

November 2023

## Quantifying and Enhancing the Security of Federated Learning

Virat Vishnu Shejwalkar  
*University of Massachusetts Amherst*

Follow this and additional works at: [https://scholarworks.umass.edu/dissertations\\_2](https://scholarworks.umass.edu/dissertations_2)



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Information Security Commons](#)

---

### Recommended Citation

Shejwalkar, Virat Vishnu, "Quantifying and Enhancing the Security of Federated Learning" (2023). *Doctoral Dissertations*. 2988.

<https://doi.org/10.7275/35985723> [https://scholarworks.umass.edu/dissertations\\_2/2988](https://scholarworks.umass.edu/dissertations_2/2988)

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact [scholarworks@library.umass.edu](mailto:scholarworks@library.umass.edu).

# QUANTIFYING AND ENHANCING THE SECURITY OF FEDERATED LEARNING

A Dissertation Presented

by

VIRAT SHEJWALKAR

Submitted to the Graduate School of the  
University of Massachusetts Amherst in partial fulfillment  
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2023

Manning College of Information and Computer  
Sciences

© Copyright by Virat Shejwalkar 2023

All Rights Reserved

# QUANTIFYING AND ENHANCING THE SECURITY OF FEDERATED LEARNING

A Dissertation Presented

by

VIRAT SHEJWALKAR

Approved as to style and content by:

DocuSigned by:  
  
B5B813BDD57045B...  
Amir Houmansadr, Chair

DocuSigned by:  
  
EB98E18D0E4D45C...  
Mohit Iyer, Member

DocuSigned by:  
  
9305D9C12A5A4F8...  
Daniel Sheldon, Member

DocuSigned by:  
  
C9A18793AC37413...  
Prateek Mittal, Member

---

Ramesh K. Sitaraman, Associate Dean for  
Educational Programs and Teaching  
Manning College of Information and Computer  
Sciences

# ABSTRACT

## QUANTIFYING AND ENHANCING THE SECURITY OF FEDERATED LEARNING

SEPTEMBER 2023

VIRAT SHEJWALKAR

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Amir Houmansadr

Federated learning is an emerging distributed learning paradigm that allows multiple users to collaboratively train a joint machine learning model without having to share their private data with any third party. Due to many of its attractive properties, federated learning has received significant attention from academia as well as industry and now powers major applications, e.g., Google’s Gboard and Assistant, Apple’s Siri, Owkin’s health diagnostics, etc. However, federated learning is yet to see widespread adoption due to a number of challenges. One such challenge is its susceptibility to poisoning by malicious users who aim to manipulate the joint machine learning model.

In this work, we take significant steps towards this challenge. We start by providing a systemization of poisoning adversaries in federated learning and use it to build adversaries with varying strengths and to show how some adversaries common in the prior literature are not practically relevant. For the majority of this thesis, we focus on untargeted poisoning as it can impact much larger federated learning population

than other types of poisoning and also because most of the prior poisoning defenses for federated learning aim to defend against untargeted poisoning.

Next, we introduce a general framework to design strong untargeted poisoning attacks against various federated learning algorithms. Using our framework, we design state-of-the-art poisoning attacks and demonstrate how the theoretical guarantees and empirical claims of prior state-of-the-art federated learning poisoning defenses are brittle under the same strong (albeit theoretical) adversaries that these defenses aim to defend against. We also provide concrete lessons highlighting the shortcomings of prior defenses. Using these lessons, we also design two novel defenses with strong theoretical guarantees and demonstrate their state-of-the-art performances in various adversarial settings.

Finally, for the first time, we thoroughly investigate the impact of poisoning in real-world federated learning settings and draw significant, and rather surprising, conclusions about robustness of federated learning in practice. For instance, we show that contrary to the established belief, federated learning is highly robust in practice even when using simple, low-cost defenses. One of the major implications of our study is that, although interesting from theoretical perspectives, many of the strong adversaries, and hence, strong prior defenses, are of little use in practice.

# TABLE OF CONTENTS

	Page
<b>ABSTRACT</b> .....	<b>iv</b>
<b>LIST OF TABLES</b> .....	<b>xi</b>
<b>LIST OF FIGURES</b> .....	<b>xiv</b>
 <b>CHAPTER</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Gaps in FL robustness literature and thesis contributions .....	4
1.1.1 No systemization of poisoning attacks against FL .....	5
1.1.2 No general framework to design FL poisoning attacks .....	6
1.1.3 Sub-optimal defenses against FL poisoning .....	8
1.1.4 Poor understanding of the impacts of poisoning on real-world FL .....	9
1.1.4.1 Evaluating traditional FL .....	10
1.1.4.2 Evaluating personalized FL .....	11
1.2 Conclusions and future works .....	12
<b>2. BACKGROUND AND RELATED WORKS</b> .....	<b>14</b>
2.1 Federated Learning .....	14
2.2 Poisoning Attacks on Federated Learning .....	15
2.2.1 Existing Untargeted FL Poisoning Attacks .....	16
2.2.2 Existing Defenses Against Untargeted Poisoning .....	18
2.2.3 Robust AGRs Studied in This Thesis .....	19
2.3 Related works .....	21

<b>3. SYSTEMATIZATION OF POISONING ATTACKS AGAINST FEDERATED LEARNING</b>	<b>24</b>
3.1 Systematization of FL Poisoning Threat Models	25
3.1.1 Dimensions of Poisoning Threat to FL	25
3.1.1.1 Adversary’s Objective	25
3.1.1.2 Adversary’s Knowledge	27
3.1.1.3 Adversary’s Capability	28
3.2 Takeaways	29
<b>4. GENERAL UNTARGETED POISONING FRAMEWORK AGAINST FEDERATED LEARNING</b>	<b>31</b>
4.1 Threat Models of Untargeted Poisoning We Use	33
4.2 Our Generic Framework for Model Poisoning	34
4.2.1 General optimization formulation	34
4.2.1.1 Introducing perturbation vectors	36
4.2.2 AGR-tailored attacks	37
4.2.2.1 Krum	37
4.2.2.2 Multi-krum	37
4.2.2.3 Trimmed-mean	38
4.2.3 AGR-agnostic attacks	38
4.2.3.1 Intuition	39
4.2.3.2 Attack-1 (Min-Max): Minimize maximum distance attack	39
4.2.3.3 Attack-2 (Min-Sum): Minimize sum of distances attack	40
4.2.4 Solving for the most effective scaling factor $\gamma$	40
4.3 Experimental setup	41
4.3.1 Datasets and model architectures	41
4.3.2 Learning and attacks settings	42
4.4 Evaluation of Our Attacks	43
4.4.1 Comparison with the state-of-the-art attacks	43



