Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer

Hongyan Chang*[1], Virat Shejwalkar*[2], Reza Shokri[1], Amir Houmansadr[2]
[1]National University of Singapore, [2]University of Massachusetts Amherst
[1]{hongyan, reza}@comp.nus.edu.sg, [2]{vshejwalkar, amir}@cs.umass.edu

***Abstract*—Collaborative (federated) learning enables multiple parties to train a global model without sharing their private data, but through repeated sharing of the parameters of their local models. Each party updates its local model using the aggregation of all parties' parameters, before each round of local training.**

Despite its advantages, this approach has many known privacy and security weaknesses and performance overhead, in addition to being limited only to models with homogeneous architectures. Shared parameters leak a significant amount of information about the local (and supposedly private) datasets. Besides, federated learning is severely vulnerable to poisoning attacks, where some participants can adversarially influence the aggregate parameters. Large models, with high dimensional parameter vectors, are in particular highly susceptible to privacy and security attacks: *curse of dimensionality* in federated learning. We argue that sharing parameters is the most naive way of information exchange in collaborative learning, as they open all the internal state of the model to inference attacks, and maximize the model's malleability by stealthy poisoning attacks.

We propose *Cronus*, a robust collaborative machine learning framework. The simple yet effective idea behind designing Cronus is to control, unify, and significantly reduce the dimensions of the exchanged information between parties, through robust knowledge transfer between their black-box local models. We evaluate all existing federated learning algorithms against poisoning attacks, and we show that Cronus is the only secure method, due to its tight robustness guarantee. Treating local models as black-box, reduces the information leakage through models, and enables us using existing privacy-preserving algorithms that mitigate the risk of information leakage through the model's output (predictions). Cronus also has a significantly lower sample complexity, compared to federated learning, which does not bind its security to the number of participants. It also allows collaboration between models with heterogeneous architectures.

I. Introduction

Collaborative machine learning has recently emerged as a promising approach for building machine learning models using distributed training data held by multiple parties. The training is distributed, and participants repeatedly exchange information about their local models, through an aggregation server. The objective is to enable all the participants to converge to a global model, while keeping their data private. This is very attractive to parties who own sensitive data, and agree on performing a common machine learning task, yet are unwilling to pool their data together for centralized training. Various applications can substantially benefit from collaborative learning. Examples include medical and financial applications, intelligent virtual assistants, speech recognition, keyboard input prediction, and mobile vision [14], [35].

A popular approach for collaborative deep learning, known as federated learning, assumes *homogeneous* local models, i.e., with the same architecture [47], [35]. Every model shares its parameters (or gradients) with a parameter server, after each rounds of training on its local data. The server aggregates the parameter vectors by computing their element-wise mean, and shares the aggregate with the participants. Each party updates its local model with the latest global aggregate, and continues with the next round of local training.

There are major obstacles hindering the scalable deployment of secure and truly privacy-preserving federated learning for sensitive applications. Existing federated learning algorithms are not robust to adversarial updates [12], [13] and backdoor attacks [12], [10], and can leak a significant amount of sensitive information about local datasets [40], [36]. Besides, federated learning cannot be used for aggregating heterogeneous models, for participants that use different models as they have different memory and computing power.

In this paper, our main focus is on protecting collaborative machine learning against poisoning attacks. There exist a long chain of recent poisoning attacks and defenses for federated learning [37], [55], [57], [13], [53], [34], [40], [12], [10]. All existing defenses focus only on replacing the vulnerable aggregation in federated learning with a robust mean function, leaving the rest of the framework intact. But, as it is shown in the literature, and as we show in this paper, all existing (aggregation) algorithms in federated learning are susceptible to some form of poisoning attack, which can significantly damage the global model.

By sharing the model parameters, federated learning completely opens the local models to the (potentially malicious) server and untrusted participants. This is a serious privacy vulnerability as the model parameters leak all the information that the model has about its training data [40], [36]. The problem is significantly worse for large models, with huge number of parameters. This high dimensionality not only magnifies the problem, but also makes it harder to design a defense mechanism with tight (robustness and privacy) guarantees. We refer to this as the *curse of dimensionality* for privacy and security in federated learning.

* Equally contributing authors. Part of the work was done when Virat Shejwalkar was a research intern at NUS.

The theoretical error bound and sample complexity[1] of the existing robust aggregation algorithms [13], [37], [16], [17], [15], [33] depend on the dimensionality of model parameters. The high dimensionality of the models makes the error bound and sample complexity of these robust aggregation algorithms prohibitively high, which makes these algorithms susceptible to some form of poisoning attacks. Besides, federated learning algorithms require overwriting the local models' parameters by the global model. This makes them extremely susceptible to poisoning and backdoor attacks, as the adversary can make small changes in the high-dimensional models which damage the model, but are hard to detect.

Thus, sharing model parameters to transfer the knowledge of local models is a wrong design choice in federated learning, especially considering that this is not the only way of exchanging knowledge between models.

Our contributions. In this paper, we design *Cronus*, a collaborative learning approach to address the fundamental shortcomings of federated learning. Instead of sharing parameters with the server, and updating them by overwriting them with the aggregated parameters, we extract, aggregate, and transfer the knowledge of models in a black-box manner. We use knowledge transfer algorithms, which are used for various compression and regularization purposes in machine learning [21], [9], [54], [7], or as part of a low-sensitivity algorithm to train a model in a centralized setting with differential privacy [43], [44].

In Cronus, we support **heterogeneous** model architectures, as knowledge transfer through distillation allows sharing models via their predictions. For this purpose, we assume that an unlabeled public dataset is available. After a few rounds of local training on their private data, the Cronus parties share their predictions on the public data.

Local models are *fine-tuned* using the aggregated predictions (instead of directly overwriting their parameters), which significantly reduces the potential of harmful updates on local models. Besides, Cronus reduces the dimensions of the update vectors to the size of the model's output, as opposed to the size of the model's parameters. This reduces the dimensionality of the vectors in robustness algorithms, by many orders of magnitudes. Thus, we are able to provide tight provable **robustness** guarantees on the aggregation algorithms, even for small networks. with a dozen of participants.

The distillation process, using disjoint data to the models' training set, is used as a regularization technique [9], and can reduce information leakage about the training data [48]. More importantly, the black-box nature of our algorithm also allows for using privacy-preserving mechanisms by the parties that are tailored to specifically protect the **privacy** of local training data in the black-box setting [39], [18].

Using benchmark datasets, we comprehensively compare the robustness of Cronus with existing federated learning algorithms. We evaluate them against strongest poisoning attacks in the literature. We also design new attacks that breaks a class of aggregation algorithms (based on multiplicative weights updates [8], [34]). We show that each and every of the aggregation schemes used in federated learning (and proposed to make it robust) can be broken using some form of poisoning attacks. The models' accuracy under attack is significantly reduced, sometimes to that of a random guess. However, none of the poisoning attacks can impact the accuracy of models learned using Cronus leaning, beyond at most 2% of drop in prediction accuracy. For the strongest attack on Cronus learning, the reduction in accuracy for Purchase models is 1.6%, for SVHN model is 1.3%, for MNIST model is 1.5%, and for CIFAR10 is 2.1%.

Due to sharing knowledge through predictions, Cronus significantly reduces the risk of active and passive membership inference attacks. The Cronus learning also allows for local party models to prevent sensitive information leakage through their predictions. We train local models with membership privacy through adversarial regularization [39]. Using this approach, we show that, for the evaluated datasets, the accuracy of membership inference attacks drops to near random guessing, with negligible accuracy loss. We empirically show that the stat-of-the-art differential privacy algorithms for deep learning [4], [14] do not provide good prediction accuracy, and are of lower accuracy even compared to stand-alone training.

Finally, we evaluate Cronus with heterogeneous model architectures, and show its effectiveness without compromising the final accuracy of party models.

II. Threat Models

In this work, we investigate two important threats against federated learning algorithms: (1) poisoning attacks against the *robustness* of aggregation schemes used in federated learning [11], [12], [37], and (2) inference attacks against the *privacy* of models' sensitive training data [40], [36].

A. Poisoning attacks

The aim of the poisoning attacks is to indiscriminately jeopardize the accuracy of the resulting models of the federated learning. To achieve this, the adversary controls an ϵ fraction of the total n parties and sends malicious updates to the server through the ϵn malicious parties. We assume that the adversary also has access to some data which is drawn from the same distribution as that of the local training data of the $(1-\epsilon)n$ *benign parties*. Therefore, the adversary has the upper hand in that it can obtain benign updates, and use them to craft effective malicious updates using various poisoning attacks (Section IV). Each (robust) aggregation algorithm has a breaking point with respect to ϵ, i.e., it cannot provide any robustness guarantee if ϵ is larger than the breaking point. We set the fraction of the attackers to be the maximum value which is less than the breaking point, so the robust aggregation algorithms can be evaluated against the strongest attack that they claim they can protect against.

[1] Sample complexity, in the context of federated learning, is the number of parties required to achieve the theoretical error bound.

B. Inference attacks

In federated learning, we consider the information leakage through each party's update, and through the aggregated updates, about the party's local data. The attacker can run membership inference attacks on an individual party's update or on the aggregate of updates from all parties [40]. The adversary's objective is to infer if a particular sample belongs to the private training data of a single party (if target update is of a single party) or of any party (if target update is the aggregate). Following the previous works [36], [40], to evaluate the risk to a party i due to a strong membership inference adversary, we assume that the adversary has knowledge of 50% of members of $\mathbf{d}_i$ and the equal number of non-members. The adversary then trains a binary classifier on various features of the victim models in order to learn to distinguish between the members and non-members of $\mathbf{d}_i$.

III. Aggregation algorithms

In this section, we first describe the setting of federated learning and, various aggregation algorithms used to combine the updates of the collaborating parties and their theoretical robustness guarantees.

A. Federated learning setting

Federated learning [35], [47] enables multiple data holders to train a global model without sharing their data, and through sharing of their training gradients/parameters [13], [6], [37], [55], [57]. For concreteness, below we describe Federated Average (FedAvg) algorithm [35] and its setting, and use it in the rest of our work.

