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

Chang Hongyan[1]* **Virat Shejwalkar**[2]* **Reza Shokri**[1] **Amir Houmansadr**[2]
[1]National University of Singapore, [2]University of Massachusetts Amherst

Abstract

Majority of existing collaborative learning algorithms propose to exchange local model parameters between collaborating clients through a central server. Unfortunately, this approach has many known security and privacy issues, primarily because of the high dimensionality of the updates involved; furthermore, it is limited to clients with homogeneous model architectures. The high dimensionality of the updates makes these approaches more susceptible to poisoning attacks and exposes the local models to private information inference attacks. Based on this intuition, we propose *Cronus* that uses *knowledge transfer via model outputs* to exchange information between clients. We show that this significantly reduces the dimensions of the clients' updates, and therefore, improves the robustness of our Cronus algorithm. Our extensive evaluations demonstrates that Cronus outperforms state-of-the-art robust federated learning algorithms. We also show that treating local models as black-box significantly reduces the information leakage measured using the membership inference attacks. Finally, Cronus also allows collaboration between clients with heterogeneous model architectures.

1 Introduction

Collaborative machine learning (ML) [1, 2] is a promising approach for building machine learning models using distributed training data held by multiple parties, called *clients*. The collaborating clients repeatedly exchange their local models through a *central server* in order to train a shared *global model*, without sharing their private data with server or any other clients. This is very attractive to parties who own sensitive data and agree on performing a common machine learning task, but are unwilling to pool their data together for centralized training. Hence, many real-world applications have adopted collaborative learning approach, e.g., Google's Gboard [2] and Apple's Siri [3].

A popular collaborative learning approach called *federated learning* (FL) assumes *homogeneous* local models, i.e., with the same architecture [2]. In each FL round: (1) the server broadcasts the global model to all clients; (2) each client trains the global model locally for few epochs and shares a *model update* with the server; (3) server aggregates the model updates using an aggregation rule (AGR) and updates the global model using the aggregate. Three of the major obstacles hindering the scalable deployment of secure and truly privacy-preserving federated learning are: (1) susceptibility to poisoning attacks [4, 5, 6], (2) susceptibility to private information inference attacks [7], and (3) lack of support to clients holding heterogeneous model architectures.

In this paper, our main focus is on securing collaborative learning from poisoning attacks. There exists a long chain of recent poisoning attacks and defenses for federated learning [8, 9, 10, 11, 12]. However, most of the defenses built upon robust AGRs [10, 8, 9, 13, 4] are susceptible to some form of poisoning attack [14]. The core reason for this susceptibility is the high-dimensionality of the clients' model updates: the theoretical error bounds of these robust AGRs [10, 8] depend on the dimensionality of their inputs, i.e., of model updates in FL. The models in modern FL generally have very high dimensionality which makes the error bounds of robust AGRs prohibitively high. The robust

*First two authors contributed equally.

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

AGRs that achieve the dimension-independent error bounds [15, 16, 17] have $\mathcal{O}(d^3)$ computational complexities, where d is the number of parameters. Hence, the large number of parameters in modern FL prohibits the use of these robust AGRs in practice.

Furthermore, sharing of model parameters exposes the benign clients to the risk of inference attacks from the (potentially malicious) server and clients; the problem is especially worse with high dimensional model updates [7]. Thus, *sharing model parameters to transfer the knowledge of local models is not a secure design choice in collaborative learning*, especially considering that this is not the only way of exchanging knowledge between models.

Our contributions. In this paper, we design *Cronus*, a robust collaborative learning method to circumvent the security issues of FL. Cronus extracts, aggregates, and transfers the knowledge of the clients' local models in a black-box fashion using the knowledge transfer paradigm [18, 19]. Knowledge transfer is commonly used for compression and regularization purposes in ML. Cronus supports *heterogeneous* model architectures, because the clients share knowledge of their local models via knowledge transfer, and therefore, they are only required to agree on the same ML task. Following the previous works that use knowledge transfer in collaborative ML [20, 21], we assume that an unlabeled public dataset is available for the knowledge transfer.

