paper, versus the aggregation of gradients, which is the approach taken by most previous work on robust aggregation for distributed learning.

B.5 Simulation Results: Convergence of Smoothed Weiszfeld's Algorithm

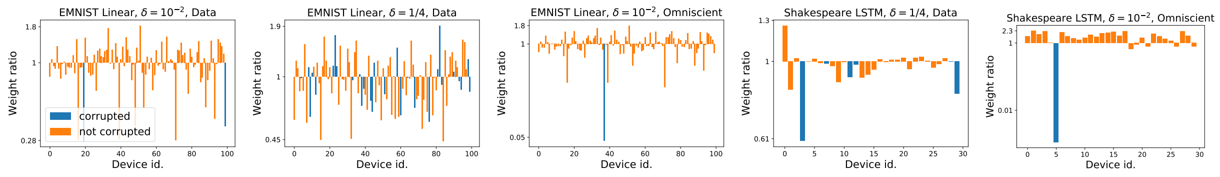
For each of these models, we freeze FedAvg at a certain iteration and try different robust aggregation algorithms. The omniscient corruption was applied manually to to parameters obtained from a run with no corruption as FedAvg diverged in this case (cf. Fig. 9-11).

We find that the smoothed Weiszfeld's algorithm is extremely fast, converging exactly to the smoothed geometric median in a few passes. In fact, the smoothed Weiszfeld's algorithm displays (local) linear convergence, as evidenced by the straight line in log scale. Meanwhile, the straight line in the log-log plots (first two rows) highlights the sublinear convergence of mirror-prox. Further, we also maintain a strict iteration budget of 3 iterations. This choice is also justified in hindsight by the results of Fig. 13.

Next, we visualize the weights assigned by the geometric median to the corrupted updates. Note that the smoothed geometric median $w_1, \cdots, w_m$ is some convex combination $\sum_{k=1}^{m} \beta_k w_k$. This weight β_k of w_k is a measure of the influence of w_k on the aggregate. We plot in Fig. 7b the ratio β_k/α_k for each device k, where α_k is its weight in the arithmetic mean and β_k is obtained by running the smoothed Weiszfeld's algorithm to convergence. We expect this



(a) Convergence of the smoothed Weiszfeld's algorithm and for robust aggregation.



(b) Visualization of the re-weighting of points in the robust aggregate.

Figure 7: Performance of robust aggregation algorithms.

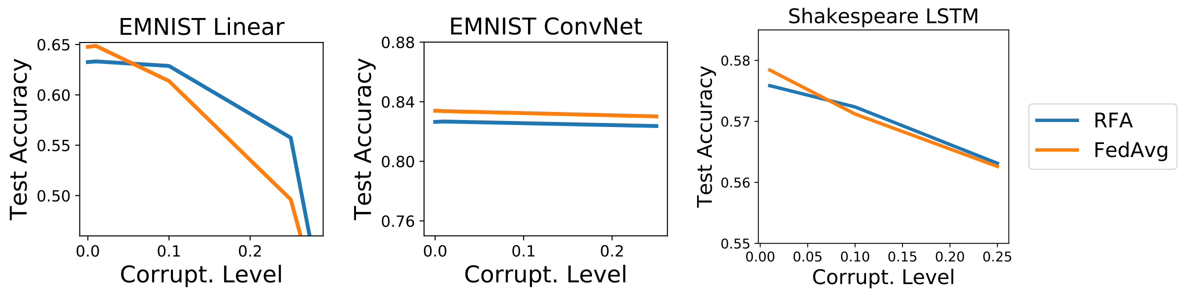


Figure 8: Comparison of robustness of RFA and FedAvg under data corruption.

ratio to be smaller for worse corruptions and ideally zero for obvious corruptions. We find that the smoothed geometric median does indeed assign lower weights to the corruptions, while only accessing the points via a secure average oracle.

B.6 Simulation Results: Comparison of RFA with other methods

Next, we study the effect of the level of corruption on the robustness of RFA and FedAvg . We study two aspects of robustness: the maximum test performance and the performance over the learning process.

We start with the former. Fig. 8 plots the maximum test accuracy of a single run, averaged over five different runs. Each run was allowed to run to convergence, unless a communication budget was exceeded. Only RFA on (EMNIST, ConvNet) failed to converge within this budget. We observe that RFA gives us improved robustness in the case of the linear model, while simple FedAvg is better for the ConvNet model [cf. 66]. On the LSTM model, both $FedAvg$ and RFA give identical curves.