In FedAvg, multiple parties collaborate over multiple epochs to learn a global machine learning model with a classification performance superior to the models learned individually. FedAvg assumes that there are n parties with their *local training datasets*, D_i's, and a *central server* which aggregates the party updates and broadcasts the aggregate to all of the parties. In the t^{th} epoch of FedAvg, parties train the aggregate θ_a^t, broadcast by the server at the end of t^{th} epoch, on their local training data, D_i. Parties use stochastic gradient descent for updating, i.e., $\theta_i^t = \theta_a^t - \nabla_\theta L(D_i; \theta_a^t)$, where $L(D;\theta)$ is loss of θ on data D. Each party then sends the parameters of the locally updated model, θ_i^t, to the server for aggregation. The central server collects all the θ_i^t updates and computes their aggregate $\theta_a^t = f_{\mathsf{AGG}}(\theta_{i\in[n]}^t)$. Specifically, FedAvg uses the weighted average as its f_{AGG}, where the weight of the i^{th} party is determined based on the size of her local training data, i.e., $w_i = \frac{|D_i|}{|D|}$; $|D|$ denotes size of dataset D. The weighted average is formally given by:

$$f_{\mathsf{Mean}}: \quad \theta_a^{t+1} = \sum_{i=1}^{n} \frac{|D_i|}{\sum_{j=1}^{n} |D_j|} \theta_i^t \tag{1}$$

The server then broadcasts the aggregate θ_a^t to all n parties. This process repeats for T epochs or until sufficient accuracy is achieved by the aggregated global model. The procedure is described in Algorithm 1. This algorithm is not robust against poisoning attack and even a single party can destroy the global model. Multiple robust aggregation schemes have been proposed in the literature to improve the resilience of federated learning to poisoning attacks while maintaining the accuracy of the final model [37], [55], [57], [13]. For instance, median is a more robust statistic of data compared to mean [25], [53], [57], and therefore, f_{Mean} in FedAvg can be replaced with weighted median aggregation f_{Median}, which computes coordinate-wise weighted median of the updates.

Algorithm 1 Federated learning algorithm [35], [47]

```
Initialize global model θ_a^0
for t ∈ [T] do
    for i ∈ [n] do                          ▷ Party i's local update
        θ_i^t ← θ_a^t − ∇_θ L(D_i; θ_a^t)
        Return θ_i^t to server
    end for
    θ_a^{t+1} = f_AGG(θ_{i∈[n]}^t)          ▷ Aggregation of updates at server
    Return θ_a^{t+1} to all the parties
end for
```

Table I: Theoretical error rates of aggregation algorithms. n is number of parties, ϵ is breaking point, i.e., malicious parties' fraction, d (d_p) is updates' dimensions, and σ^2 is the variance of each of the dimensions (assume each dimension has the same variance). Cronus, unlike other aggregations, has error rate independent of the dimension of updates.

	Breaking point	Statistical error rate	Computational cost
Mean [35]	$1/n$	Unbounded	$O(nd)$
Median [33]	$1/2$	$O(\sigma\epsilon\sqrt{d})$	$O(nd\log n)$
Krum [13]	$(n-2)/2n$	$O(\sigma n\sqrt{d})$	$O(n^2 d)$
Bulyan [37]	$(n-3)/4n$	$O(\sigma\sqrt{d})$	$O(n^2 d)$
Cronus	$1/2$	$O(\sigma\sqrt{\epsilon})$	$O(d_p^3 + n)$

B. Krum

Weighted average aggregation cannot tolerate even a single malicious party [13], [53]. To solve this problem, Blanchard et al. [13] proposed Krum aggregation, which is based on geometric median of vectors [51], [37]. The intuition behind Krum is as follows: Krum assumes that party updates have a normal distribution and that the benign updates lie close to each other in the parameter space. Hence, instead of computing the average of the updates, Krum selects as aggregate the update that is closest to its $(1-\epsilon)n$ neighbor updates. The details of the aggregation are as follows.

Let $\theta_1, ..., \theta_n$ be the updates received by the server. For $i \neq j$, $i \to j$ denotes that θ_j belongs to the $(1-\epsilon)n - 2$ updates closest to θ_i. Let $s(\theta_i) = \sum_{i\to j} ||\theta_i - \theta_j||^2$ be the *score* of θ_i. Then, Krum selects the θ_k with the lowest score. The Krum aggregation algorithm is formalized in (2).

$$f_{\mathsf{Krum}}: \quad \theta_a^{t+1} = \underset{\theta_{i\in[n]}^t}{\operatorname{argmin}} \sum_{i\to j} ||\theta_i^t - \theta_j^t||^2 \tag{2}$$

C. Bulyan

The breaking point of Krum is $\epsilon = (\frac{n-2}{2n})$, i.e., it can tolerate up to $(\frac{n-2}{2})$ malicious parties, while maintaining high utility of the final model [13]. However, El Mhamdi et al. [37] proposed an attack on Krum assuming an omniscient adversary who has access to all the benign updates. The attack exploits the fact that, in a vector space of dimension $d \gg 1$, small disagreements on each coordinate translate into a distance $\|\mathbf{x}-\mathbf{y}\|_p = \mathcal{O}(\sqrt[p]{d})$. Therefore, the adversary crafts a malicious update with a single dimension set to a large value, and the other dimensions set to the average of the benign updates. Such malicious update pushes the parameter vector to a sub–optimal parameter space and destroys the global model's accuracy.

Essentially, Krum filters outliers based on the entire update vector, but does not filter coordinate-wise outliers. To address this, [37] proposes a meta-aggregation rule Bulyan, which performs vector-wise, e.g. Krum, and coordinate-wise, e.g. TrimmedMean [57], filtering in two steps. At first, Bulyan uses some Byzantine resilient aggregation $\mathcal{A}$, e.g., Krum in Algorithm 2, to filter outliers based on the distances between the update vectors, and then aggregates these updates using a variant of TrimmedMean. Algorithm 2 describes the Bulyan aggregation.

Algorithm 2 Bulyan aggregation: f_{Bulyan} [37]

Input: $\mathcal{A} = f_{\mathsf{Krum}}$, $\mathcal{P} = (\theta_1^t, ..., \theta_n^t)$, n, ϵ
$S \leftarrow \emptyset$
while $|S| < (1-2\epsilon)n$ **do**
  $p \leftarrow \mathcal{A}(\mathcal{P} \backslash S)$
  $S \leftarrow S \cup \{p\}$
end while
Output: $\theta_a^{t+1} = \mathsf{TrimmedMean}(S)$

Among the different variants of TrimmedMean [55], [37], [57], we follow the one used in the original work [37] given in (3). Here, U_j is defined as the set of indices of the top-$(1-2\epsilon)n$ values in $(\theta_1^j, ..., \theta_n^j)$ nearest to their *median* μ_j.

$$\mathsf{TrimmedMean}(\theta_1, ..., \theta_N) = \left\{ \theta_a^j = \frac{1}{|U_j|} \sum_{i \in U^j} \theta_i^j \ \forall j \in [d] \right\} \tag{3}$$

D. Multiplicative weight update (MWU)

Multiplicative weight update (MWU) technique lies at the core of many learning algorithms [8], [19], [45], [20], [34]. The general framework of MWU is given in Algorithm 3. The intuition behind MWU-based aggregations is to reduce the weights of malicious parties using the distance between their malicious updates and the aggregated update. This is based on two assumptions: malicious updates lie farther away from the mean compared with the benign updates, and the number of malicious updates is smaller than that of the benign updates. Therefore, MWU-based aggregation schemes have the breaking point of $\lfloor \frac{n-1}{2} \rfloor$ malicious parties.

Algorithm 3 Multiplicative weights update: f_{MWU}

Input: $\mathcal{P} = (\theta_1^t, ..., \theta_n^t)$
Initialize parties' weight vector $\mathbf{w}^0 \leftarrow \mathbf{1}$ and $\theta_a^0 \leftarrow f_{\mathsf{AGG}}(\mathbf{w}^0, \mathcal{P})$ at $t = 0$
repeat
  $\mathbf{w}^{t+1} \leftarrow$ WeightUpdate $(\mathbf{w}^t, \theta_a^t, \mathcal{P})$
  $\theta_a^{t+1} \leftarrow f_{\mathsf{AGG}}(\mathbf{w}^{t+1}, \mathcal{P})$
  $t \leftarrow t+1$
until Convergence criterion is satisfied
Output: final θ_a

There are different variants of MWU that use different functions for WeightUpdate and f_{AGG} in Algorithm 3. We detail two of them next.

1) MWU *with mean aggregation:* In MWU, if the weighted mean (Section III-A) is used as the aggregation algorithm, f_{AGG} in Algorithm 3, it is called MwuAvg. Here, the weight of the i^{th} party is updated based on the distance between the weighted average, θ_a^t, of all the updates and θ_i; this is given by (4). The weights of all the parties are equal at the beginning.

$$w_i^{t+1} = w_i^t \ \exp(-||\theta_a^t - \theta_i||_p) \tag{4}$$

$$\theta_a^{t+1} = \frac{\sum_{i=1}^n w_i^{t+1}\theta_i}{\sum_{i=1}^n w_i^{t+1}} \tag{5}$$

2) MWU *with optimization:* Li et al. [34] propose a truth discovery framework CRH, to aggregate the responses in a crowd-sourcing setting. In each epoch, the framework updates the weights of parties based on the solution of an optimization problem. Essentially, the weight update algorithm considers the distance of parties' updates from the aggregate of all the updates in each epoch. The weight update and aggregate computation are given by (6) and (7), respectively. For further details of the aggregation, please refer to [34].

$$w_i^{t+1} = -\log\left(\frac{||\theta^t - \theta_i||_p}{\sum_{i=1}^n ||\theta^t - \theta_i||_p}\right) \tag{6}$$

$$\theta^{t+1} = \sum_{i=1}^n w_i^{t+1}\theta_i \tag{7}$$

IV. Attacks on Aggregation Algorithms

In this section, we detail the poisoning and membership inference attacks used in our work to evaluate the robustness and privacy in federated learning. The poisoning attacks are of two types: *availability* and *targeted* attacks. The earlier attacks aim to jeopardize the overall accuracy of the final model/s [11], [23], [50], while the latter attacks aim to mis-classify only a specific set of samples of the attacker's choice at the test time [10], [12]. We focus on the poisoning availability attacks and below introduce such attacks from the literature, and also introduce a new poisoning attack targeting the MwuAvg and MwuOpt aggregations.