4.4.1.1	Comparing AGR-tailored attacks	43
4.4.1.2	Comparing AGR-agnostic attacks	46
4.4.2	Impact of our attacks on cross-device FL setting	48
4.5	Takeaways	50
4.5.1	Lessons learned from our attacks	51
<b>5.</b>	<b>DEFENDING FEDERATED LEARNING AGAINST POISONING ATTACKS</b>	<b>52</b>
5.1	Divide-n-Conquer (DnC)	53
5.1.1	Intuition	53
5.1.2	Our DnC algorithm	53
5.1.3	Theoretical analysis	54
5.1.4	An adaptive attack against DnC	56
5.1.5	Evaluation of Our Defense	56
5.1.5.1	Robustness of DnC for iid data	59
5.1.5.2	Robustness of DnC for non-iid data	61
5.2	Federated Rank Learning (FRL)	62
5.2.1	Preliminaries for FRL	65
5.2.1.1	Edge-popup algorithm	65
5.2.2	Details of federated rank learning (FRL)	66
5.2.2.1	Server: Initialization (only for round $t = 1$ )	68
5.2.2.2	Clients: Calculating the ranks (for each round $t$ )	69
5.2.2.3	Server: Majority vote (for each round $t$ )	70
5.2.3	Robustness of FRL to poisoning	71
5.2.3.1	Intuition behind robustness of FRL	71
5.2.3.2	The worst-case untargeted poisoning attack on FRL	71
5.2.3.3	Theoretical analysis of robustness of FRL algorithm	73
5.2.4	Robustness evaluation of FRL	74
5.2.4.1	Experimental setup	74
5.2.4.2	Robustness of FRL against untargeted poisoning	76

5.2.4.3	Robustness of FRL against targeted poisoning	78
<b>6.</b>	<b>A CRITICAL EVALUATION POISONING ATTACKS ON PRODUCTION FEDERATED LEARNING</b>	<b>82</b>
6.1	Practical Considerations for Poisoning Threat Models	84
6.1.1	Salient Features of Production Federated Learning	84
6.1.2	Understanding the practicality of threat models	85
6.1.3	Practical Ranges of FL Parameters	86
6.1.4	Threat Models in Practice	88
6.1.4.1	Nobox Offline Data Poisoning (T4)	88
6.1.4.2	Whitebox Online Model Poisoning (T5)	89
6.2	Exploring The Space of FL Poisoning Attacks	89
6.2.1	Improved FL Poisoning Attacks	89
6.2.1.1	Improved Data Poisoning Attacks (DPAs)	90
6.2.1.2	Improved Model Poisoning Attacks (MPAs)	94
6.3	Analysis of FL Robustness in Practice	96
6.3.1	Experimental setup	96
6.3.1.1	Datasets and Model Architectures	97
6.3.1.2	Details of Federated learning and attack parameters	97
6.3.1.3	Attack impact metric	98
6.3.2	Evaluating Non-robust FL (Cross-device)	98
6.3.3	Evaluating Robust FL (Cross-device)	100
6.3.3.1	Cross-device FL with robust AGRs is highly robust in practice	100
6.3.3.2	Investigating simple and efficient robustness checks is necessary	102
6.3.3.3	Thorough empirical assessment of robustness is inevitable	103
6.3.4	Effect of FL Parameters on Poisoning (Cross-device)	103
6.3.4.1	Effect of the Size of Local Poisoning Datasets ( $ D_p $ ) on DPAs.	103
6.3.4.2	Effect of the Average Dataset Size of Benign FL Clients ( $ D _{\text{avg}}$ )	106

6.3.4.3	Number of Clients Selected Per Round. . . . .	107
6.3.4.4	Effect of Unknown Global Model Architecture on DPAs. . . . .	108
6.3.5	Evaluating Robustness of Cross-silo FL . . . . .	109
6.4	Takeaways . . . . .	111
<b>7.</b>	<b>ROBUSTNESS EVALUATION OF PERSONALIZED FEDERATED LEARNING . . . . .</b>	<b>112</b>
7.1	Need for personalization in FL . . . . .	112
7.1.1	State-of-the-art PFL algorithms that we evaluate . . . . .	114
7.1.1.1	FedAvg + Fine-tuning (FedAvg-FT) . . . . .	115
7.1.1.2	Ditto . . . . .	115
7.2	A robustness evaluation personalized FL . . . . .	117
7.2.1	Why should we care about robustness of PFL? . . . . .	117
7.2.2	Methodology . . . . .	117
7.2.2.1	The backdoor attack we use . . . . .	118
7.2.2.2	Untargeted attack we use . . . . .	120
7.2.3	Experimental setup . . . . .	120
7.2.3.1	Datasets and model architectures . . . . .	120
7.2.3.2	Attack settings . . . . .	121
7.2.3.3	FL training hyperparameters . . . . .	122
7.2.3.4	Measurement metrics . . . . .	122
7.2.4	Experimental results . . . . .	123
7.2.4.1	Personalized FL improves average per-client accuracy . . . . .	123
7.2.4.2	Impact of personalization on edge-case backdoor attacks . . . . .	124
7.2.4.3	Impact of personalization on untargeted attacks . . . . .	125
7.3	Takeaways . . . . .	128
	<b>BIBLIOGRAPHY . . . . .</b>	<b>129</b>

## LIST OF TABLES

Table	Page
2.1 Comparing state-of-the-art AGRs in terms of accuracy, computation/memory cost, and theoretical guarantees. We show results for CIFAR10 with 1,000 clients. Red cells show limitations of the corresponding AGR. For brevity, we choose representative AGRs (in bold) from each type and show that they are sufficient to protect FL in practice. . . . .	17
3.1 The key dimensions of the threat models of poisoning attacks on FL. Each combination of these dimensions constitutes a threat model (Table 6.2). However, we argue in Section 6.1.2 that only two of these combinations are practical threat models. . . . .	30
4.1 Knowledge-based classification of model poisoning adversaries in FL. . . . .	34
4.2 Comparing state-of-the-art model poisoning attacks and our attacks under various threat models from Table 4.1, when cross-silo FL is used. In all the settings, the impact of our AGR-tailored attacks is significantly higher than that of AGR-tailored Fang attacks. While both of our AGR-agnostic attacks outperform AGR-agnostic LIE attacks in most cases. We assume 20% malicious clients and, except for ‘No attack’ column, report <i>the attack impact</i> $I_\theta$ (Section 4.3.2). For each adversary, we bold $I_\theta$ of the strongest attack. MMax and MSum are our Min-Max and Min-Sum attacks, respectively. . . . .	44
4.3 Comparing the state-of-the-art model poisoning attacks and our attacks under all threat models in Table 4.1 when <i>cross-device FL</i> is used. Our AGR-tailored attacks significantly outperform Fang attacks, while at least one of our AGR-agnostic attacks significantly outperforms LIE attack in most cases. Experimental setup is exactly the same as that of Table 4.2. MMax and MSum are our Min-Max and Min-Sum attacks, respectively. . . . .	49