Cronus algorithm. Specifically, Cronus operates in two phases: (1) In the *initialization phase*, clients train their models on their local data until convergence. (2) In each round of the *collaboration phase*: (a) clients compute predictions on the *unlabeled* public data and share the predictions with the server; note that, unlabeled public data is abundant, but labeled public data is not available due to the expensive manual labelling process involved, (b) the server aggregates the predictions of clients for each sample of public data and shares the *aggregate predictions* with all clients, (c) clients *fine-tune* their local models using the unlabeled public data and aggregate labels, while also training with their own private training data. We run the collaboration phase until convergence.

Intuition behind security and privacy due to Cronus. By sharing the predictions on the public data, Cronus reduces the number of dimensions of the clients' updates to the size of the outputs of client models, which is orders of magnitude smaller than the number of client model parameters. Cronus aggregates the clients' predictions on each sample from public data using state-of-the-art robust AGR [16]. The computational complexity of this robust AGR is $\mathcal{O}(d^3)$ (d is the number of model parameters), and hence, it is computationally prohibitive to use with model parameter based FL. Cronus, on the other hand, significantly reduces the dimensionality of updates, and therefore, makes the use of the state-of-the-art robust AGR practical. In addition, knowledge transfer using data that is disjoint to the target models' training data is an effective regularization technique [22], and can reduce information leakage about the training data [23]. To summarize, Cronus leverages knowledge transfer to exchange the knowledge of clients' local data and mitigates security and privacy issues in FL algorithms based on parameter sharing.

Evaluations. We comprehensively evaluate the security and privacy of Cronus. We use state-of-the-art attacks to compare security due to Cronus and existing robust FL algorithms; we also design new attacks to break a class of AGRs based on *multiplicative weights updates* [24, 25]). We show that, in parameter sharing based FL, one or more of poisoning attacks reduce the accuracy of the global model to random guessing for all of the considered robust AGRs. However, **under any attack, the reduction in average of accuracies of client models in Cronus is negligible** and is less than 2%; for the strongest attack on Cronus (of all considered attacks), the reduction in average accuracy of client models is 1.6% for Purchase, 1.3% for SVHN, 1.5% for MNIST, and 2.1% for CIFAR10.

We also empirically show that, the **clients in Cronus enjoy significantly higher privacy compared to existing FL algorithms**; we measure the privacy as the risk of active and passive membership inference attacks [26]. Finally, we empirically show that **Cronus is highly effective even when the clients have heterogeneous model architectures**.

2 Background and Related Work

Threat Models. We consider a *poisoning* adversary who controls an ϵ fraction of the total n clients, called *malicious clients*. The goal of the adversary is to mount an *untargeted* poisoning attacks [6, 14, 4] that reduce the accuracy of the global model (or of local models in case of Cronus) on most of the test inputs. For this, the adversary instructs the ϵn malicious clients to send malicious updates during collaborative learning. For the worst-case security analyses, we assume that the adversary has access to the benign clients' updates. Therefore, the adversary can use this data to

Table 1: **Theoretical error rates of various AGRs.** Error rate is the L_2 distance between the outputs of robust AGR in benign setting (all benign clients) and under attack (some clients malicious). n is the number of clients, ϵ is the breaking point, d is the dimensionality of AGR's input, and σ^2 is the variance of each of the dimensions.

Aggregation rule (AGR)	Breaking point	Statistical error rate	Computational cost
Mean [2]	$1/n$	Unbounded	$O(nd)$
Median [17]	$1/2$	$O(\sigma\epsilon\sqrt{d})$	$O(nd\log n)$
Krum [10]	$(n-2)/2n$	$O(\sigma n\sqrt{d})$	$O(n^2d)$
Bulyan [8]	$(n-3)/4n$	$O(\sigma\sqrt{d})$	$O(n^2d)$
RobustFilter (Our work) [16]	$1/2$	$O(\sigma\sqrt{\epsilon})$	$O(d^3+n)$

craft effective malicious updates. We also assume that the adversary has complete knowledge of the server's AGR.