A. Label flip poisoning (Label flip)

We consider a type of data poisoning attacks [26], [38], [23], where the adversary flips the labels of her local training data in a particular fashion to poison it. We call this attack *label flip poisoning* attack. The label flipping strategy is performed consistently across all of the ϵn malicious parties that the adversary controls, i.e., all the malicious parties flip labels in the exact same way. Then, the ϵn malicious parties use this poisoned data to train their local models and then share corresponding updates with the central server.

B. Naive poisoning (PAF)

Our threat model from Section II considers an omniscient adversary who knows the distribution of benign updates, i.e., the mean and standard deviation of each dimension of benign updates. The adversary can estimate this distribution as she possesses data drawn from the distribution same as that of benign parties. Given this, the adversary crafts the malicious update θ_m to be arbitrarily far from the mean of the benign updates:

$$\theta_m = \frac{\sum_{i=1}^{n} \theta_i}{(1-\epsilon)n} + \theta' \tag{8}$$

This malicious update, θ_m, is then shared by each malicious party with the central server. In (8), θ' is a vector of size $|\theta_i|$ with arbitrarily large coordinate values. This attack can jeopardize the weighted averaging, and interestingly, weighted median aggregations based federated learning (Section III-A).

C. Little is enough attack (LIE)

Baruch et al. [11] propose an attack called *little is enough* (LIE). The attack successfully circumvents state-of-the-art robust aggregation algorithms, including Bulyan and Krum. These aggregations are vulnerable to the attack, because they are tailored to an adversary that crafts a malicious update with at least one arbitrarily large dimension. However, in practice, a malicious update, θ^m, obtained by small perturbations in a large number of dimensions of a benign update suffice to affect model's convergence and also circumvent the defenses. Therefore, note that, *the root cause of the success of the LIE attack is also the high dimensionality of the updates.* The attack is described in Algorithm. 4. El Mhamdi et al. [37] propose an attack against geometric median based aggregations such as Krum, but does not work against Bulyan, and therefore is omitted from the evaluation.

Algorithm 4 Little is enough attack (LIE) [11]

1: **Input**: n, ϵ, mean and variance vectors of benign updates $\bar{\mu}$, $\bar{\sigma}$
2: Number of workers required for majority:
$$s = \lfloor \frac{n}{2} + 1 \rfloor - \epsilon n$$
3: Using z-table, set $z = \max_z \left(\phi(z) < \frac{n-s}{n} \right)$
4: **for** $j \in [d]$ **do**
5: $\quad \theta_m^j \leftarrow \bar{\mu}_j + z\bar{\sigma}_j$
6: **end for**
7: **Output**: malicious update θ_m

D. Our poisoning attack (OFOM)

In this section, we propose an attack which targets aggregation schemes that perform weighted aggregation of data by assigning the weights based on the distance of the data points from an aggregate of the data. These aggregations are robust to all the above attacks, as we show in our evaluation. We discussed two such aggregation schemes: MWU with averaging [8] and MWU with optimization [34] in Section III. In any given epoch, the aforementioned aggregation schemes start with equal weights to all parties and update a party's weight based on the distance of the party's update from weighted average of all party updates. *The attack exploits the fact that, all parties are given equal weights to start with.* The OFOM attack craft two malicious updates: The first update, θ_1^m, is arbitrarily far away from the true mean, and is obtained by adding an arbitrarily large vector θ' to the mean of benign updates. The second malicious update, θ_2^m, is at the empirical mean of benign updates and θ_1^m. The malicious updates are formalized in (9).

$$\theta_1^m = \frac{\sum_{i=1}^{n} \theta_i}{n} + \theta', \quad \theta_2^m = \frac{\sum_{i=1}^{n} \theta_i + \theta_1^m}{n+1} \tag{9}$$

This way, at the end of the first epoch of the MWU aggregation, the adversary manages to assign a weight close to 1 to the parties with update θ_2^m. In the case of MWUAvg and MWUOpt, all the benign parties are assigned negligible weights, which completely jeopardizes the accuracy of aggregation. To be effective, the adversary needs just two malicious parties who share the two malicious updates.

E. Membership inference attacks

Recent research has shown the susceptibility of the federated learning to active and passive inference attacks [36], [40]. In the passive case, the attacker, either the server or some of the parties, simply observes the updated model parameters to mount membership inference attacks. In the active case, however, the attacker tampers with the training of the victim model/s in order to infer membership of target data in *any of the benign party's data*. Specifically, the attacker shares malicious updates and forces the victim model/s to share more information about the members of their training data that are of the attacker's interest. This attack, called *gradient ascent* attack [40], exploits that the SGD optimization updates model parameters in the opposite direction of the gradient of the loss over the private training data. Let $\mathbf{x}$ be a record of attacker's interest and θ_a be the current global model. The attacker *crafts the malicious update θ^m by updating parameters of θ_a in the same direction of the gradient of the loss on* $\mathbf{x}$, i.e., performs gradient ascent as: $\theta^m = \theta_a + \gamma \frac{\partial L_{\mathbf{x}}}{\partial \theta_a}$. Such θ^m, when combined with the benign updates, increases the loss of the resulting global model, θ'_a, on $\mathbf{x}$. If $\mathbf{x}$ is in the training data of some party, SGD on θ'_a by this party will sharply reduce the loss of $\mathbf{x}$. On the other hand, if $\mathbf{x}$ is not in any party's training data,

the loss of $\mathbf{x}$ will remain almost unchanged. Therefore, this attack increases the gap between the losses of θ_a on members and non-members and facilitates membership inference.

V. Cronus

In the existing federated learning algorithms, described in Section III-A, the server repeatedly collects the parameters of the local models, aggregates them by computing their mean, and sends the aggregate parameter vector back to the parties. This is the simplest way of sharing knowledge between the participants, and has the following fundamental drawbacks:

Robustness: As shown in Table I, the upper bound on the error of aggregation algorithms (in the adversarial setting) depends on the dimensionality of the model parameters. This makes the aggregated models to be highly error-prone, for large models. Thus, they are very susceptible to poisoning attacks, as described in Section IV.

Privacy: Sharing the model parameters facilitates strong white-box inference attacks, as it opens up the model to the adversary [36], [40]. The larger the models are, the more significant their leakage about their local training data is.

Heterogeneity: Parameter aggregation is restricted to homogeneous architectures, i.e., all parties need to have the same model architecture.

Instead of sharing the raw models at each round of training, the participants would need to share the knowledge that they have learned from their training data, in a succinct way. This is the main objective of *knowledge transfer* [21], [9], [43], [44], [54], [7], which is a powerful tool to share knowledge of a model through its predictions.

Knowledge transfer efficiently transfers the represented function by a model (or an ensemble of models) to a student model [21]. It makes learning very effective for the student model by placing equal weight on the relationships learned by the teachers across all the classes and significantly improves the convergence of student model (as compared to training directly on hard-labeled data) [21], [9], [7]. Furthermore, knowledge transfer is an effective regularization method [21].

What makes this approach, in particular, suitable for **robust, privacy-preserving, and heterogeneous** federated learning is three-fold. First, it significantly reduces the dimensions of the updates, from the size of the model parameters, to its output size. This enables using robust aggregation algorithms with very tight robustness guarantee. Second, it limits the interactions with each local model to the black-box access setting. Thus, the negative impact of poisoning attacks *onto* the model, and the information leakage *from* its training data, are both limited. The knowledge transfer through the predictions on a dataset, that does not overlap with the model's training set, itself reduces the information leakage of a model about its training data [48]. In addition, the black-box setting allows us to make use of utility-preserving algorithms to make the knowledge transfer privacy-preserving, as a party only needs to mitigate the privacy risk of sharing its predictions [39], [43], [18]. Third, all the models agree on a particular learning task, so are homogeneous on their output vectors. Yet, they can be of heterogeneous architectures, or even different families of machine learning algorithms.

Table II: Sample complexity, $\Theta((d/\epsilon)\log d)$, of Cronus aggregation, using [17], for parameters and predictions updates. For parameters, d is size of model; for predictions, d is the number of classes in the classification task. The ratio shows that Cronus learning can achieve the same error guarantee as in federated learning, but with a network which is 5 orders of magnitude smaller, for benchmark ML tasks.

Dataset	Sample complexity ratio of Federated learning over Cronus
SVHN	1.2×10^5
MNIST	3.3×10^5
Purchase	2.75×10^5
CIFAR10	10.4×10^5

In this paper, we leverage knowledge transfer and propose the ***Cronus* federated learning** algorithm, where the aggregation algorithm devours the knowledge of local models, and disgorges their robust aggregation. The parties update their models using the aggregated knowledge as well as their local data. Then, each party shares their knowledge through (privacy-preserving) predictions of their local models on a public dataset. This repeats in every round of the federated learning.

A. Cronus Collaborative Learning

In addition to the local private datasets of parties, Cronus assumes a set of *unlabeled* public data, X_p. Such public data is essentially a set of feature vectors that Cronus uses to extract and exchange the knowledge of local models.

Algorithm 5 describes the Cronus collaborative learning algorithm. Cronus has two training phases: In the first phase, called the *initialization phase*, every party i updates its local model parameters θ_i on its local training data D_i for T_1 times without any collaboration. In the second phase, called the *collaboration phase*, the parties share the knowledge of their local models via their predictions on the public dataset, X_p. Specifically, each epoch of this phase includes:

- Each party computes soft labels for X_p, using its local model parameters θ_i to get prediction vectors Y_i and shares them with the server.
- The server aggregates the predictions (separately for each public data), i.e., computes $\bar{Y} = f_{\mathsf{Cronus}}(Y_i, .., Y_n)$, and sends $\bar{Y}$ to all parties; f_{Cronus} is an aggregation algorithm, which we describe in Section V-B.
- Each party updates its local model parameters θ_i using their private data $\mathbf{D}_i$ and the soft-labeled public data $(X_p, \bar{Y})$.