5.1	Our robust DnC AGR defends against all the existing model poisoning attacks for independently and identically distributed datasets. We consider the adversaries with complete knowledge of gradients of benign clients with 20% malicious clients. For each attack, we report its attack impact on DnC and on the existing defense with the highest global model accuracy $A_\theta^*$ , computed as $(A_\theta - I_\theta)$ from Table 4.2. . . . .	58
5.2	Results of empirical robustness analysis of DnC for <i>cross-device FL</i> setting. We consider the adversaries with complete knowledge of gradients of benign clients with 20% malicious clients, and report $A_\theta^*$ as described in Table 4.2. . . . .	58
5.3	For non-iid FEMNIST dataset, DnC cannot mitigate our attacks in the worst case settings when the adversary knows gradients of the benign devices. But, mitigates all the attacks in the more practical settings when the gradients of benign devices are unknown. We report $I_\theta$ on DnC of all adversaries in Table 4.1 with 20% malicious clients. ‘No attack’ accuracy $A_\theta$ for FEMNIST with DnC is 86.6%. MMax and MSum refer to Min-Max and Min-Sum attacks, respectively. . . . .	58
5.4	Resampling [73] significantly reduces the robustness of existing defenses against our attacks (20% malicious clients) for all the adversaries from Table 4.1. Because, resampling increases the number of malicious updates processed, and therefore, their poisoning impacts. . . . .	62
5.5	State-of-the-art model architectures that we use for FRL experiments. . . . .	75
5.6	Comparing the robustness of various FL algorithms: FRL and Sparse-FRL (SFRL) (in <b>bold</b> ) outperform the state-of-the-art robust AGRs and SignSGD against our strong untargeted poisoning attacks from Section 4.2. . . . .	76
6.1	Practical ranges of FL parameters based on the literature and discussions on FL production systems [6, 82, 32] and the ranges used in <i>untargeted</i> FL poisoning and robust AGRs literature [64, 22, 132, 30, 108]. MPA means model poisoning attack and DPA means data poisoning attack. Red (green) cells denote impractical (practical) ranges. . . . .	84
6.2	The eight possible threat models for <i>untargeted poisoning attacks</i> on FL. T3-T8 are valid, but only T4 and T5 represent practical FL deployments (Section 6.1.4). . . . .	85

6.3	The architecture of the surrogate model that we use to emulate the unknown architecture setting (Section 6.3.4.4). . . . .	109
7.1	Personalization helps improve the overall performance of FL. . . . .	124

## LIST OF FIGURES

Figure	Page
3.1 Classes of FL poisoning attacks and their objectives defined using the taxonomy in Section 3.1.1.1: <i>Targeted</i> attacks [148, 27] aim to misclassify only a specific set/classes of inputs (e.g., certain 10 samples from CIFAR10), <i>semantic backdoor</i> attacks [18, 155] aim to misclassify inputs with specific properties (e.g., cars with stripes in background), <i>artificial backdoor</i> attacks [164] aim to misclassify inputs with an artificial (visible or invisible) trigger pattern (e.g., shape of letter "F"), and <i>untargeted</i> attacks [132, 64] aim to reduce model accuracy on <i>arbitrary</i> inputs (e.g., the entire CIFAR10 distribution). . . . .	25
4.1 <b>Schematics of our attacks:</b> (a) Our AGR-tailored attack, unlike Fang attack, fine tunes the malicious gradient ( $\nabla^b + \gamma\nabla^p$ ), using optimal $\gamma$ and dataset-optimized $\nabla^p$ . (b) Our AGR-agnostic Min-Max attack finds its malicious gradient $\nabla^m$ (red cross) whose maximum distance from any other gradient is less than the maximum distance between any two benign updates (black arrows). (c) Our AGR-agnostic Min-Sum attack finds $\nabla^m$ (red cross) whose sum of distances from the other updates is less than the sum of distances of any benign gradient from the other benign updates. Due to stricter constraints, $\nabla^m$ of Min-Sum attack is closer to the benign aggregate, $\nabla^b$ , than $\nabla^m$ of Min-Max attack. LIE attack computes very suboptimal $\nabla^m$ due to extremely small amounts of noise additions. . . . .	35
5.1 DnC selects high fractions of malicious gradients (red plots) iff the distances between $\mu_B$ and $\mu_M$ , the means of benign and malicious gradients, are low (blue plots), i.e., poisoning impact of the malicious gradients is low. Upper row is for MNIST and lower row is for CIFAR10 + Alexnet. We use the strongest full knowledge agr-updates adversary. . . . .	57

5.2	The space of client updates. Green circles represent benign updates and red triangles represent malicious updates. To defend against poisoning, existing robust AGRs filter the updates by creating a safe space (continuous $\in \mathbb{R}^d$ ). On the other hand, FRL limits the choices of clients by enforcing a discrete space of updates (a permutation of integers $\in [1, d]$ ). $\theta_g^b$ (green square) demonstrates the aggregated model for benign users, and $\theta_g^m$ (red square) demonstrates the aggregated model considering malicious updates. Black objects are updates that are ruled out by the server. . . . .	63
5.3	A single FRL round with three clients and supernetwork of 6 edges. . . . .	68
5.4	Upper bound on the failure probability of VOTE(.) function in FRL. $\alpha$ is the percentages of malicious clients and $p$ is the probability that a benign client puts a good edge in its top $k$ ranks. . . . .	74
5.5	FL backdoor poisoning attacks on CIFAR10 distributed over 1000 clients with Dirichlet ( $\beta = 1.0$ ) for presence pf adversary in 1000 FL rounds. . . . .	80
6.1	Effect of varying the sizes of poisoned data, $D_p$ , on the objectives of DPAs (Section 6.2.1.1) on various AGRs. We compute $D_p$ by flipping the labels of benign data. . . . .	91
6.2	Schematic of our PGA attack: PGA first computes a <i>poisoned update</i> $\nabla'$ using stochastic gradient ascent (SGA). Then, $f_{\text{project}}$ finds the scaling factor $\gamma$ that maximizes the deviation between benign aggregate $\nabla^b$ and poisoned aggregate $\nabla_\gamma^p$ . Robust aggregations easily discard the scaled poisoned updates, $\gamma\nabla'$ , with very high $\gamma$ (e.g., $\gamma_{\{4,5\}}$ ), while those with very small $\gamma$ (e.g., $\gamma_{\{1,2\}}$ ) have no impact. . . . .	93
6.3	Attack impacts ( $I_\theta$ ) of state-of-the-art data (DPA-DLF/SLF) and model (MPA) poisoning attacks on cross-device FL with average AGR. $I_\theta$ 's are <i>significantly lower for practical percentages of compromised clients (<math>\leq 0.1\%</math>) than previously thought.</i> . . . . .	98
6.4	Even with a very large number of FL rounds (5,000), the state-of-the-art model poisoning attacks with $M=0.1\%$ cannot break the robust AGRs (Section 6.3.3). . . . .	99
6.5	FEMNIST with CNN architecture . . . . .	101
6.6	CIFAR10 with VGG9 architecture . . . . .	101