Next, we plot the performance of competing methods versus the number of calls to the secure average oracle and versus the number of outer iterations (the counter t in Algo. 5 in Figures 9 to 11. Since FedAvg converged by round 2000 in Fig. 10, we extend the curve up to end of the time horizon under consideration.

We note that FedAvg is better in the low corruption case $\rho = 0$ or $\rho = 10^{-2}$ under data corruption when measured in the number of calls to the secure average oracle. However, it is identical in performance to RFA when measured in terms of the number of outer iterations. Further, both FedAvg and SGD diverge under omniscient corruption. Fig. 12 shows that RFA converges even under omniscient corruption.

Furthermore, we observe that RFA is qualitatively more stable than FedAvg in its behavior over time in the high data corruption setting of $\rho = 1/4$.

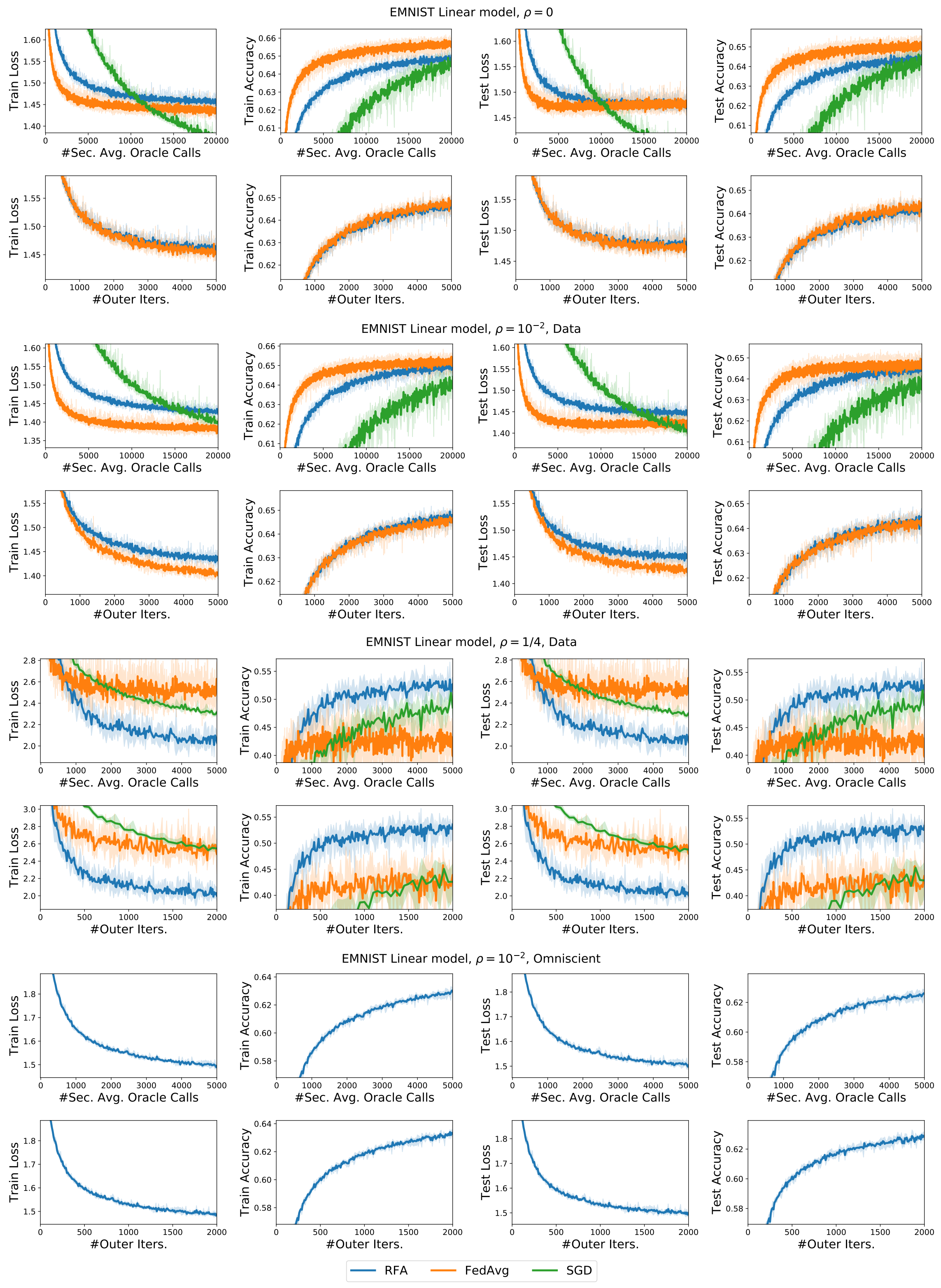


Figure 9: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the EMNIST dataset with a linear model.