Sharing predictions instead of parameters completely eliminates the risk of white-box inference attacks, as no party releases its model's parameters. While training on their local sensitive data, parties can anticipate the privacy risks of information leakage through their predictions and TRAIN their models with, for example, a membership privacy mechanism [39]. They can also make use of prediction privacy mechanisms before sharing their predictions with the server [18].

It's important to note that, unlike the existing federated learning algorithms, based on FedAvg [35], *Cronus does*

Algorithm 5 Cronus algorithm. Initialization phase does not involve collaboration. D_i and θ_i are local dataset and model paramters from i-th party. Y_i^t are predictions from i-th party on public dataset X_p in epoch t and $Y_i^t[k]$ is the prediction on k-th public data in D_p.

Initialization phase

Each party $i \in [n]$ updates parameters in parallel:
for $t \in [T_1]$ epochs **do** ▷ Training *without* collaboration
  Update $\theta_i \leftarrow \textsf{TRAIN}\ (\theta_i, D_i)$
end for
$Y_i^0 = \textsf{PREDICT}(\theta_i; X_p)$ ▷ Compute initial predictions on X_p
Send Y_i^0 to the server

Collaboration phase

$\bar{Y}^0 = f_{\textsf{Cronus}}(\{Y_{i\in[n]}^0\})$ ▷ Initial aggregation at the server
for $t \in [T_2]$ epochs **do**
  for $i \in [n]$ parties **do** ▷ Each party updates parameter in parallel
    $D_p = \{X_p, \bar{Y}^t\}$
    $\theta_i \leftarrow \textsf{TRAIN}\ (\theta_i, D_i \cup D_p)$ ▷ Update local model parameter θ_i
    $Y_i^t = \textsf{PREDICT}(\theta_i; X_p)$
    Send Y_i^t to the server
  end for
  $\bar{Y}^{t+1} = f_{\textsf{Cronus}}(\{Y_{i\in[n]}^t\})$ ▷ Aggregation at the server
end for

基于local data训练local model然后再计算softmax of public data

client再基于public data和local data更新本地模型，上传public data的softmax，server继续整合softmax

not force a single global model onto local models. Instead, each local model is updated separately by improving their classification accuracy and resilience to inference attacks. This, can further improve the robustness of local models against active attacks (such as poisoning and active inference attacks), as they do not blindly overwrite their local parameters with the aggregated knowledge.

The size of the updates and the size of the network determine the error bound for robust mean estimation algorithms. By reducing the dimensionality of the updates, Cronus significantly reduces the guaranteed error, using any robust aggregation algorithm. Then, the sample complexity of the aggregation algorithm determines how many participants are needed to achieve a tight error bound. Cronus uses the aggregation algorithm proposed by Diakonikolas et al. [17], whose sample complexity is $\Theta(d \log d)$, and has the least dependence on the dimensionality of updates among existing robust mean estimation algorithms. Table II shows the sample complexities for training different ML benchmark models with federated learning versus Cronus, using Diakonikolas's algorithm. We note that Cronus significantly reduces sample complexity (by an order of 10^5), and therefore, unlike any existing federated learning, can achieve strong theoretical error guarantees even with a small number of parties in the network.

The computational complexity of the state-of-the-art robust aggregation algorithm is $O(d^3 + n)$, where d is the dimensionality of the updates [17]. This makes it impractical to use such optimal algorithms for the high dimensional parameter updates. However, the complexity cost is negligible on Cronus, due to the small number of dimensions on its updates.

Algorithm 6 Cronus aggregation: $f_{\textsf{Cronus}}$ [Algorithm 3 [17]]

1: **Input**: $S = (Y_1, ..., Y_n)$, ϵ, k=0
2: **while** $k < |X_p|$ **do** ▷ Compute prediction for each public data
3: $S_k = (Y_1[k], ..., Y_n[k])$
4: **while** True **do**: ▷ Robust mean aggregation algorithm
5: Compute $\bar{Y}_k$,Σ_k, the mean and covariance matrix of S_k.
6: Find the eigenvector v^* with highest eigenvalue λ^* of Σ_k
7: **if** $\lambda^* \leq 9$ **then**
8: Let $\bar{Y}[k] = \bar{Y}_k$ and k = k+1
9: break
10: **else**
11: Draw Z from the distribution on [0,1] with probability density function $2x$
12: Let T = $Z \max\{|v^* \cdot (Y - \bar{Y}_k)| : Y \in S_k\}$.
13: Set $S_k = \{Y \in S_k : |v^* \cdot (Y - \bar{Y}_k)| < T\}$
14: **end if**
15: **end while**
16: **Output:** $\bar{Y} = \{\bar{Y}[0], \bar{Y}[1], ..., \bar{Y}[k]\}$
17: **end while**

n个client对public data k的softmax

直接采用being robust (in high dimensions) can be practical的算法
robust mean estimation，但是这里demension不大吧，套的太生硬了

B. Robust Mean Estimation in Cronus

We use the robust mean estimation algorithm 6 proposed by Diakonikolas et al. [17] in Cronus. Algorithm 6 achieves dimension independent error guarantee and nearly optimal sample complexity as shown in the Theorem 1.

The intuition behind Algorithm 6 (step 4-15) is that when the empirical mean of the data is corrupted, then along the corrupted direction, the empirical variance is much larger than the population variance. The algorithm finds the direction v^*, which has the largest variance and projects the deviation of all the inputs from the empirical mean in this direction. Then filter out a randomized fraction of the data which are farthest from the mean, $\bar{Y}_k$, along this direction. Repeat the process until the variance is not large in every direction and then output the sample mean on the subsets. We apply the robust mean estimation for each data in the public dataset X_p separately.

Theorem 1. *[Theorem A.16 [17]] Let P be a distribution on $\mathbb{R}^d$ with an unknown mean vector μ^p and an unknown covariance matrix $\Sigma_p \preceq \sigma^2 I$. Let S be an ϵ-corrupted set of samples from P of size $\Theta((d/\epsilon)\log d)$. Then, there exists an efficient algorithm that, on input S and $\epsilon > 0$, with probability 9/10 outputs $\hat{\mu}$ with $||\hat{\mu} - \mu^P||_2 = O(\sigma\sqrt{\epsilon})$.*

For the theoretical analysis, Algorithm 6 uses randomized filtering (step 12) and repeats until the stop condition is satisfied (step 7). The follow-up works from the same authors [33], [15] suggest a simpler algorithm to obtain a better performance in practice: (1) in each iteration, remove a deterministic fraction of the data instead of a random fraction. (2) repeat the filter for constant iterations in total. In the evaluation, we filter out $\epsilon/2$ fraction of the inputs in each iteration (step 12) and repeat the filter process 2 times (step 7) and to obtain a good performance.

Table III: Experimental setup

Dataset	# of benign parties	Model architecture	Private data per party	Public data
SVHN [41]	32	CNN	5,000	10,000
MNIST [32]	28	FC	2,000	10,000
Purchase [48]	16	FC	10,000	10,000
CIFAR10 [30]	16	DenseNet	2,500	10,000

Table IV: Comparison of the classification accuracy of Cronus models (on average) with that of the models trained in stand-alone (on average), centralize, and FedAvg settings, in the **benign** setting.

Dataset	Stand-alone	Centralized	FedAvg	Cronus
SVHN	87.5	96.4	95.9	91.1
MNIST	92.8	97.9	96.7	95.2
Purchase	76.3	94.3	93.3	89.6
CIFAR10	66.8	90.2	88.4	80.1

VI. Experimental Setup

Table III presents the datasets and corresponding model architectures, number of parties, and size of the data sets that we use in our experiments. We defer the reader to Appendix A, for the descriptions of the datasets, and details of the model architectures, and training hyper-parameters. We use PyTorch [2] for our evaluations. We evaluate the security of federated learning algorithms with respect to the following measures.

Robustness. We measure the robustness of an algorithm as the ratio between the accuracy of the model against the strongest poisoning attack, over the accuracy of the model in the benign setting. For finding the strongest attack, we evaluate an algorithm against all the attacks which are presented in Section IV.

Membership inference risk. We measure the membership inference risk as the accuracy of the inference attack, which is the percentage of data records for which the attack model correctly predicts their membership [48]. We test the attack with the same number of members and non-members. Hence, 50% accuracy corresponds to a random guess.

VII. Empirical results

In this section, we first present the performance of the models in the benign setting, as the baseline. Then, we compare Cronus with other algorithms, with respect to their robustness, privacy, heterogeneity, and communication efficiency.

Table IV shows the classification accuracy of the models trained using Cronus, versus stand-alone, centralized, and FedAvg federated learning, all in the benign setting. In the stand-alone setting, each party trains its only on its local data, without any collaboration. In the centralized learning, a single model is trained on the union of the participants' data. FedAvg federated learning is described in Section II. For stand-alone setting, and also for Cronus, the accuracy of models for different parties is different. In these two cases, we report the average classification accuracy of the models.

Table V: The number of malicious parties used for robustness assessment of different aggregations, based on their breaking points. The number of benign parties are shown on top of each column.

f_{AGG}	Breaking point	Number of malicious parties		
		SVHN	MNIST	Purchase/CIFAR10
		32 benign	28 benign	16 benign
Mean	$1/n$	1	1	1
Median	$1/2$	31	27	15
MWU	$1/2$	31	27	15
Krum	$(n-2)/2n$	29	25	13
Bulyan	$(n-3)/4n$	9	8	4
Cronus	$1/2$	31	27	15

A. Robustness

We compare the robustness of federated learning, using different aggregation algorithms (Section III), against an adversary who mounts the strong poisoning attacks (described in Section IV). To assess the worst case robustness, we evaluate algorithms against the largest fraction of malicious parties which is smaller than the breaking point. The number of benign and malicious parties are given in Table V.