We also evaluate the risk of membership inference attacks (MIAs) [23, 26] when the central server is the adversary. The goal of the adversary, i.e., the server, is to distinguish between the members and non-members of the private training data of collaborating clients using their updates in each round. We consider both *passive* and *active* MIAs: in the passive MIA, the server mounts MIAs using the updates without tampering with the training protocol. While in the active MIAs, to facilitate MIA [26], the server modifies the aggregated updates before sharing it with all the clients.

Aggregation Rules (AGRs). The most basic and effective AGR, federated averaging (FedAvg) [2] is based on *weighted averaging* of client updates. Here, the weight of the i^{th} client is proportional to the size of their local training data. FedAvg is widely used in non-adversarial settings, but it is not robust against even a single malicious client [10]. To improve the resilience of federated learning to poisoning attacks, multiple *robust* AGRs [10, 8, 9, 13, 4] are proposed in the literature as reviewed in [14]. Robust AGRs aim to remove the malicious updates to reduce their poisoning impact on the global model. We compare the upper bound of the error on the aggregate results for each aggregation rule in Table 1. We defer reader to Appendix A for more details about AGRs.

Little is enough attack (LIE). Existing robust AGRs, e.g., Krum and Bulyan, assume that in order to mount an effective attack, the malicious updates should lie far from the benign updates. However, Baruch et al. [6] challenge this assumption and propose an attack called *little is enough* (LIE). LIE crafts its malicious updates by adding small amounts of noises to each of the dimensions of a benign update; a benign update can be an average of the updates from benign clients. LIE mounts successful attacks on the state-of-the-art robust AGRs, including Bulyan, Krum, and Median AGRs. From Table 1, we note that the error rates of all of these AGRs depend on the dimensionality of their inputs, which is very large in FL due to the extremely large number of modern neural networks. Hence, we argue that this *curse of dimensionality makes existing AGRs more susceptible to untargeted poisoning.*

High-Dimensional Robust Mean Estimation. The robust AGRs in FL try to recover the mean of the benign updates from the set of benign and malicious updates. This is precisely the objective of the robust mean estimation problem from robust statistics. Diakonikolas et al. [16] propose a filter-based algorithm RobustFilter that achieves optimal error guarantee $(O(\sigma\sqrt{\epsilon}))$ and optimal sample complexity $(\Theta((d/\epsilon)\log d))$. From Table 1, we note that RobustFilter has tighter error rate guarantees than existing robust AGRs, i.e., it is more robust to untargeted poisoning. However, using this RobustFilter in parameter sharing based FL is impractical due to its dimension-dependent sample and computational complexities. We address this impracticality issue in our work.

Multiplicative Weight Update (MWU). In this work, we also evaluate the AGRs based on multiplicative weight update (MWU) technique [24, 27, 28, 29, 25]. These techniques assume that, compared to the benign updates, malicious updates lie farther away from the average of all input updates. Hence, for an update, they assign weight that is inversely proportional to the distance of the update from the average of all updates. We evaluate two AGRs: MWU with averaging (MWUAvg) [24] and MWU with optimization (MWUOpt) [25]. In Section 5, we empirically show that MWU based robust AGRs are robust to existing poisoning attacks, but can be broken using our improved poisoning attack that we will describe in Section 3.

3 Poisoning attack against MWU-based aggregations (OFOM).

MWU-based AGRs are robust against the existing poisoning attacks. However, we propose a simple, yet highly effective attack against these AGRs. MWU-based AGRs treat the updates near the empirical

weighted mean as the benign updates. However, the weighted mean is not robust and our attack exploits this fact. Specifically, the adversary crafts two malicious updates: the first malicious update is arbitrarily far from the mean of the benign updates. While the second malicious update is set to the empirical mean of benign updates and the first malicious update. Therefore, our attacks force MWU-based AGRs to assign very high weights to the second malicious update. Note that the second malicious update is very far from the mean of the benign updates, and hence, effectively corrupts the global model, as we will show in Section 5.