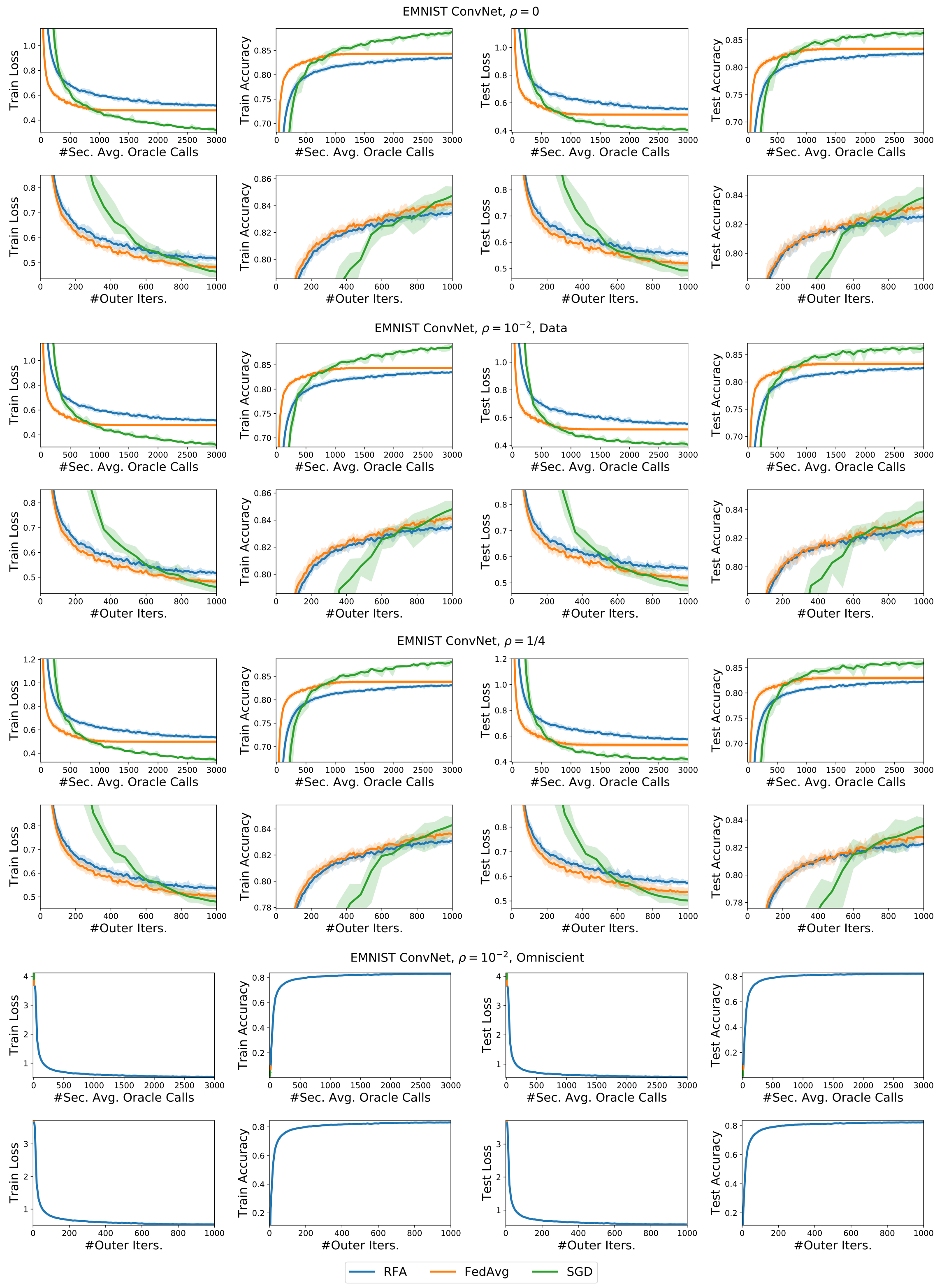


Figure 10: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the EMNIST dataset with a convolutional neural network.