The robustness assessment results are shown in Table VI. Each column corresponds to one aggregation algorithm, and the *Worst accuracy* row shows the accuracy of the final models when the attack in the *Strongest attack* row is mounted.

Figure 1 shows the convergence of Cronus and federated learning. We show the convergence plots of the algorithms in the benign setting, as a baseline for comparison, in Figure 4 in Appendix B. The federated learning algorithm works well in the absence of malicious parties, however, **all of the existing aggregation schemes in federated learning are significantly vulnerable to at least one poisoning attack.**

The weighted average aggregation (Section III-A), i.e., FedAvg [35], is susceptible to all of the attacks: *The accuracy of FedAvg reduces close to random guess accuracy for all the datasets*. Median aggregation is susceptible to PAF attack because the attack shifts the final aggregate along all the dimensions by a small amount to remain undetected, yet it can considerably damage the utility of the aggregated model. Although Bulyan and Krum are robust aggregations, they are susceptible to the LIE attack. As explained in Section IV-C, LIE attack exploits the sensitivity of the parameters of neural networks to small perturbations. The attack completely jeopardizes the accuracy of Krum aggregation, because the attack successfully forces Krum aggregation to select the malicious update as the aggregate in most of the epochs. Note that, the attack is not effective against MNIST classification task due to its simplicity, which allows the corresponding models to withstand the small perturbations. MwuAvg and MwuOpt withstand all the attacks, but are susceptible to our OFOM attack, which we propose in Section IV-D. For all the datasets, the OFOM attack reduces the accuracy of the aggregations close to the random guess accuracy.

On the other hand, **the robustness of Cronus remains almost 1.0 as the classification accuracy of its models**

Table VI: Robustness of the federated learning using different aggregation algorithms versus Cronus. Robustness is measured as the ratio of the accuracy of the final models against the strongest attack (Section IV) over the models' accuracy in the benign setting. The number of benign and adversarial parties are provided in Table V. The best accuracy for each setting (row) is highlighted.

Dataset		Federated learning with various aggregation algorithms						Cronus
		Mean	Median	MwuAvg	MwuOpt	Bulyan	Krum	
SVHN	Accuracy (Benign)	**95.9**	94.8	93.9	94.4	94.5	89.6	91.1
	Worst accuracy (Adversarial)	0.9	14.5	0.9	0.7	15.5	16.2	**89.8**
	Strongest attack	(OFOM)	(LIE)	(OFOM)	(OFOM)	(LIE)	(LIE)	(Label flip)
	Robustness	0.01	0.15	0.01	0.01	0.16	0.18	**0.99**
MNIST	Accuracy (Benign)	96.7	96.5	**97.2**	97.4	96.9	93.3	95.2
	Worst accuracy (Adversarial)	9.6	91.5	25.3	12.7	**94.1**	89.9	93.7
	Strongest attack	(PAF)	(PAF)	(OFOM)	(PAF)	(LIE)	(Label flip)	(Label flip)
	Robustness	0.09	0.95	0.26	0.13	0.97	0.96	**0.99**
Purchase	Accuracy (Benign)	93.3	93.0	**93.6**	92.5	92.8	72.1	89.6
	Worst accuracy (Adversarial)	1.1	12.5	1.8	1.1	81.8	49.6	**88.0**
	Strongest attack	(PAF)	(PAF)	(OFOM)	(OFOM)	(LIE)	(LIE)	(Label flip)
	Robustness	0.01	0.75	0.02	0.01	0.87	0.69	**0.98**
CIFAR10	Accuracy (Benign)	88.4	**89.1**	86.2	87.6	89.0	84.5	80.1
	Worst accuracy (Adversarial)	11.3	15.1	14.2	12.8	75.6	18.0	**78.0**
	Strongest attack	(PAF)	(PAF)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LIE)
	Robustness	0.13	0.17	0.16	0.15	0.85	0.21	**0.97**

remains unaffected by any of the tested poisoning attacks. For the strongest attack on Cronus, the maximum reduction in accuracy is 0.4% for Purchase, 1.3% for SVHN, 1.5% for MNIST, and 4.8% for CIFAR10 models. The reason for the high resilience of Cronus to the poisoning attacks is three-fold: First, as detailed in Section V-A, reduced dimensionality of updates reduces the aggregate error and the sample complexities required to achieve the error. Second, in Cronus, the models are not overwritten by aggregate models. We instead take advantage of knowledge transfer algorithms [21]. The parameters of a model are sensitive to small perturbations which can prevent the model from converging [11]. Therefore, in the existing federated algorithms, the success of the poisoning attacks stems from the fact that a little noise introduced via the malicious parameter updates suffices to both prevent the model from convergence and to bypass the filtering of robust aggregation algorithms. Third, benign models have a stronger agreement on their predictions than on their parameter values. Thus, in Cronus, it is much easier to detect (or cancel the effect of) the poisoning vectors.

B. Privacy

We evaluate the risk of membership inference attacks on the participants' private training data during collaboration, and the effect of privacy preserving mechanisms [4], [39]. As described in the threat model in Section II-B, we assume the central server and other participants to run passive and active membership inference attacks [40] on individual party updates and their aggregates. We use Purchase, SVHN, and CIFAR10 datasets for our evaluation when 4 parties collaborate.

1) Passive membership inference attacks: In the case of passive membership inference attacks, the server isolates the parties and mounts the attack separately on each of the collected updates, i.e., in case of FedAvg, attack is mounted on the parameter updates of each party and in case of Cronus, attack is mounted on the model obtained by training on the predictions shared by each party. We also evaluate the privacy risk when the attack is mounted on the aggregate of these updates.

The results are shown in Table VII. **The updates in FedAvg are highly susceptible to membership inference unlike the updates in Cronus.** For the Purchase dataset, attack accuracy against the individual and aggregated updates in FedAvg is 78.1% and 80.1%, respectively, whereas in Cronus they are 51.7% and 51.9%. Unlike the prediction updates in Cronus, the high dimensional parameter updates in FedAvg encode a significantly higher amount of information about the party's local data. Furthermore, knowledge transfer acts as a strong regularization method and mitigates the risk of membership inference attacks [21], [49]. It's important to note that knowledge transfer through predictions, on a dataset other than the training data, makes the behavior of the student model more indistinguishable on its training versus unseen data. This happens as the distillation process does not carry the exceptionally distinguishable characteristics of the model on its training data, and results in smooth decision boundaries of the student model around the teacher's training data. We observe similar results for SVHN and CIFAR10 datasets.

We also evaluate Cronus and FedAvg, when models use adversarial regularization during local training to improve membership privacy. We note that, **adversarial regularization improves membership privacy of both FedAvg and Cronus**, but the increase is smaller for Cronus due to its inherent resilience to membership inference. For Purchase dataset with FedAvg, the risk to individual and aggregated updates reduces by 9.9% and 12.7%, respectively, while for Purchase with

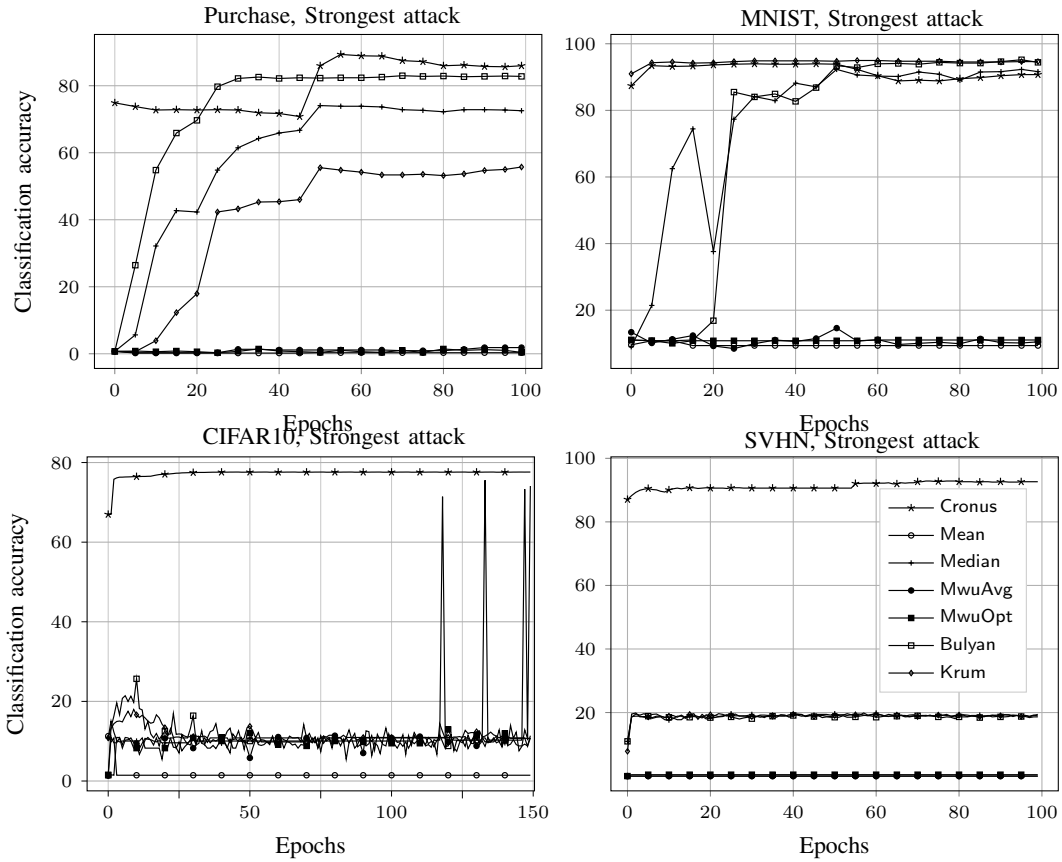

Figure 1: Convergence of Cronus and existing federated learning algorithms in **adversarial setting**. Accuracy of Cronus in adversarial setting is almost the same as in benign setting (shown in Figure 4) due to its high robustness. Except for CIFAR10, for which only collaboration phase is shown, both the Cronus training phases are shown in figure and the collaboration phase starts at epoch 50.