4 Cronus Collaborative Learning

In this section, we first explain the shortcomings of parameter sharing based FL algorithms and then detail our Cronus learning algorithm. The major limitations of existing parameter sharing based FL (Section 1) algorithms [2] are:

Robustness: As shown in Table 1, the upper bound on the error rates of existing robust AGRs depend on their inputs' dimensionality, i.e., on the large number of model parameters in FL. This makes these AGRs significantly susceptible to untargeted poisoning attacks such as LIE [6].

Privacy: Clients directly sharing their local model parameters facilitates strong white-box inference attacks by the (potentially malicious) central server [7, 26]. Furthermore, the larger the clients' models, the more the private information leakage through their model parameters.

Heterogeneity: Parameter sharing based FL algorithms are restricted to *homogeneous* architectures, i.e., all clients must have the same model architecture. This is not always possible, especially when clients have different resource constraints, e.g., on-device memory and Internet connection stability.

To remedy the above shortcomings, the collaborating clients should share the knowledge that they learn from their training data in a succinct way. This is the main objective of *knowledge transfer* [18, 19, 30, 31]. In particular, *knowledge distillation* [18] is introduced as a means of compressing large models, called *teachers*, into smaller models, called *students*, while retaining their accuracy. It efficiently transfers the functionality if a model (or an ensemble of models) to a student model [18]. 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 models (as compared to training directly on hard-labeled data) [18, 19, 31]. Furthermore, knowledge transfer is an effective regularization method [22].

Motivated from the aforementioned benefits of knowledge transfer, in this paper, we leverage knowledge transfer and propose ***Cronus***, a collaborative learning algorithm, where the server extracts the knowledge of clients' local models via their outputs and aggregates them in a robust manner.

Overview of Cronus. To extract and exchange the knowledge of local models, Cronus uses a set of *unlabeled* public data, X_p, which is essentially a set of feature vectors. Algorithm 1 describes the two training phases of Cronus: In the first phase, called the *initialization phase*, a client i trains their local model θ_i only on their local training data D_i for T_1 epochs *without any collaboration*.

In the second phase, called the *collaboration phase*, the clients share the knowledge of their local models via their predictions on the public dataset, X_p. Specifically, epoch t of this phase includes three steps: (**S1**) Client i computes prediction vectors Y_i^t for the public data X_p using their local model θ_i^t and shares Y_i^t with the server. (**S2**) The server aggregates the predictions (separately for each sample $\in X_p$), i.e., computes $\bar{Y}^t = f_{\mathsf{Cronus}}(Y_i^t, .., Y_n^t)$, and then sends $\bar{Y}^t$ to all the clients; we use the robust mean estimation algorithm RobustFilter [16] as the aggregation algorithm f_{Cronus} of the server in Cronus. (**S3**) Client i updates their local model θ_i^t using their private data $\mathbf{D}_i$ and the labeled public data $(X_p, \bar{Y}^t)$ to obtain θ_i^{t+1} for the next epoch.

Recall that the size of the updates determines the error rate of the existing robust AGRs. Hence, just by reducing the dimensionality of the updates (from the size of the model to the size of the prediction vector), Cronus already reduces the error rate guaranteed by any AGR, including that of Bulyan and Median. However, when the size of the prediction vector is larger, these AGRs might still be highly susceptible to untargeted poisoning. To address this issue, Cronus uses RobustFilter [16], which achieves the dimension independent error rate guarantees. Furthermore, its sample and computational complexities have the least dependence on the dimensionality of updates among existing robust mean estimation algorithms. Nevertheless, we note that, Cronus is compatible with any robust AGR.