6.7	Purchase with Fully connected architecture .....	101
6.8	Attack impacts of various poisoning attacks (Section 6.2) on cross-device FL with robust AGRs (Section 2.2.3). <i>These AGRs are highly robust for practical percentages of compromised clients; for in-depth analysis, please check Section 6.3.3.</i> .....	101
6.9	Effect of varying sizes of local poisoned dataset $D_p$ on impacts $I_\theta$ of the best of DPAs. When $ D_p $ and $M$ are in practical ranges, $I_\theta$ 's are negligible for robust AGRs and are dataset dependent for non-robust Average AGR. ....	104
6.10	With 1% compromised clients, <i>increasing <math> D _{\text{avg}}</math> has no clear pattern of effects of on attack impacts, but it increases the global model accuracy</i> .....	105
6.11	As discussed in Section 6.3.4.3, the number of clients, $n$ , chosen in each FL round has no noticeable effect on the attack impacts, with the exception of model poisoning on Average AGR. We use $M = 1\%$ of compromised clients. ....	107
6.12	Impacts of the DPA-DLF (Section 6.2.1.1) attack, which uses the knowledge of model architecture, reduce if the architecture is unknown.....	108
6.13	All data poisoning attacks have negligible impacts on cross-silo FL, when compromised clients are concentrated in a few silos or distributed uniformly across silos (Section 6.3.5). ....	110
7.1	Conventional FL algorithms, e.g., FedAvg, compute a single global model for all clients. As we can see in the two plots here, such single model cannot provide good performance for all the clients. In fact, for many clients (as in the case of Reddit) the accuracy of the model trained only on the local data is more than the accuracy of the global model on their local test data. This implies participating in FL does not benefit these clients. ....	113
7.2	Samples from the tail of the distribution of CIFAR10 (a) and FEMNIST (b) datasets. (a) Images of Southwest airline planes which are not in the original CIFAR10 data although it has multiple plane images (b) Images of digit '7' from ARDIS dataset [93] written in a different style than '7' in the original FEMNIST data.....	119

7.3	Impact of varying the % of total edge data (660 samples in our case) that the adversary holds; benign clients hold rest of the data. Edge-case backdoor attacks are very effective against FedAvg, but personalized FL algorithms are very effective in mitigating their threat to FL. For cross-device FL here, adversary participates every 10 <sup>th</sup> round starting from round 1,000. We use FedAvg + Norm-bounding here. . . . .	124
7.4	Impact of attack frequency. . . . .	125
7.5	Cross-device FL + local fine-tuning . . . . .	126
7.6	Cross-silo FL + local fine-tuning . . . . .	126
7.7	Cross-silo FL + Ditto . . . . .	126
7.8	Impact of our state-of-the-art untargeted poisoning attacks (Chapter 4) on PFL algorithms. We make the following observations. First, under production FL settings (e.g., with practical percentages of malicious clients), PFL algorithms remain highly robust. However, personalization algorithms fail to salvage FL against stronger adversary, e.g., with higher percentages of malicious clients. . . . .	126

# CHAPTER 1

## INTRODUCTION

*“If a machine is expected to be infallible, it cannot also be intelligent.”*

Alan Mathison Turing

Machine learning (ML) has made a tremendous progress over the past decade and is the driving force of numerous modern applications, e.g., object detection in self-driving cars [90], next word prediction on digital keyboards [104, 121], medical diagnosis based on patients’ data [85], captions generation in YouTube videos, etc. It is well known that these ML models are data hungry and the more the data they are trained on the better they perform. Such large amounts of data are generally collected directly (e.g., patients’ data) or indirectly (e.g., from Internet) from a large number of users (to increase data quantity) with diverse backgrounds (to improve data quality). However, increasing amounts of training data raises significant concerns about the privacy of users who contribute to the data.

The training data of ML models can be of sensitive nature as it belongs to real users, e.g., the training data of next word prediction models of digital keyboard applications may contain sensitive information about the data owner, while the clinical data of a patient used to train a medical diagnosis model generally contains highly sensitive information about the mental or physical health of the patient (i.e., data owner). Numerous recent researches have demonstrated how various ML models leak sensitive information [138, 143, 146, 145, 144, 142, 40, 41, 135, 117, 152, 131, 133, 40, 39, 71, 19, 142, 107, 144, 50, 31] about their training data, which has raised concerns about the privacy of users who contribute data to train ML models.

To address the numerous concerns around privacy, research community has been putting significant efforts to design privacy preserving ML algorithms. A major effort towards privacy preserving ML is to develop *differentially private ML* algorithms [10, 34, 105, 106, 130, 92, 23, 62, 120, 77, 44, 141, 79, 16] that satisfy differential privacy (DP) [60]. However, majority of the DP ML algorithms consider the *centralized ML* [115, 26, 46, 10] setting, where a set of users share their private data with a central ML service provider (e.g., Amazon AWS or Google Cloud) and then the service provider trains a ML model on the entire data. Unfortunately, in such centralized ML setting, due to sharing of data with the (honest but curious or potentially malicious) service provider, users have no privacy from the service provider; note that this applies even when the ML algorithms are differentially private. Furthermore, any other privacy preserving ML solutions [116, 131, 133, 150] in centralized ML setting expose the sensitive information of the users contributing to the training data to the ML service provider.

*Federated learning (FL)* [104, 137, 33, 32, 81, 11, 20, 94, 128, 119, 166, 102, 174, 99, 149, 30, 168, 67, 14, 157, 101], another major effort towards privacy preserving ML, aims to alleviate the aforementioned privacy issues due to sharing of private data with untrusted third parties. For training ML models, FL service provider (called the *server*) does not use the private data of the collaborating/participating users (called the *clients*) in plaintext form, and instead uses some function of their data, e.g., gradients [137], model updates [104] or model outputs [42, 101]. In a typical FL setting, the server repeatedly collects some updates that the clients compute using their local private data, aggregates the clients' updates using an *aggregation rule (AGR)*, and finally uses the aggregated client updates to tune the jointly trained model (called the *global model*), which is broadcasted to a subset of the clients at the end of each FL training round. FL is increasingly adopted by various distributed platforms, in

particular by Google’s Gboard [3] for next word prediction, by Apple’s Siri [121] for automatic speech recognition, and by WeBank [5] for credit risk predictions.