Cronus, the risk to individual and aggregated updates is already very small, and it further reduces by 0.6% and 1.1%, respectively. Similarly for CIFAR10, privacy improvement in FedAvg is significantly more than in Cronus. However, the privacy improvement for SVHN is very small even in case of FedAvg, due to large gaps in train and test accuracies at stronger adversarial regularization.

2) Active membership inference attacks: In each epoch, the server manipulates the aggregate update that it broadcasts to the parties by performing gradient ascent on the *aggregated update* for a set of target data [40]. In FedAvg, gradient ascent is performed directly on the aggregated parameters. In Cronus, for running such attack, the server needs to train a model on the aggregated predictions while performing gradient ascent on the target data, and then, shares predictions of this model with the parties; we ensure that such model has accuracy close to the accuracy of party models in given epoch.

Table VII shows the results. **The active attacks significantly increase the privacy risk of the target data in case of FedAvg**: for Purchase dataset, the risk due to individual update increases by 7.8% (77.1% to 84.9%), while due to aggregated update increases by 8% (74.7% to 82.7%). But, **in case of Cronus, the active attacks are ineffective and the increase in risk is negligible**: 0.3% for individual update while 1.1% for aggregated update. In Figure 2, we show the effect of gradient ascent on the difference in gradient-norms of target and non-target data for aggregated model for the Purchase dataset. This directly correlates with the success of membership inference [40]. We observe that, for FedAvg on SVHN dataset, the active attacks increase the risk to individual and aggregate updates by 2.5% and 4.4%, respectively, but, the increased risk negligible for Cronus.

C. Differential privacy

We compare the performance of Cronus and the conventional federated learning with user-level [14] or record-level [4] differential privacy. For both of these comparisons, we use SVHN dataset with 32 benign parties and the model architecture as described in Table III, and use the moments accountant method (and the code) [4], [3] to bound the total privacy risk. Note that we train the *whole model* using differential privacy, as opposed to only training the last layer [4]. Our results shows the significant accuracy cost of some existing DP algorithms for federated learning, and calls for designing DP algorithms with high utility and tight privacy loss.

1) User-level DP [14]: The user-level DP (UDP-FedAvg) algorithm proposed by McMahan et al. [14] cannot lead to training good accuracy models, when the number of parties in each epoch is small. Even for large privacy budgets, i.e., $\epsilon = 100$, the parties do not benefit from collaboration, and with the user-level DP, the global model in FedAvg achieves

Table VII: Accuracy of passive and active membership inference attacks with central server as adversary. We also evaluate effect of adversarial regularization (with parameter λ) used to preserve membership privacy. We use 4 parties and data per party as in Table III.

Dataset	Federated learning algorithm	Attack on party update		Attack on aggregated update	
		Passive attack acc.	Active attack acc.	Passive attack acc.	Active attack acc.
Purchase (without privacy)	FedAvg	77.1	84.9	74.7	82.7
	Cronus	54.6	54.9	53.6	54.7
Purchase (with membership privacy, $\lambda = 3$)	FedAvg	70.8	77.3	69.9	77.0
	Cronus	54.1	51.5	53.7	54.6
SVHN (without privacy)	FedAvg	64.8	67.3	59.9	64.3
	Cronus	55.6	53.1	56.5	55.7
SVHN (with membership privacy, $\lambda = 0.5$)	FedAvg	64.9	67.0	60.0	64.2
	Cronus	54.1	56.9	55.6	55.0
CIFAR10 (without privacy)	FedAvg	79.9	80.5	76.8	77.1
	Cronus	57.0	57.8	55.5	56.7
CVIFAR10 (with membership privacy, $\lambda = 1$)	FedAvg	59.9	64.1	59.6	62.2
	Cronus	52.9	54.4	52.6	57.0

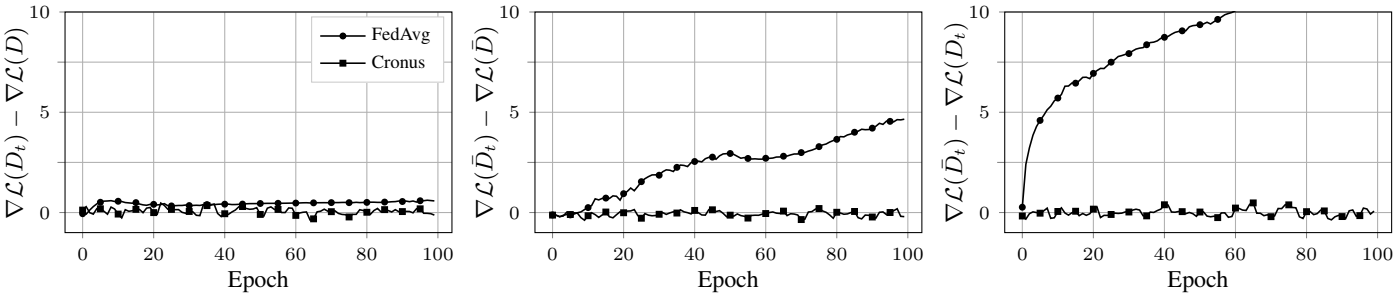

Figure 2: Difference in the gradient-norms, $\nabla\mathcal{L}(D)$, of the last layer of aggregated model on the target and non-target data (Purchase100 data). In the context of active membership inference attacks, D, D_t, $\bar{D}$, and $\bar{D}_t$ denote non-target members, target members, non-target non-members, and target non-members, respectively.

close to random-guess accuracy. We observed similar results for running user-level DP on Cronus with small number of participants. This result is expected as the sensitivity of the element-wise mean aggregation algorithm is inversely proportional to the number of parties, and a very large number of parties is required to reduce the noise, e.g., [14] uses 5000 parties in each epoch.

2) Record-level DP [4]: We empirically show that the conventional parameter aggregation in FedAvg is not suitable to provide the record-level DP, and is also susceptible to poisoning attacks.

The results are shown in Table VIII for SVHN dataset. The accuracy of the models for federated learning or Cronus, with DP-SGD, cannot reach the accuracy of stand-alone training, which makes the collaboration useless. The results of the strongest poisoning attacks show that DP-SGD FedAvg has no robustness against the attacks.

D. Cronus with heterogeneous model architectures

Due to the use of predictions based updates, Cronus allows parties with heterogeneous model architectures to participate in collaboration. Below, we compare different aspects of the homogeneous and heterogeneous collaborations. We use Purchase data and 5 fully connected models, which we call

Table VIII: Accuracy and robustness of models for record level DP [4] with $\epsilon = 15.4$ on the SVHN dataset. The baseline stand-alone accuracy is 87%.

# of parties	Accuracy (Benign)		Worst accuracy		Robustness	
	FedAvg	Cronus	FedAvg	Cronus	FedAvg	Cronus
32	65.7	85.8	4.5	83.1	0.07	**0.97**
16	74.3	84.8	8.1	84.0	0.11	**0.99**
8	77.9	84.2	0.9	82.2	0.01	**0.98**
4	81.6	81.5	1.7	81.2	0.02	**0.98**

A1, A2, A3, A4, and A5, with hidden layer sizes {}, {1024}, {512, 256}, {1024, 256}, and {1024, 512, 256} respectively. Note that, A1 models, called *bad models*, have lower capacity and accuracy than A2-5 models, which we call *good models*. We denote by $P_{j:k}$ the model architectures of parties $\in [j, k]$. We denote the entire collaboration in curly brackets, e.g., we denote the collaboration of 3 sets of 4 models, i.e. 12 models in total, each with either of A3, A3, or A4 models by $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4$\}$. In tables, accuracy of an architecture is the average accuracy of all the models with that architecture, e.g., in Table X accuracy of A2 is average of accuracies of all models with A2 architecture in the two collaborations.

First, we show that the heterogeneous collaboration between

models of equivalent capacities does not reduce the accuracy of party models compared to its homogeneous counterparts. We consider four homogeneous collaborations each of 16 parties such that $\{P_{1:16}$ = A2$\}$, $\{P_{1:16}$ = A3$\}$, $\{P_{1:16}$ = A4$\}$, and $\{P_{1:16}$ = A5$\}$, and compare it with a heterogeneous collaboration: $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A5$\}$. The results are shown in Table IX.

Next, we show that **the presence of a few bad models does not affect the accuracy of the good models in the heterogeneous collaboration, while significantly benefits the bad models**. Specifically, we show that accuracy of the collaboration of 12 good models, i.e., $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4$\}$ remains unaffected even if 4 bad models are added to it, i.e., $P_{13:16}$ = A2, as shown in Table X.

Finally, we show that **heterogeneity allows for more knowledge sharing via collaboration and always improves the utility of collaborations**. We consider 4 homogeneous collaborations: $\{P_{1:4}$ = A1$\}$, $\{P_{1:4}$ = A2$\}$, $\{P_{1:4}$ = A3$\}$, and $\{P_{1:4}$ = A4$\}$ and compare them with a heterogeneous collaboration that includes all these 16 parties, i.e., $\{P_{1:4}$ = A1, $P_{5:8}$ = A2, $P_{9:12}$ = A3, $P_{13:16}$ = A4$\}$. Table XI shows that including more participants clearly benefits all types of models, although the bad models benefit more than the good ones. For instance, A1 models improve by 8% from 70.1% to 78.1% due to heterogeneous collaboration, while A2, A3, and A4 models improve by 3.2%, 2.2%, and 1.4%, respectively.

Table IX: Comparison between heterogeneous and homogeneous collaborations in Cronus.

Homogeneous				Heterogeneous
$P_{1:16} \rightarrow$ A2	A3	A4	A5	$\{P_{1:4}$ = A2, $P_{5:8}$ = A3 $P_{9:12}$ = A4, $P_{13:16}$ = A5$\}$
89.6	89.3	88.4	88.6	89.3

Table X: Effect of the presence of low accuracy bad models on the performance of higher accuracy good models. n is the number of collaborating parties.

Models	Heterogeneous $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4$\}$	Heterogeneous $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A1$\}$
A1	-	78.1
A2	88.5	88.7
A3	88.6	88.1
A4	88.7	88.1

Table XI: More participation due to heterogeneity always improves the overall utility of the collaboration.