Algorithm 1 Cronus algorithm. Initialization phase does not involve collaboration. D_i and θ_i are local dataset and model parameters 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 $\theta_i \leftarrow \mathsf{TRAIN}\ (\theta_i, D_i)$
end for
$Y_i^0 = \mathsf{PREDICT}(\theta_i; X_p)$ ▷ Compute initial predictions on X_p
Send Y_i^0 to the server

Collaboration phase

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

The sample complexity of RobustFilter is $\Theta(d \log d)$, which is the number of clients required to achieve the theoretical error bound. This dimension-dependent sample complexity makes RobustFilter [16] impractical to use in parameter sharing based FL. In contrast, the low dimensional updates in Cronus significantly reduce the number of clients required for RobustFilter to achieve desired error rate. For instance, Cronus reduces sample complexity by an order of 10^5 when training DenseNet on Cifar10 dataset. Therefore, unlike any existing FL, Cronus can enjoy the strong theoretical error guarantees of RobustFilter, even with a small number of clients in the network.

Unlike the parameter sharing based FL algorithms, *Cronus does not force a single global model* onto local models. Instead, each local model is updated separately by improving its classification accuracy and resilience to inference attacks. This further improves the utility of local models. Note that, such fine-tuning has significant fairness advantages [32, 33]. Furthermore, Cronus completely eliminates the risk of white-box inference attacks, as no client releases their model's parameters. Furthermore, Cronus allows its clients to use any privacy preserving mechanism [34] during local training.

In conclusion, by combining knowledge transfer and state-of-the-art robust mean estimation RobustFilter, Cronus provides a practical collaborative learning algorithm that is robust and private, and also support clients with heterogeneous model architectures.

5 Experiments

Experimental Setup. We use four datasets, Purchase [23], CIFAR10 [35], SVHN [36], and MNIST [37], to evaluate efficacy of Cronus. We use PyTorch [38] for our evaluations. We defer the details of the dataset splits, model architectures, and hyper-parameters to Appendix D.1.

Baseline attacks. We evaluate FL and Cronus under LIE attack (Section 2) and our poisoning attacks (Section 3), as well as with *Label flip poisoning attack (Label flip)* and *Naive poisoning attack (PAF)*. In the label flip attack, the adversary flips the labels of malicious clients' local training data in the same way and shares the updates computed on these poisoned datasets. The naive poisoning attack (PAF) crafts malicious updates that are arbitrarily far from the average of the benign updates.

We compare the classification accuracies of the models trained using Cronus, stand-alone, central, and parameter sharing (FedAvg) settings, in the absence of any attack. In the stand-alone setting, each client trains its model only on its local data, without any collaboration. In centralized learning, a single model is trained on the union of the clients' data. For stand-alone and Cronus settings, the accuracy of models for different clients is different; therefore, we report the average classification

Table 2: **Experimental setup:** The number of malicious clients used for robustness assessment of different AGRs 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

Table 3: **Comparison of the robustness** of Cronus and of the parameter sharing based FL with various AGRs. Robustness is measured as the ratio of *worst accuracy* and *benign accuracy*: worst accuracy is the accuracy of the global model (for Cronus, this is the average accuracy of client models) under the strongest of all attacks (Section 5); shown in "Strongest attack" row. While the benign accuracy is the accuracy without any attack. Table 2 gives the numbers of benign and malicious clients. We highlight the best accuracy for each setting. Table 5 in Appendix D.2.1 shows all the results.