Although FL is a promising approach to alleviate many concerns around user data privacy, there are numerous challenges to adopting FL in real-world applications, e.g., designing provably private FL algorithms [81, 20, 103, 17, 33, 80, 47], designing FL algorithms that treat all clients fairly [74, 35, 100, 156, 49], designing communication efficient FL algorithms [14, 15, 157, 25], designing FL algorithms robust to various adversarial circumstances [171, 122, 84, 83, 64, 22, 30, 132, 108, 156], etc. There are dire consequences of poisoning on the performance of FL (as we detail in the next paragraph), yet relatively fewer works have tried to investigate the robustness aspect of FL. To fill this gap, *in this thesis, we focus on thoroughly quantifying and enhancing the robustness of FL against poisoning attacks.*

**The threat of poisoning to FL:** A key feature that makes FL highly attractive in practice is that it allows training models via collaboration between *mutually un-trusting* clients, e.g., Android users or competing banks. Unfortunately, this makes FL susceptible to a threat known as *poisoning*: a small fraction of FL clients, called *malicious clients*, who are either owned or controlled by an adversary, may act maliciously during the FL training process in order to corrupt the jointly trained global model. Specifically, the goal of the poisoning adversary is to attack FL by instructing its malicious clients to contribute *poisoned* model updates during FL training in order to *poison* the global model.

There are three major approaches to poisoning FL: *Targeted* attacks [148, 27] aim to misclassify only a specific set/classes of inputs (e.g., certain 10 samples from CIFAR10), *semantic backdoor* attacks [18, 155] aim to misclassify inputs with specific properties (e.g., cars with stripes in background), *artificial backdoor* attacks [164] aim to misclassify inputs with an artificial (visible or invisible) trigger pattern (e.g., shape of letter “F”), and *untargeted* attacks [132, 64] aim to reduce model accuracy

on *arbitrary* inputs (e.g., the entire CIFAR10 distribution). Figure 3.1 in Chapter 3 provides a visual illustration of these attacks and their differences.

Each of the poisoning types significantly impact the real users’ experience of FL in specific settings. For instance, targeted [27] and backdoor [155] attacks can make the FL model completely useless for a subset of minority FL population, some backdoor attacks [18] may allow unauthorized access to FL applications, while untargeted poisoning can reduce the utility of FL for a large fraction of FL population.

**Overview of our contributions:** In this thesis, we first provide a comprehensive systemization of poisoning adversaries in FL and provide a general framework to design poisoning attacks in various adversarial settings. Using these attacks we show that many of existing state-of-the-art defenses fall short of their theoretical robustness guarantees and also fail in the adversarial settings where they claimed to be robust. Based on lessons from our evaluations, we design two novel defenses against FL poisoning. For the above contributions, we considered very strong (theoretical) adversaries that helped us assess the robustness guarantees of the FL poisoning defenses, but such adversaries may not be very relevant in practical FL deployments. Therefore, in the last part of the thesis, we analyze robustness of (robust) FL algorithms under practical adversaries in production FL settings and make significant conclusions about FL robustness in practice.

## 1.1 Gaps in FL robustness literature and thesis contributions

In this section, we discuss in detail the gaps in the literature on FL robustness against poisoning and contributions of this thesis towards overcoming the gaps. Along the discussion, we detail the structure of the thesis.

### 1.1.1 No systemization of poisoning attacks against FL

Since the inception of FL [104], numerous works [30, 108, 64, 22, 168, 155, 18, 27, 165, 132, 98] have proposed various poisoning attacks against FL under various adversarial settings, and consequently many works have also proposed defenses to mitigate these poisoning attacks. For instance, Sun et al. [147] introduce an attack that is only applicable to multi-task learning based FL, while Cao et al. [38] introduce a defense that is only applicable in cross-silo FL settings. Furthermore, different works make different assumptions about the FL setting and adversary they consider while designing their attacks/defenses. But there is no comprehensive taxonomy of attacks or threat models that can help practitioners compare these various attacks/defenses and to understand the suitability of certain threat models in the settings of their interest.

In Chapter 3, we start by establishing a comprehensive systemization of threat models of FL poisoning. Specifically, we discuss three key dimensions of the poisoning threat to FL: the adversary’s objective, knowledge, and capability. We also provide a wide spectrum of values for each of these dimensions, e.g., for knowledge, adversary can have the knowledge of the global model and/or the knowledge of the data from benign distribution. We detail the significance of each value of each of the dimension.

Any combination of these dimensions constitutes a threat model, but not all threat models are valid threat models. For instance consider a *model poisoning adversary* who directly manipulates the malicious model updates that they share with the server; to achieve this, the adversary should have significant knowledge of about the security protocols of the device running the FL application, and must break into the device to be able to access the on-device FL model and updates. This is a challenging task, but once done, the adversary has complete undeterred access to the device. Also consider *offline adversary*, who poisons the model only once. However, note combining model poisoning attacks with offline attacks will be wasting the efforts

that model poisoning adversary put into breaking into the FL application device. Therefore, offline model poisoning adversary is not a valid threat model. In Chapter 6, we discuss the practicality of all possible threat models obtained by combining these dimensions.

### 1.1.2 No general framework to design FL poisoning attacks

As mentioned before, literature has introduced numerous poisoning attacks against FL, and consequently, has also introduced a number of defenses [30, 108, 168, 68, 123, 58, 42, 99] to mitigate FL poisoning. Naturally, these defense works use the attacks from the FL poisoning literature to assess the robustness of their proposed defenses. However, we observe that many of these attacks are extremely sub-optimal and perform poorly in the adversarial setting (also called, *threat model*) that they consider. For instance, Baruch et al. [22] propose a model poisoning attack where they assume complete access to the global model parameters and the local datasets of all FL clients, in all FL rounds. This is a very strong threat model, but their attack adds a very conservative amounts of Gaussian noise (they call their attack “Little-is-enough” attack) which does not perform very well against FL unless the percentages of malicious clients is impractically high.

An unfortunate consequence of using such sub-optimal attacks for robustness assessment is that FL poisoning defenses may incorrectly claim to be robust against FL poisoning. Furthermore, this may lead to FL practitioners using these defenses to design FL applications that are vulnerable to poisoning.

To this end, in Chapter 4, we provide a general framework to design *untargeted* poisoning attacks against FL. In this thesis, we choose to focus on the untargeted poisoning attacks for two major reasons: (1) it is highly relevant to production FL deployments, as it can be used to impact a large population of FL clients and it can remain undetected for long duration, and (2) majority of the poisoning defenses aim



to defend FL against untargeted poisoning attacks; furthermore, we also discuss the importance of untargeted poisoning in real-world FL systems (Section 3.1.1.1.1).

We consider the strong *whitebox online model poisoning* adversary, who can access and directly manipulate their model updates in all FL rounds that they participate in. However, unlike prior works [64, 22, 108, 165], we consider a comprehensive set of possible threat models by further varying the adversary’s knowledge along two dimensions: the knowledge of the updates of benign clients and the knowledge of the poisoning defense that the server uses. The high-level approach of our attack is as follows. The adversary computes a benign reference aggregate using some benign updates they have; then they compute a malicious perturbation (whose generation we explain in Section 4.2), e.g., a unit vector in the opposite direction of the benign aggregate. Finally, the adversary computes their malicious model update by maximally perturbing the benign reference aggregate in the malicious direction, while also evading detection by robust aggregation algorithms.