Models	Homogeneous 4 small collaborations $P_{1:4}$ = A1/A2/A3/A4	Heterogeneous $\{P_{1:4}$ = A2, $P_{5:8}$ = A3, $P_{9:12}$ = A4, $P_{13:16}$ = A1$\}$
A1	70.1	78.1
A2	85.5	88.7
A3	85.9	88.1
A4	86.7	88.1

E. Communication overhead of Cronus

Cronus significantly reduces the communication due to sharing of predictions instead of parameters of party models. Now, we demonstrate that using random subset of public data in each epoch further reduces communication without compromising the accuracy of Cronus. In Figure 3, we show the *Cronus training progress in its collaboration phase* when public data of sizes 2k, 4k, 6k, 8k, 10k is sampled randomly in each epoch; the corresponding accuracies are 88.4, 89.4, 88.7, 89.3, 89.6, respectively. Knowledge transfer improves convergence of student models [7], and therefore, sub-sampling does not reduce the convergence rate of Cronus. Therefore, Cronus does not only reduce the per-epoch communication, but also reduces the overall communication.

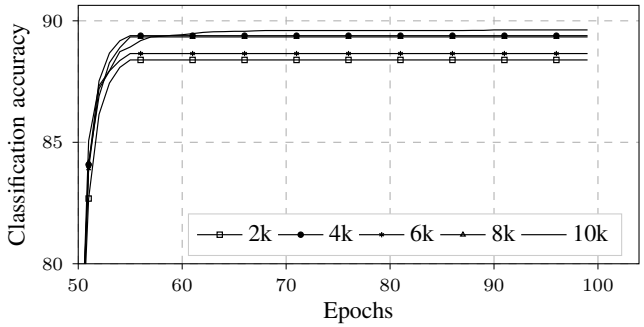

Figure 3: Communication reduction by randomly subsampling from the 10k public data in each epoch of Cronus collaboration phase. The final Cronus accuracy is not affected even when just 2k samples are used in each Cronus epoch.

VIII. Related Work

Federated learning aims at training machine learning models via collaboration, without sharing data among the participants [47]. However, existing federated learning algorithms are susceptible to various poisoning attacks which deters data holders from trusting the algorithm [12], [10], [36]. Bagdasaryan et al. [10] demonstrate a targeted poisoning attack against FedAvg [35] with just one malicious party. Bhagoji et al. [12] also propose a targeted attack with a single malicious party to work against two robust aggregation schemes, Krum [13] and coordinate-wise median [57].

To address the poisoning attacks, many robust aggregation schemes are proposed in the literature ([37], [13], [57]). Yin et al. [57] analyze median and trimmed-mean aggregation rules with provable robustness guarantees. Median and geometric-median based robust aggregation rules are also extensively explored [55], [51], [5]. Blanchard et al. [13] propose Krum aggregation, a computationally tractable modification of geometric-median, which selects the update that is closest to its neighbors as the aggregate in a given epoch. El Mhamdi et al. [37] propose Bulyan, a two-step meta-aggregation algorithm based on the Krum and trimmed median, which filters malicious updates followed by computing the trimmed median of the remaining updates. In spite of their robustness guarantees, these algorithms are shown to be susceptible to targeted or generic poisoning attacks against deep learning models [11], [12].

There is a long line of research that considers the problem of robust mean estimation of a corrupted set of data samples [16],

[17], [15], [33], and proposes algorithm to tighten the error bounds and reduce its dependence on the data dimensionality. The error rate of most of these aggregation algorithms depends on the dimensionality of their inputs [31], [17], [16]. Therefore, the high dimensional updates in federated deep learning setting significantly reduces the effectiveness of these algorithms (as their error bounds in presence of poisoning large vectors are lose). In summary, the high dimensional updates hinder the use of sophisticated robust algorithms and also enable strong poisoning attacks. Our Cronus algorithm, inspired by the knowledge transfer algorithms [9], [21], addresses these concerns by sharing the knowledge of the local party models via their predictions on a non-sensitive public data. This novel knowledge sharing federated learning scheme significantly reduces the dimensions of updates, and therefore, reduces the aggregation error.

Although federated learning prevents its participants from sharing their private data, recent works [40], [42], [36] demonstrate passive and active inference attacks against this setting, and successfully infer sensitive information about the parties' private data. Such inference attacks can be defended by using differentially private (DP) learning [4], [14], [18]. By trusting the aggregator, McMahan et al. [14] consider a federated setting where the server collects clipped gradient updates from the parties and then adds a calibrated noise using bounded sensitivity estimators; however, this algorithm can achieve acceptable accuracy only with a large number of participants. Besides, it does not protect data privacy against the aggregator. Treux et al. [52] propose a hybrid approach for privacy-preserving federated learning that leverages DP and secure multi-party computation among collaborators. These works, although provably privacy-preserving, are not robust to poisoning attacks and produce models with undesirably poor privacy-utility tradeoffs [27].

Knowledge of ensemble of teacher models has been used to train a student model in a few previous works [29], [43], [44], [7], [28]. Papernot et al. [43] propose PATE, a centralized learning approach that uses ensemble of teachers to label a subset of unlabeled non-sensitive public data, and then, trains a student in a semi-supervised fashion with differential privacy [46]. PATE's setting is fundamentally different from that of Cronus in that all the teacher models and the noise generation mechanisms in PATE are performed by a trusted entity who owns all the data. Co-distillation approach is a method to use distillation on private training data with other parties [7], [28], however, without defending against data poisoning and inference attacks [48], [56], [22].

IX. Conclusions

We have proposed Cronus learning algorithm as a variant of federated learning to mitigate the three fundamental limitations of federated learning: models are susceptible to data/parameter poisoning attacks, and inference attacks against local training data, and cannot operate on models with heterogeneous architectures. We have used the predictions of the local models, instead of their parameters, to exchange the knowledge between the local private models. This enables heterogeneous federated learning. It also significantly reduces the dimensionality of inputs of aggregation algorithms, which results in tight robustness guarantees even for networks with a small number of participants. Distilling the models' knowledge, through their prediction vectors, completely eliminates the possibility of white-box inference attacks, reduces the overall leakage about local datasets, and enables using utility-preserving mechanisms for prediction privacy.

References

[1] "Acquire Valued Shoppers Challenge," https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data, 2019, [Online; accessed 11-September-2019].

[2] "PyTorch Documentation," https://pytorch.org/, 2019, [Online; accessed 11-September-2019].

[3] "Tensorflow Privacy," https://github.com/tensorflow/privacy/tree/master/privacy/analysis 2019, [Online; accessed 23-September-2019].

[4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[5] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2018, pp. 4613–4623.

[6] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[7] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," *arXiv preprint arXiv:1804.03235*, 2018.

[8] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and it's applications," *Theory of Computing*, 2012.

[9] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[10] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[11] M. Baruch, B. Gilad, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Advances in Neural Information Processing Systems*, 2019.

[12] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*, 2019, pp. 634–643.

[13] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

[14] M. H. Brendan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *International Conference on Learning and Representation*, 2018.

[15] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *International Conference on Machine Learning*, 2019, pp. 1596–1606.

[16] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high dimensions without the computational intractability," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 655–664.

[17] ——, "Being robust (in high dimensions) can be practical," *In Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

[18] C. Dwork and V. Feldman, "Privacy-preserving prediction," in *Conference On Learning Theory*, 2018, pp. 1693–1702.

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences 55, no. 1*, 1997.

[20] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing 37, no. 2*, 2007.

[21] H. Geoffrey and V. O. amd Dean Jeff, "Distilling the knowledge in a neural network," *NIPS 2014 Deep Learning Workshop*, 2014.

[22] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[23] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigations in multi-party machine learning," *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

[24] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[25] P. J. Huber, *Robust statistics*. Springer, 2011.

[26] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures against regression learning," *39th IEEE Symposium on Security and Privacy*, 2018.

[27] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX Security Symposium*, 2019.

[28] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[29] H. Jihun, C. Yingjun, and B. Mikhail, "Learning privately from multiparty data," *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[31] K. Lai, A. Rao, and S. Vempala, "Agnostic estimation of mean and covariance," *In 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016.

[32] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] J. Z. Li, "Principled approaches to robust machine learning and beyond," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[34] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014.

[35] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics*, 2017.

[36] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *40th IEEE Symposium on Security and Privacy*, 2019.

[37] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*, 2018, pp. 3518–3527.

[38] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 27–38.

[39] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial tuning," *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[40] ——, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *Security and Privacy (SP), 2019 IEEE Symposium on*, 2019.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[42] T. Orekondy, S. J. Oh, B. Schiele, and M. Fritz, "Understanding and controlling user linkability in decentralized learning," *arXiv preprint arXiv:1805.05838*, 2018.

[43] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *International Conference on Learning and Representation*, 2017.

[44] N. Papernot, S. Song, U. Erlingsson, I. Mironov, A. Raghunathan, and T. Kunal, "Scalable private learning with PATE," *International Conference on Learning and Representation*, 2018.

[45] S. A. Plotkin, D. B. Shmoys, and E. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Foundations of Computer Science*, 1991.

[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[47] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015.

[48] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[49] G. Song and W. Chai, "Collaborative learning for deep neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1832–1841.

[50] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," *Advances in Neural Information Processing Systems*, 2017.

[51] L. Su, "Defending distributed systems against adversarial attacks: Consensus, consensus-based learning, and statistical learning," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2017.

[52] S. Treux, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, and R. Zhang, "A hybrid approach to privacy-preserving federated learning," *arXiv preprint arXiv:1812.03224v1*, 2018.

[53] D. Wagner, "Resilient aggregation in sensor networks," *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004.

[54] X. Wang, R. Zhang, Y. Sun, and J. Qi, "Kdgan: knowledge distillation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 775–786.

[55] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, 2018.

[56] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing connection to overfitting," *In 2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018.

[57] D. Yin, K. Chen, Yudong Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

APPENDIX

A. Details datasets and model architectures

We use four datasets in our evaluation, whose details follow.