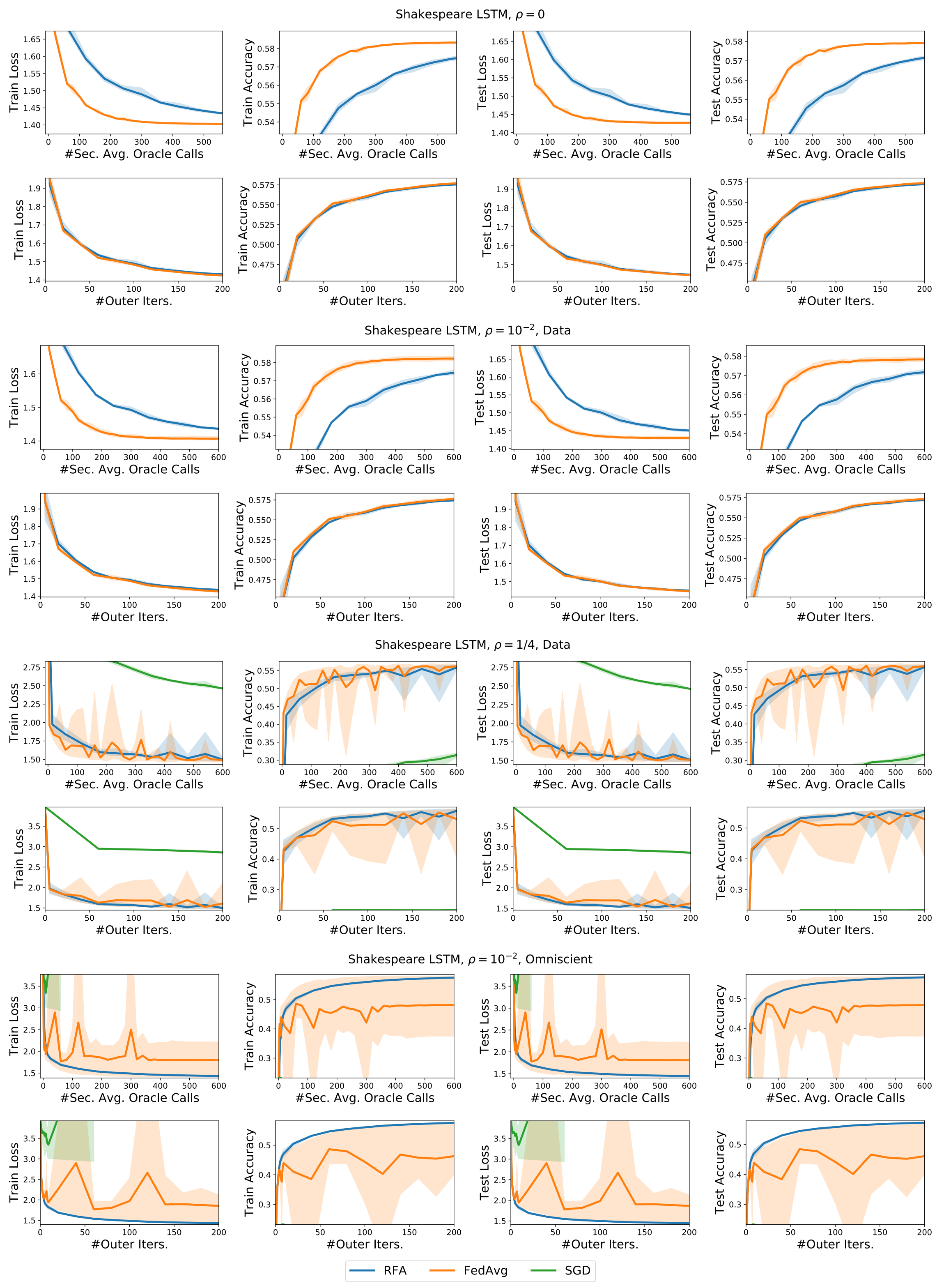


Figure 11: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations on the Shakespeare dataset with an LSTM model.