Our work is first to design strong poisoning attacks that *do not* use any knowledge about server’s defense (Section 4.2.3). Through extensive evaluations (Section 4.4) on four datasets, seven defenses and two types of FL settings, we demonstrate that the poisoning attacks launched using our framework outperform state-of-the-art model poisoning attacks against major, state-of-the-art FL poisoning defenses. Our attacks lead to significant conclusions, e.g., **(1)** many of state-of-the-art FL defenses are far more susceptible to poisoning than what their theoretical robustness guarantees claim, **(2)** poisoning adversary can mount a strong poisoning attack even without the knowledge of the server’s defense.

Finally, in Section 4.5, we provide concrete lessons we learn from our evaluations that we hope will guide future designs of robust AGRs.

### 1.1.3 Sub-optimal defenses against FL poisoning

Our general poisoning attacks framework from Chapter 4 highlights the brittle nature of seven of the major state-of-the-art FL poisoning defenses and highlights the common shortcomings of the designs of prior defenses. For instance, as one of the lessons in Section 4.5 highlights, for non-convex FL settings, many defenses provide provable theoretical robustness guarantees in terms of the convergence of global model. However, in non-convex settings, such guarantees are not very useful due to a large number of sub-optimal local minima. This and other shortcomings of prior defenses clearly motivates the need to design more principled and strong defenses against FL poisoning.

To address this, in Chapter 5, we design two principled defenses: *Divide-n-conquer (DnC)* and *Federated rank learning (FRL)*. The design of both of these defenses is motivated from our lessons from Section 4.5. For instance, most prior defenses suffer from the *curse of dimensionality*: the theoretical error bounds of prior defenses [30, 108, 168, 59, 95, 13] depend on the dimensionality of their inputs. Hence, the theoretical as well as empirical errors of these defenses explode for high dimensional gradients of neural networks [42] in FL. To address this issue, DnC uses a random subnetwork of client updates to effectively filter out malicious updates, while in FRL, clients compute optimal subnetwork of the received global model using their local data and share these subnetworks (which are as small as 10% of the size of global model) with the server. Furthermore, to address the aforementioned issue of convergence based robustness guarantee, DnC provides theoretical guarantee on the removal of malicious clients (Section 5.1.3), while FRL provides theoretical guarantee on the optimality of the subnetwork (Section 5.2.3.3) it computes.

We perform extensive empirical analyses of our defenses and show that DnC and FRL can defend FL against state-of-the-art model poisoning attacks. We evaluate both the defenses against a very strong adversary with full knowledge of the global

model, benign data, and the defense algorithm; for this evaluation, we designed strong adaptive attacks tailored to the defenses. We note that under extremely heterogeneous settings, DnC fails to defend FL, but FRL, due the use of optimal subnetworks, successfully defends FL from poisoning.

#### 1.1.4 Poor understanding of the impacts of poisoning on real-world FL

As repeatedly mentioned above, a significant amount of work has investigated the robustness of FL against untargeted poisoning from the lens of attacks and defenses. The defense works generally aim to defend FL against a very strong, and sometimes even hypothetical, poisoning adversary with significant knowledge and capabilities, and consequently, attack works also claim the success of their attack assuming a similarly strong adversary. However none of the prior works evaluates the practicality of such strong assumptions about the poisoning adversary. Consequently, the current literature lacks the understanding of impact of poisoning on real-world, *production FL systems* [82, 32].

To this end, in Chapter 6, first, we perform a thorough investigation of practicality of all the threat models that our systemization from Chapter 3 can build. We show that, out of all possible combinations, only two threat models, i.e., *nobox offline data poisoning* and *whitebox online model poisoning*, are of practical value to production FL. We believe that prior works [22, 30, 64, 132] have neglected the crucial constraints of production FL systems on the parameters relevant to FL robustness. *Our work is the first to consider production FL environments* [82, 32] and provide practical ranges for various parameters of poisoning threat models. As a result, *our evaluations lead to conclusions that contradict the common beliefs in the literature*, e.g., we show that production FL even with the non-robust Average AGR is significantly more robust than previously thought.

#### 1.1.4.1 Evaluating traditional FL

Next, we design improved data and model poisoning attacks for the two aforementioned threat models. We present *the first* attacks that systematically consider the data poisoning threat model for FL (Section 6.2.1.1). We propose novel model poisoning attacks that outperform the state-of-the-art. Our model poisoning attacks (Section 6.2.1.2) use *gradient ascent* to fine-tune the global model and increase its loss on benign data.

Next, we extensively evaluate all existing poisoning attacks as well as our own improved attacks across three benchmark datasets, for various FL parameters, and for different types of FL deployments. We make several significant deductions about the state of FL poisoning literature for production FL. For production cross-device FL, which contains thousands to billions of clients, we provide four key lessons. For instance, our evaluation shows that, *enforcing a limit on the size of the dataset contributed by each client can act as a highly effective (yet simple) defense against data poisoning attacks* with no need to any of the state-of-the-art, sophisticated robust FL aggregation algorithms.

On the other hand, for production cross-silo FL, which contains up to hundred clients [82], we show that *data poisoning attacks are completely ineffective, even against non-robust Average AGR*. We also argue that model poisoning attacks are unlikely to play a major risk to production cross-silo FL, where the clients involved are bound by contract and their software stacks are professionally maintained (e.g., in banks, hospitals, etc.).

One of the major implications of our study in Chapter 6 is as follows. Numerous recent works have proposed sophisticated aggregation rules for FL with strong theoretical robustness guarantees [30, 108, 168, 13, 165, 123, 54, 61]. However, our work shows that, when it comes to production FL deployments, even simple, low-cost defenses can effectively protect FL against poisoning. We also believe that our sys-

tematization of practical poisoning threat models can steer the community towards practically significant research problems in FL robustness.

#### 1.1.4.2 Evaluating personalized FL

The key reason behind a major failure of traditional FL, i.e., fair treatment of entire FL population, is that the traditional FL tries to cater to the heterogeneous needs of the clients using a single global model. To alleviate this issue, *personalized FL* algorithms aim to compute one model for each of the FL clients, that is tailored to the client’s local data distribution. Multiple works have argued that personalization is inevitable in FL [149, 99, 102, 82], as it improves the overall performance of FL as well as mitigates the issues of robustness and fairness to some extent.

In spite of its popularity, the robustness of personalized FL is unclear. Therefore, in Chapter 7, we evaluate the robustness of two state-of-the-art personalized FL algorithms, *FedAvg + local fine-tuning* [169] and *Ditto* [99], against untargeted as well as backdoor poisoning. We consider the stronger whitebox online model poisoning threat model for the evaluations. The key observations from our evaluations are: (1) For backdoor poisoning: the personalized FL algorithms can completely neutralize the threat of backdoor attacks even under (impractically) strong threat models, e.g., accuracy of the backdoor reduces from 100% in poisoned global model to close to 0% after a few steps of local fine-tuning. (2) For untargeted poisoning: under production FL environments, personalized FL remains robust, however as expected, under impractical threat models, e.g., with a very large number of malicious clients, personalized FL is not robust against our state-of-the-art untargeted attacks from Chapter 4.