Dataset		Parameter sharing based FL with various aggregation rules (AGRs)						Cronus
		Mean	Median	MwuAvg	MwuOpt	Bulyan	Krum	
SVHN	Benign accuracy	**95.9**	94.8	93.9	94.4	94.5	89.6	91.1
	Worst accuracy	0.9	14.5	0.9	0.7	15.5	16.2	**89.8**
	Strongest attack	(OFOM)	(LIE)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LF)
	Robustness	0.01	0.15	0.01	0.01	0.16	0.18	**0.99**
MNIST	Benign accuracy	96.7	96.5	**97.2**	97.4	96.9	93.3	95.2
	Worst accuracy	9.6	91.5	25.3	12.7	**94.1**	89.9	93.7
	Strongest attack	(PAF)	(PAF)	(OFOM)	(PAF)	(LIE)	(LF)	(LF)
	Robustness	0.09	0.95	0.26	0.13	0.97	0.96	**0.99**
Purchase	Benign accuracy	93.3	93.0	**93.6**	92.5	92.8	72.1	89.6
	Worst accuracy	1.1	12.5	1.8	1.1	81.8	49.6	**88.0**
	Strongest attack	(PAF)	(PAF)	(OFOM)	(OFOM)	(LIE)	(LIE)	(LF)
	Robustness	0.01	0.75	0.02	0.01	0.87	0.69	**0.98**
CIFAR10	Benign accuracy	88.4	**89.1**	86.2	87.6	89.0	84.5	80.1
	Worst accuracy	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**

accuracy of all the client models. The accuracies of stand-alone, centralized, and Cronus settings for Purchase are 76.3%, 94.3%, and 89.6%, respectively, while for CIFAR10 they are 66.8%, 90.2%, and 80.1%, respectively. Cronus uses unlabeled data for knowledge transfer, which any client can use to improve its model's accuracy via semi-supervised learning [39, 40, 41, 42]. However, a CIFAR10 model trained in a semi-supervised manner using Mean-Teacher method [39] with 2,500 labeled and 10,000 unlabeled data achieves 69.75% accuracy while Cronus achieves 80.1% accuracy. That is, local models can achieve better accuracy via Cronus compared to the state-of-the-art semi-supervised learning methods.

Robustness. We compare the robustness of FL, using different aggregation algorithms (Section 2 and Appendix A), against an adversary who mounts the strong poisoning attacks. To assess the worst-case robustness, we evaluate algorithms against the largest fraction of malicious clients, which is smaller than the breaking point. The number of benign and malicious clients are given in Table 2. The robustness assessment results are shown in Table 3. From those results, it is clear that the FL algorithm works well in the absence of malicious clients; however, **all of the existing aggregation schemes in FL are significantly vulnerable to at least one poisoning attack.** The FedAvg [2] 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 AGRs, they are susceptible to the LIE attack. As explained in Section 2, 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 new OFOM attack, which we propose in Section 3. 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 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 two-fold. First, as detailed in Section 4, reduced dimensionality of updates enables the use of robust aggregation algorithm RobustFilter [16]. It guarantees that the aggregate prediction is robust even when the dimensionality of the prediction vector is high. For example, Cronus is still robust on the Purchase dataset when the dimensionality of the prediction vector is 100. Second, benign models have a stronger agreement on their predictions than on their parameter values. Thus, the variance of the prediction vector is smaller than that of gradient, which makes the error guarantee for aggregation algorithm RobustFilter even smaller. Hence, Cronus enjoys the tight robustness grantees of RobustFilter providing a practical, robust collaborative learning protocol.

Privacy. We show that Cronus considerably reduces the privacy risks compared to FL; we measure the privacy risk using the state-of-the-art membership inference attacks [26]. First note that, by design, **Cronus enjoys the benefits of knowledge transfer as a membership inference risk mitigation technique** [22, 43]. We further show the compatibility of Cronus with existing privacy-preserving mechanisms [44, 34] in Tables 6 and 7. The key discoveries from our experimental results are: (1) 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%. (2) The active membership inference attacks [26] significantly increase the privacy risk of the target data in the case of FedAvg. For Purchase dataset, the risk due to individual updates increases by 7.8% (77.1% to 84.9%), while due to aggregated update increases by 8% (74.7% to 82.7%). (3) In the case of Cronus, the active membership inference attacks are ineffective, and the increase in privacy risk is negligible. On Purchase dataset, the risk increases by 0.3% for individual updates while 1.1% for aggregated updates. See experimental results and discussion in Appendix D.2.2.