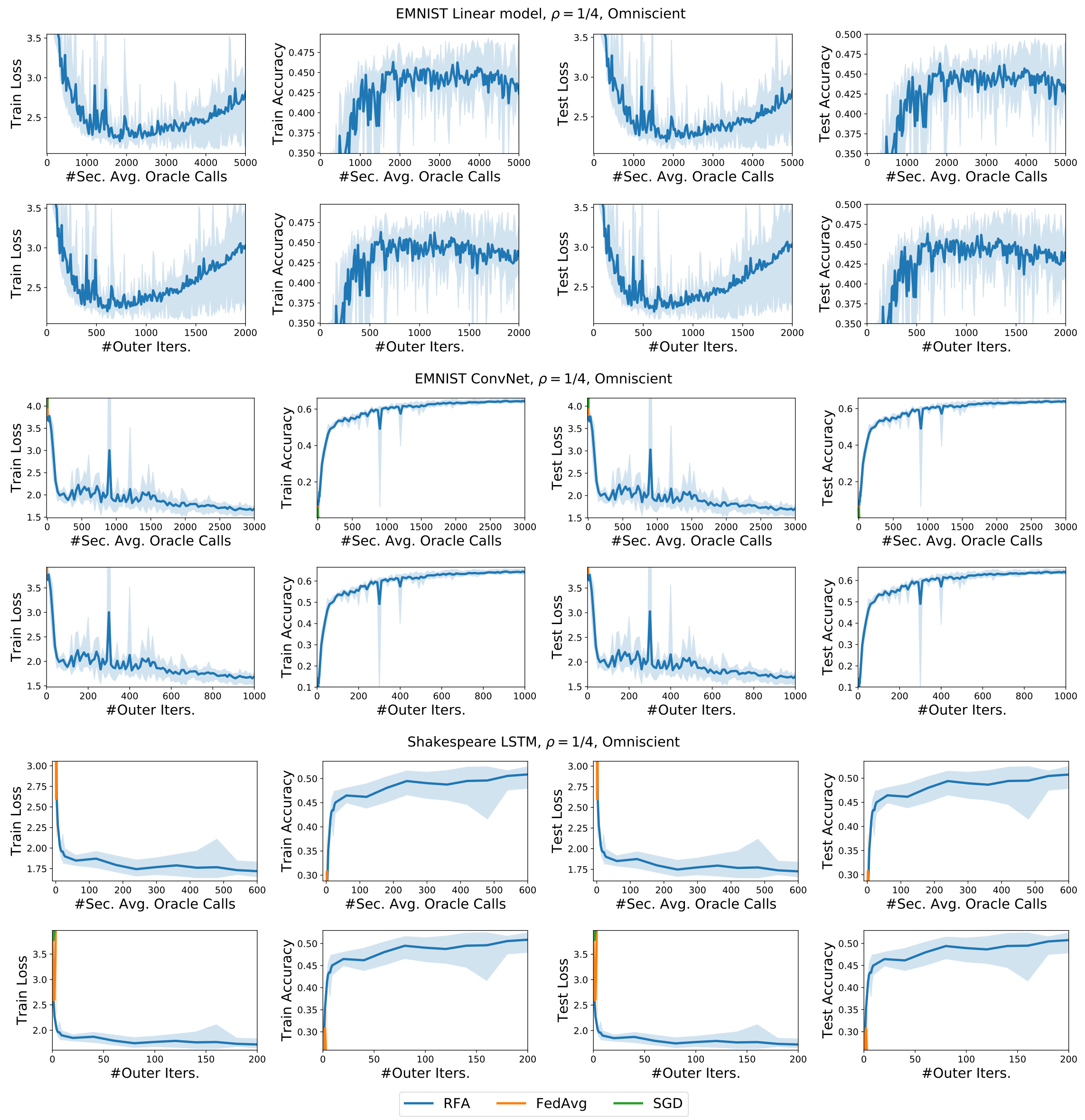


Figure 12: Comparison of methods plotted against number of calls to the secure average oracle or outer iterations for omniscient corruption with $\rho = 1/4$.

B.7 Simulation Results: Hyperparameter Study

Here, we study the effect of the various hyperparameters on RFA .

Effect of Iteration Budget of the Smoothed Weiszfeld's Algorithm. We study the effect of the iteration budget of the smoothed Weiszfeld's algorithm in RFA . in Fig. 13. We observe that a low communication budget is faster in the regime of low corruption, while more iterations work better in the high corruption regime. We used a budget of 3 calls to the secure average oracle to trade-off between the two scenarios in Sec. B.6.