## 1.2 Conclusions and future works

In this thesis, we thoroughly evaluate federated learning (FL), an emerging privacy preserving ML solution, from the lens of robustness against poisoning attacks and provide defenses to mitigate these attacks. We make many significant observations and conclusions about the state of robustness of FL that contradict the prior beliefs of the scientific community. First, we use our general framework for designing poisoning attacks to demonstrate that the theoretical guarantees and empirical claims of robustness of many of existing state-of-the-art FL poisoning defenses do not hold and provide concrete lessons that will aid future works to design more principled defenses. Our study of the impact of poisoning on current real-world production FL settings paints a very different picture of FL robustness. In particular, using our comprehensive systemization of FL poisoning adversaries, we argue that in a specific FL deployment, called cross-silo FL, certain adversaries that prior work commonly uses, cannot exist. We also show that although numerous recent works have proposed sophisticated aggregation rules for FL with strong theoretical robustness guarantees, even simple, low-cost defenses, e.g., bounding the size of clients' local data, can effectively protect FL against poisoning.

FL is a very complex system and FL algorithms should fulfil multiple design goals, including privacy, fairness, robustness, communication, to make FL useful in practice. Some of these goals are contradictory to each other, e.g, using robust FL algorithms produce models that are unfair to the minorities in FL population and vice-versa. Hence an important direction for future work is to design FL algorithms that can strike appropriate balance between these two goals. Similarly, in many FL settings, FL applications run on resource-constrained devices and many benign client updates may never reach the server, while adversary can ensure that the malicious updates always reach the server thereby increasing the strength of poisoning. It

is critical to understand whether existing FL defenses can handle such practically relevant adversarial setting, and if not, how can we design novel defenses.

## CHAPTER 2

### BACKGROUND AND RELATED WORKS

In this chapter, we discuss some of the preliminaries required to understand the rest of the thesis. Specifically, we give (i) a technical background of federated learning (FL), (ii) introduce the issue of poisoning attacks in FL and discuss prior state-of-the-art poisoning attacks, (iii) discuss prior state-of-the-art defenses designed to mitigate poisoning attacks in FL. Along the way, we also discuss any other existing works related to the above aspects of FL robustness.

#### 2.1 Federated Learning

Section 1 gives a high level picture of federated learning (FL). In FL [82, 104, 86], a service provider, called *server*, trains a *global model*,  $\theta^g$ , on the private data of multiple collaborating clients without directly collecting their data. In the  $t^{\text{th}}$  FL round, the server selects  $n$  out of total  $N$  clients and shares the most recent global model, i.e.,  $\theta_g^t$ , with them. Then, a client  $k$  uses her local data  $D_k$  to fine-tune  $\theta_g^t$  using stochastic gradient descent (SGD) for a fixed number of local epochs  $E$ ; we denote the resulting updated model by  $\theta_k^t$ . Then, the  $k^{\text{th}}$  client computes her FL *update* as the difference  $\nabla_k^t = \theta_k^t - \theta_g^t$  and shares  $\nabla_k^t$  with the server. The server then computes an aggregate of all client updates using some aggregation rule,  $f_{\text{agg}}$ , i.e., using  $\nabla_{\text{agg}}^t = f_{\text{agr}}(\nabla_{\{k \in [n]\}}^t)$ . Then, the server updates the global model of the  $(t+1)^{\text{th}}$  round using SGD as  $\theta_g^{t+1} \leftarrow \theta_g^t + \eta \nabla_{\text{agg}}^t$ ; here  $\eta$  is the server's learning rate.

FL can be either *cross-device* or *cross-silo* [82]. Each of these types have certain salient features, e.g., in cross-device FL,  $N$  is large (from few thousands to billions)



and only a small fraction of them is chosen in each FL training round, i.e.,  $n \ll N$ . While, in cross-silo FL,  $N$  is moderate (up to 100) and all of them are chosen in each round, i.e.,  $n = N$ . Please refer to Table 1 of the comprehensive survey by Kairouz et al. [82] for more details on production FL environments.

## 2.2 Poisoning Attacks on Federated Learning

As briefly discussed in Chapter 1, FL is a distributed learning system with mutually untrusting clients, e.g., Android users or competing banks. Some of these clients, called *malicious* clients, who are either owned or controlled by an adversary, may act maliciously during the FL training process in order to corrupt the jointly trained global model. Specifically, the goal of the poisoning adversary is to attack FL by instructing its compromised clients to contribute poisoned model updates during FL training in order to poison the global model.

We classify the approaches to poisoning FL in three categories: *targeted* [27, 151, 148] attacks aim to reduce the utility of the global FL model on specific test inputs of adversary’s choice; *untargeted* [64, 132, 22] attacks aim to reduce the utility of global model on arbitrary test inputs; and *backdoor* [18, 155, 164] attacks aim to reduce the utility on test inputs that contain a specific signal called the trigger. In this thesis, *we focus on untargeted poisoning attacks*, because we find them to be significantly relevant to production deployments: it can be used to impact a large population of FL clients and it can remain undetected for long duration. In this thesis, we focus on untargeted poisoning as clarified in Chapter 1. Next, we first provide a brief overview of existing untargeted poisoning attacks on FL, followed by defenses to mitigate these attacks.

### 2.2.1 Existing Untargeted FL Poisoning Attacks

Recent works have presented various techniques to poison FL [64, 22, 132]. The core idea behind these poisoning attacks is to generate poisoned updates (either by direct manipulation of model updates, called *model poisoning* [18, 132, 64, 22, 27], or through fabricating poisoned data, called *data poisoning* [151, 155]) that deviate maximally from a benign direction, e.g., the average of benign clients’ updates, and at the same time *circumvent* the given robust AGR, i.e., by bypassing its detection criteria.

**Data Poisoning Attacks (DPAs):** DPAs have been studied mainly in the context of centralized ML [162, 161, 167, 114, 48], and no prior work has studied untargeted DPAs that are tailored to FL settings.

Label flipping (LF) However, the simplest form of data poisoning is label flipping, where each malicious client flips the labels of their local data from true label  $l$  to false label  $C - 1 - l$ , where  $C$  is the number of classes. A number of works have used this attack to assess the robustness of FL defenses [99, 64, 136].

**Model Poisoning Attacks (MPAs):** Multiple works have proposed MPAs on FL [64, 22, 132]. They consider our whitebox online model poisoning threat model (T4) from Section 6.1.4.2, but use unrealistic FL parameter values, e.g., very high percentages of compromised clients.