Cronus with heterogeneous model architectures. Due to the use of prediction based updates, Cronus allows clients with heterogeneous model architectures to participate in collaboration. The key findings of our experimental evaluations are: (1) **the heterogeneous collaboration between models of equivalent capacities does not reduce the accuracy of client models compared to their homogeneous counterparts**, (2) 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, and (3) **heterogeneity allows for more knowledge sharing via collaboration and always improves the utility of collaborations**. Due to space restrictions, we defer the details of experiments and discussion to Appendix D.2.3.

6 Conclusions

We propose Cronus, a collaborative learning framework, to address three major issues of existing FL algorithms: susceptibility to poisoning attacks, information leakage to the service provider (server), and inability to support clients with heterogeneous model architectures. Instead of model parameters, Cronus uses knowledge transfer via the predictions of the clients' local models on an unlabeled public dataset to exchange information between clients. Due to use of low-dimensional model outputs as updates, Cronus allows the use of state-of-the-art robust mean estimation algorithms from robust statistics literature. The combination of knowledge transfer and robust mean estimation algorithms makes Cronus highly robust to poisoning attacks. Furthermore, due to sharing of predictions, Cronus reduces the risk of information leakage to the server and allows collaboration between clients with heterogeneous model architecture. Via extensive evaluations, we demonstrate all the three benefits of Cronus—security, privacy, and support for heterogeneity.

References

[1] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 2015.

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

[3] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, et al. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503*, 2021.

[4] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *The Network and Distributed System Security Symposium (NDSS)*, 2021.

[5] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, aug 2020. USENIX Association.

[6] Moran Baruch, Baruch Gilad, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, 2019.

[7] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. *40th IEEE Symposium on Security and Privacy*, 2019.

[8] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3518–3527, 2018.

[9] Dong Yin, Kannan Chen, Yudong Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

[10] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[11] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643, 2019.

[12] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[13] Hamid Mozaffari, Virat Shejwalkar, and Amir Houmansadr. Fsl: Federated supermask learning. *arXiv preprint arXiv:2110.04350*, 2021.

[14] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on federated learning. *arXiv preprint arXiv:2108.10241*, 2021.

[15] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 655–664. IEEE, 2016.

[16] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. *In Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

[17] Jerry Zheng Li. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.

[18] Hinton Geoffrey and Vinyals Oriol amd Dean Jeff. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*, 2014.

[19] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[20] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[21] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[22] Virat Shejwalkar and Amir Houmansadr. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[23] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 2017.

[24] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: A meta-algorithm and it's applications. *Theory of Computing*, 2012.

[25] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei 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.

[26] Milad Nasr, Reza Shokri, and Amir Houmansadr. 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.

[27] Yoav Freund and Robert 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.

[28] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Foundations of Computer Science*, 1991.

[29] Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing 37, no. 2*, 2007.

[30] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. Kdgan: knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 775–786, 2018.

[31] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

[32] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[33] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

[34] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

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

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[40] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[41] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[42] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2019.

[43] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. *arXiv preprint arXiv:2110.08324*, 2021.

[44] Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015.

[45] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

[46] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

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

[48] Lili Su. *Defending distributed systems against adversarial attacks: Consensus, consensus-based learning, and statistical learning*. PhD thesis, University of Illinois at Urbana-Champaign, 2017.

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

[50] Jacob Steinhardt, Pang Wie Koh, and Percy Liang. Certified defenses for data poisoning attacks. *Advances in Neural Information Processing Systems*, 2017.

[51] Matthew Jagielski, Aline Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures against regression learning. *39th IEEE Symposium on Security and Privacy*, 2018.

[52] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[53] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning*, pages 1596–1606, 2019.

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

[55] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[56] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial tuning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[57] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1832–1841, 2018.

[58] McMahan H Brendan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *International Conference on Learning and Representation*, 2018.

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