Effect of Number of Devices per Iteration. Figure 14 plots the performance of RFA against the number K of devices chosen per round. We observe the following: in the regime of low corruption, good performance is achieved by selecting 50 devices per round (5%), where as 10 devices per round (1%) is not enough. On the other hand, in high

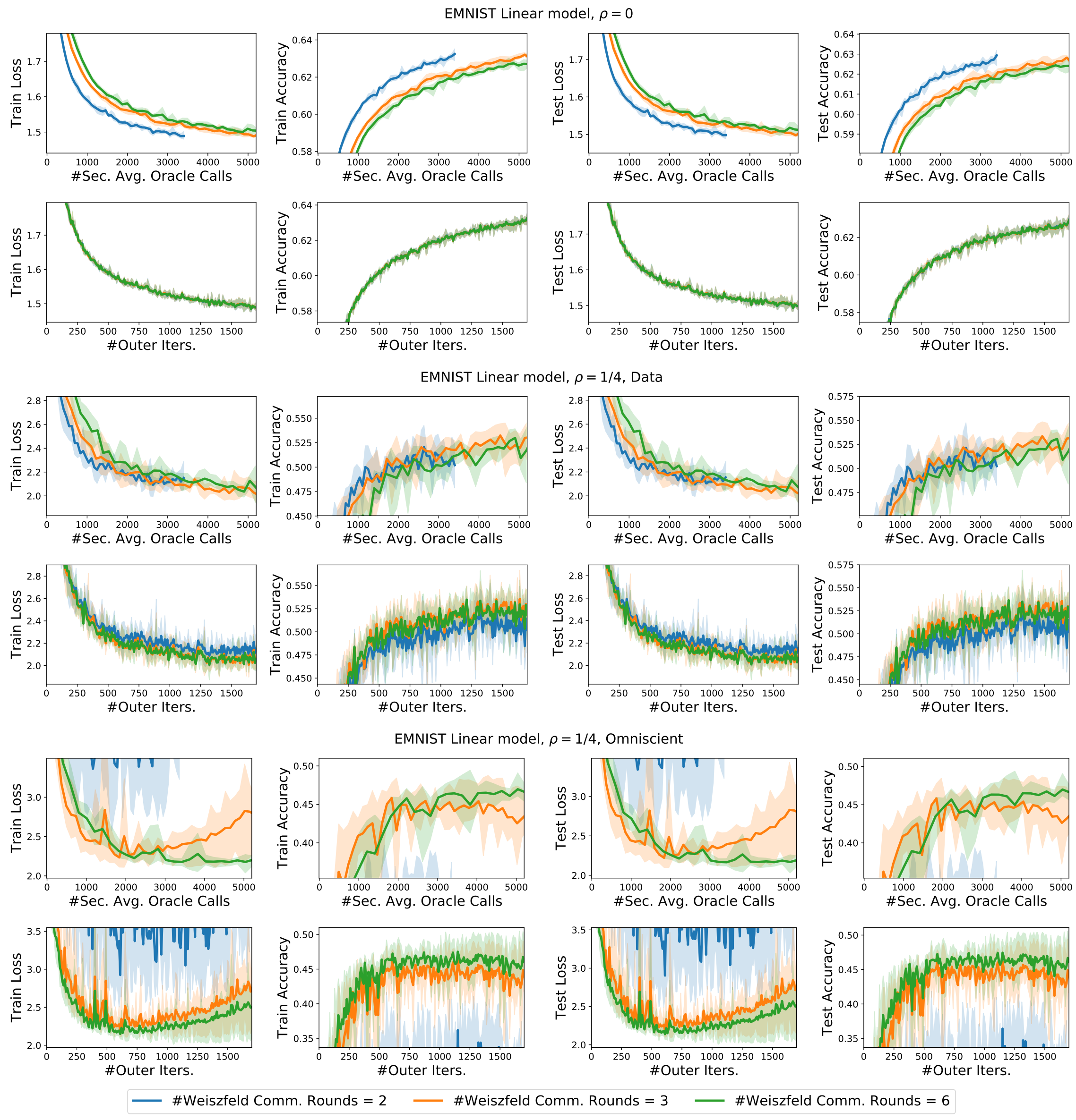


Figure 13: Hyperparameter study: effect of the maximum number of the communication budget on the smoothed Weiszfeld's algorithm in RFA on the EMNIST dataset with a linear model.

corruption regimes, we see the benefit of choosing more devices per round, as a few runs with 10 or 50 devices per round with omniscient corruption at 25% diverged.