SVHN. SVHN [41] dataset contains Google's street view images of house numbers. The images are 32x32, with 3 floating point numbers containing the RGB color information of each pixel. We use the extended SVHN dataset with 630,420 samples to train 32 party models each with 5,000 training samples; the public data size is 10,000. We use validation and test data of sizes 2,500 each. The reference data required for adversarial regularization is of the same size as that of training data for the cases of all the datasets.

MNIST. MNIST [32] dataset contains 28x28 images of handwritten digits and is composed of 60,000 training samples and 10,000 test samples. The dataset contains 10 classes each with 60,000 training and 1,000 test samples. We use validation and test data of sizes 1,000 each. We use 28 parties each with 2,000 training and reference samples, and public data size is 10,000.

Purchase. Purchase [1] dataset contains the shopping records of several thousand online customers, extracted during Kaggle's Acquire Valued Shopper challenge [1]. The dataset contains 197,324 data records with feature vectors of 600 dimensions and corresponding class label from one of total 100 classes. We use validation and test data of sizes 2,500 each. We use 16 parties each with 10,000 training and reference data, and public data size is 10,000.

CIFAR10. CIFAR10 [30] has 60,000 color (RGB) images (50,000 for training and 10,000 for testing), each of 32×32 pixels. The images are clustered into 10 classes based on the objects in the images and each class has 5,000 training and 1,000 test images. We use validation and test data of sizes 2,500 each. We use 16 parties each with 2,500 training data, and public data size is 10,000.

Model architectures. For SVHN, we use a neural network with three convolution layers and one fully connected layer, and Relu activations. For the MNIST dataset, we use a fully connected neural network with layer sizes {784, 256, 64, 10} and Relu activations. For the Purchase dataset, we use fully connected neural networks with layer sizes {600, 1024, 100} and Tanh activations. For CIFAR10 dataset, we use DenseNet architecture [24] with 100 layers and growth rate of 12. For the heterogeneity experiments with Purchase dataset, we use 5 fully connected networks with hidden layer sizes [{},{1024},{512, 256},{1024, 256}, {1024, 512, 256}]; here {} implies that the corresponding model has no hidden layers.

Training hyper-parameters. The initialization and collaboration phases of SVHN, MNIST, and Purchase trained models are of 50 epochs each. In both the phases, we train party models on their local training data using Adam optimizer at 0.0005 learning rate. Additionally, in collaboration phase, i.e., for epochs 50-100, we train party models on public data, $(X_p, \bar{Y})$, using SGD optimizer at a learning rate of 0.001. For CIFAR10, we train models for 200 epochs using SGD optimizer with 0.1 learning rate, 0.9 momentum, and 10^-4 weight decay in both the phases. Additionally, in collaboration phase, we train the models on public data using SGD optimizer at 0.01 learning rate, 0.99 momentum, and 10^{-6} weight decay.

For experiments of membership privacy assessment, we use state-of-the-art whitebox inference model proposed in [40], and use the gradients and outputs of its last layer, in addition to the blackbox access features including prediction of input and its cross-entropy loss. We train the inference model using Adam optimizer at a learning rate of 0.0001 for 100 epochs.

B. Additional Details of the Experiment

In this section, we provide the experimental details omitted in Section VII.

Complete robustness assessment. In Table VI of Section VII-A, for each dataset and each aggregation algorithm, we showed the accuracy of the attack that is strongest among all the attacks discussed in Section IV. We compute empirical robustness of aggregation algorithms using this strongest attack as described in Section VI. Here, in Table XII, we give the complete evaluation of all the attacks on all of the aggregation algorithms and datasets we consider in this work. The 'Accuracy (Benign)' row of each dataset shows the results in the absence of adversary. The worst accuracy for a combination of aggregation algorithm and dataset is highlighted in the corresponding column; the corresponding strongest attack can be found from the label of the row of the highlighted cell. Observe that, label flip attack seems to have lower effect on mean aggregation than the other aggregations; this is because, unlike other aggregations, in case of mean, there is only single malicious client. Note that, MWUAvg and MWUOpt aggregations are robust against all the existing attacks in the literature, but are completely ineffective against the attack we introduced in Section IV-D. Also, note that, Bulyan aggregation is empirically the most robust aggregation after Cronus, but it allows only 25 - 33% malicious clients compared to other aggregation algorithms such as Krum, in other words, Bulyan has a very low breaking point. The numbers of malicious parties used in each of our experiments are given in Table V.

Convergence plots for benign setting. In Figure 1 of Section VII, we show the convergence plots of various aggregation algorithms in federated learning in the adversarial setting. Figure 4 shows the similar convergence plots for the benign setting. Cronus incurs only a slight reduction in accuracy at a significantly higher gain in robustness and privacy as shown in Sections VII-A and VII-B.

Table XII: Evaluation of the conventional federated learning with various aggregation schemes with Cronus learning using the strong poisoning attacks described in Section IV. Robustness in Table VI is measured as the ratio of the accuracy of the final model/s when the strongest attack is mounted and the accuracy in the benign setting; the strongest attack is determined empirically as the one that maximally reduces the accuracy of the corresponding federated learning aggregation.

Dataset		Federated learning with various aggregation algorithms						Cronus
		Mean	Median	MwuAvg	MwuOpt	Bulyan	Krum	
SVHN	Accuracy (Benign)	**95.9**	94.8	93.9	94.4	94.5	89.6	91.1
	Label flip	92.9	90.1	91.2	89.3	93.9	88.6	89.8
	LIE	14.8	14.5	91.6	92.0	15.5	16.2	91.5
	OFOM	0.9	94.5	0.9	0.7	94.4	89.0	91.0
	PAF	12.8	16.4	95.1	93.1	93.4	87.5	91.1
MNIST	Accuracy (Benign)	96.7	96.5	**97.2**	97.4	96.9	93.3	95.2
	Label flip	96.3	94.4	94.7	93.6	96.8	89.9	95.0
	LIE	95.1	93.1	95.6	96.7	94.1	94.3	95.9
	OFOM	22.1	97.3	25.3	36.0	97.1	94.4	96.1
	PAF	9.6	91.5	96.9	12.7	97.1	94.0	96.2
Purchase	Accuracy (Benign)	93.3	93.0	**93.6**	92.5	92.8	72.1	89.6
	Label flip	88.9	89.9	63.4	67.6	91.7	74.8	88.0
	LIE	2.5	69.3	92.2	85.6	81.8	49.6	89.2
	OFOM	1.4	92.8	1.8	1.1	92.6	74.5	89.4
	PAF	1.1	12.5	93.0	88.0	91.0	76.6	89.4
CIFAR10	Accuracy (Benign)	88.4	**89.1**	86.2	87.6	89.0	84.5	80.1
	Label flip	–	–	–	–	–	–	79.8
	LIE	18.9	61.2	86.0	84.3	75.6	18.0	78.0
	OFOM	12.9	89.5	14.2	12.8	89.1	85.0	78.5
	PAF	11.3	15.1	86.4	85.0	89.0	839	79.0

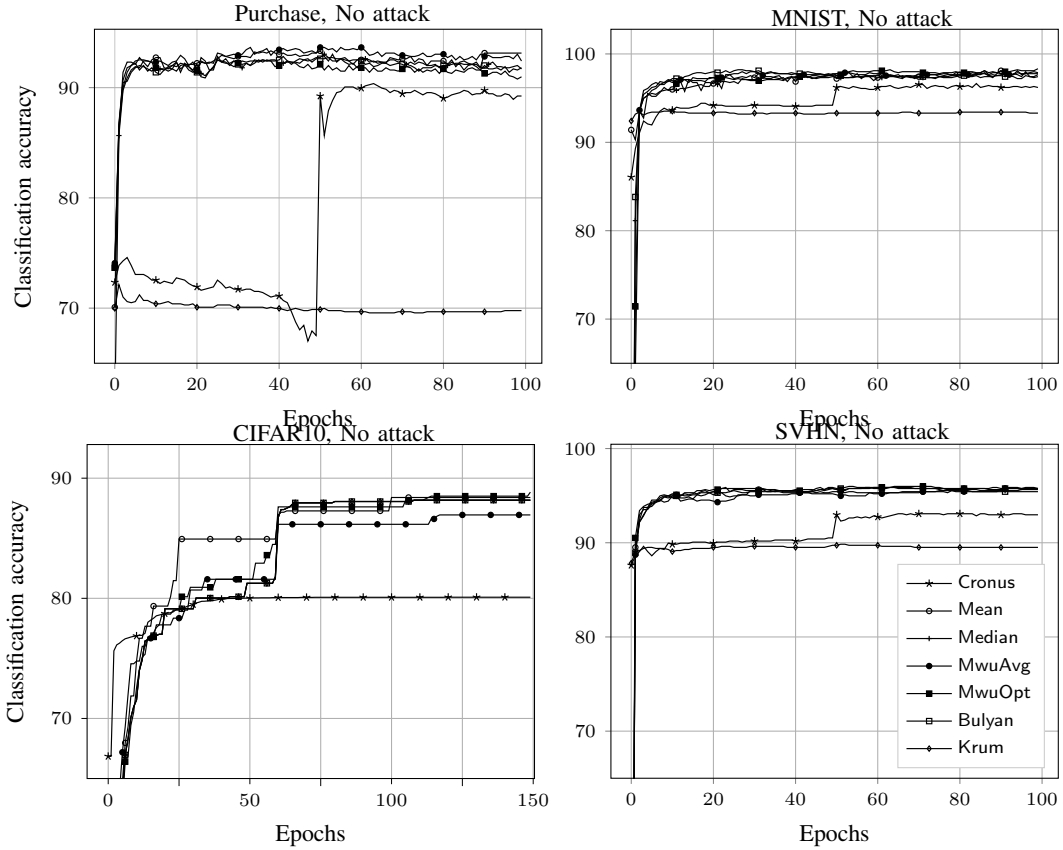

Figure 4: Convergence of Cronus and existing federated learning algorithms in **benign setting**. Cronus incurs only a slight degragation in accuracy compared to existing algorithms, while improves significantly over stand-alone training.