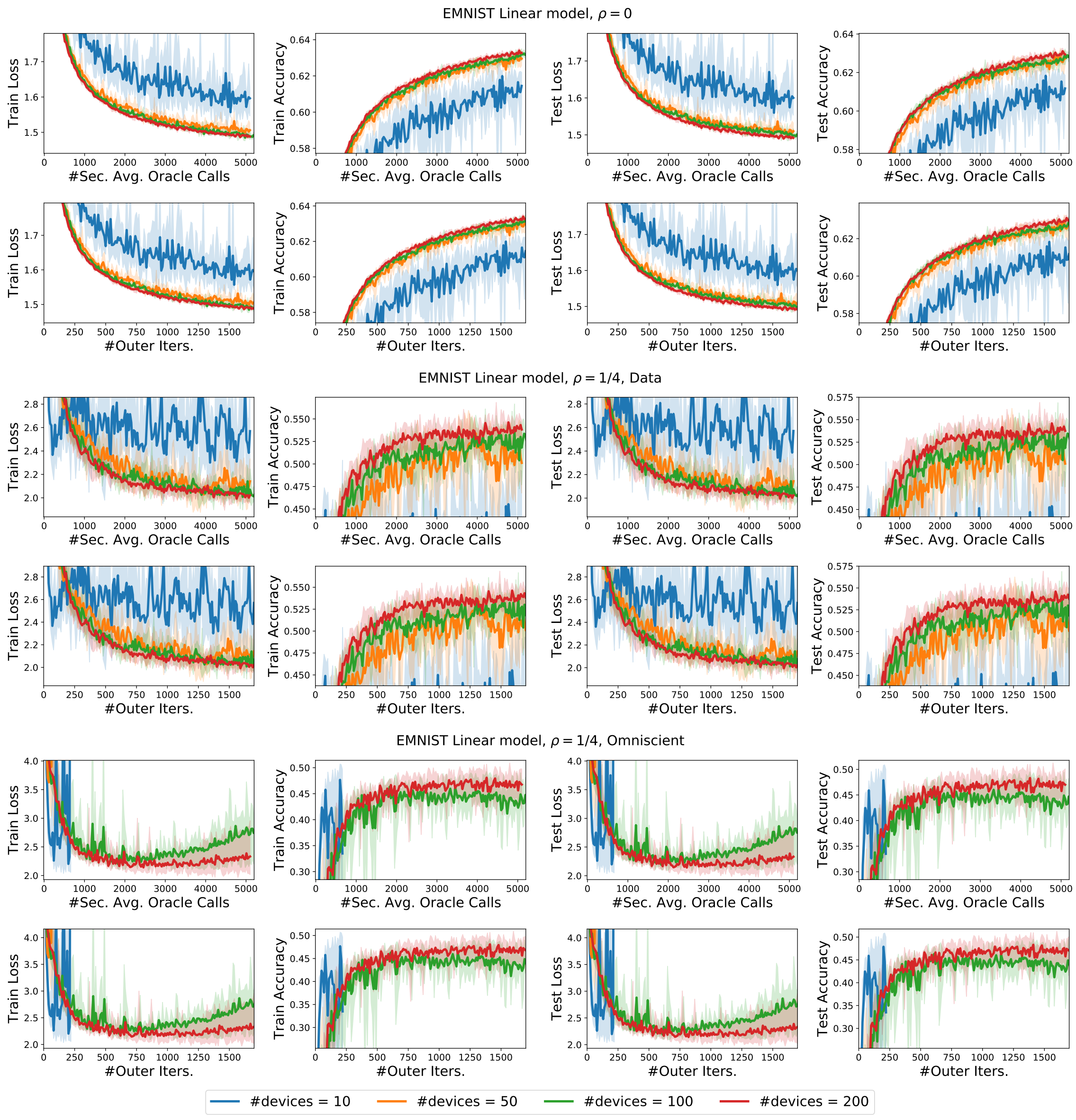


Figure 14: Hyperparameter study: effect of the number of selected client devices per round in RFA on the EMNIST dataset with a linear model.