Little Is Enough (LIE) attack [22] adds small amounts of noise to each dimension of the average of the benign updates. Specifically, the adversary computes the average ( $\nabla^b$ ) and the standard deviation ( $\sigma$ ) of the available benign updates; then computes a coefficient  $z$  based on the number of benign and compromised clients; and finally computes the poisoned update as  $\nabla' = \nabla^b + z\sigma$ . [22] shows that such noises easily evade the detection by robust AGRs as well as effectively poison the global model.

Static Optimization (STAT-OPT) attack [64] proposes a general FL poisoning framework and then tailors it to specific AGRs. STAT-OPT computes the average ( $\nabla^b$ )

Table 2.1: Comparing state-of-the-art AGRs in terms of accuracy, computation/memory cost, and theoretical guarantees. We show results for CIFAR10 with 1,000 clients. Red cells show limitations of the corresponding AGR. For brevity, we choose representative AGRs (in bold) from each type and show that they are sufficient to protect FL in practice.

Type of aggregation rule (AGR)	AGR	Accuracy in non-iid FL	Compute cost to server	Memory cost to client	Theoretical robustness based on
Non-robust	<b>Average</b> [104]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	None
Dimension-wise filtering	Median [168]	84.2	$\mathcal{O}(dn \log n)$	$\mathcal{O}(d)$	convergence
	<b>Trimmed-mean</b> [168]	86.6	$\mathcal{O}(dn \log n)$		convergence
	Sign-SGD + majority voting [25]	35.1	$\mathcal{O}(d)$		convergence
Vector-wise scaling	<b>Norm-bound</b> [148]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	None
Vector-wise filtering	Krum [30]	46.9	$\mathcal{O}(dn^2)$	$\mathcal{O}(d)$	convergence
	<b>Multi-krum</b> [30]	86.2	$\mathcal{O}(dn^2)$		convergence
	Bulyan [108]	81.1	$\mathcal{O}(dn^2)$		convergence
	RFA [123]	84.6	$\mathcal{O}(dn^2)$		convergence
	RSA [98]	35.6	$\mathcal{O}(d)$		convergence
	DnC [132]	86.1	$\mathcal{O}(d)$		filtering
Certification	Emsemble [37]	74.2	$\mathcal{O}(d)$	$\mathcal{O}(Md)$	Certification
	CRFL [163]	64.1	$\mathcal{O}(d)$		Certification
Knowledge transfer	Cronus [42]	Needs public data	$\mathcal{O}(d)$	$\mathcal{O}(d)$	filtering
Personalization	Ditto [99]	86.6	$\mathcal{O}(d)$	$\mathcal{O}(d)$	None
	EWC [169]				
Singular value decomposition	Sever [57], etc.	Prohibitive computation	$\mathcal{O}(d^3)$	$\mathcal{O}(d)$	filtering
Encoding	Draco [45], etc.	Designed for settings with complete control over all the data, e.g., data centers			filtering

of the available benign updates and computes a *static malicious direction*,  $\omega = -\text{sign}(\nabla^b)$ ; the final poisoned update,  $\nabla'$ , is  $-\gamma\omega$  and the attack finds a suboptimal  $\gamma$  that circumvents the target AGR; for details please refer to [64]. Unlike LIE, STAT-OPT attacks carefully tailor themselves to the target AGR, and hence, perform better.

*Gaussian attacks* Xie et al. [165] introduce these simple model poisoning attacks, where the adversary sends arbitrary updates that are Gaussian random vectors with zero mean and isotropic covariance matrix with large standard deviations. We do not consider these (and several other attacks that use arbitrary vectors drawn from some probability distribution) attacks in this thesis as they are very suboptimal and are not useful to understand the true robustness of FL.

### 2.2.2 Existing Defenses Against Untargeted Poisoning

To protect FL against such poisoning attacks, the literature has designed various *robust aggregation rules (AGRs)* [30, 168, 42, 108, 64, 132], which aim to remove or attenuate the updates that are more likely to be malicious according to some criterion. For instance, Multi-krum [30] repeatedly removes updates that are far from the geometric median of all the updates. Below we introduce the *types* of robust AGRs designed to defend untargeted poisoning attacks. In Section 2.2.3, we provide details of the robust AGRs that we study in this thesis.

Dimension-wise filtering techniques separately filter potentially malicious values for each dimension of clients' updates. Example systems are Median [168], Trimmed-mean [168], and sign-SGD with majority voting [25].

Dimension-wise filtering techniques separately filter potentially malicious values for each dimension of clients' updates. Example systems are Median [168], Trimmed-mean [168], and sign-SGD with majority voting [25].

Vector-wise filtering defenses aim at removing potentially poisoned client updates. They differ from dimension-wise filtering, as they attempt to remove entire malicious updates, as opposed to removing malicious values. Example techniques include RFA [123], RSA [98], Krum [30], Multi-krum [30], Bulyan [108], and Divide-and-conquer (DnC) [132].

Vector-wise scaling defenses, e.g., Norm-bounding [148], aim to reduce the impact of malicious updates by scaling their norms.

Certified defenses [37, 163] provide certified accuracy for each test sample when the number of malicious clients or perturbation to the test sample is below a certified threshold.

Knowledge transfer based defenses [42, 101] aim to reduce the dimensionality of the client updates, because theoretical robustness guaranty of most of robust AGRs is directly proportional to updates' dimensionality. Hence, they use knowledge transfer

and, instead of sharing parameters of client models, share predictions of client models on some public data.

Personalization techniques, e.g. Ditto [99] and EWC [169], fine-tune the potentially corrupt global model on each client’s private data to improve its performance for the client.

Singular value decomposition based defenses [59, 58, 57, 95] filter poisoned updates to robustly estimate the mean of input updates. Their computational complexity is  $\mathcal{O}(d^3)$ , where  $d$  is number of parameters in the model. These defenses are prohibitively expensive in production FL, because it uses models with millions of parameters. Hence, we omit them from our evaluations.

Encoding based defenses [125, 45, 70, 87, 55] use artificial redundancy, i.e., duplication of client updates, to recover the failures due to compromised clients. These defenses are designed for data-center environments, hence we omit them from our evaluations.

### 2.2.3 Robust AGRs Studied in This Thesis

There are more than 100 robust AGRs introduced in the literature that provide robustness to FL in specific settings. For brevity, we study the state-of-the-art AGRs compatible with general FL settings (i.e., do not need specific FL algorithm, e.g., clustering). Below, we detail the robust AGRs that we study in this thesis.

**Norm-bounding** Sun et al. [148] were the first to use Norm-bounding to defend FL against poisoning. Norm-bounding bounds the  $L_2$  norm of all submitted client updates to a fixed threshold, with the intuition that the effective poisoned updates should have high norms. For a threshold  $\tau$  and an update  $\nabla$ , if the norm,  $\|\nabla\|_2 > \tau$ ,  $\nabla$  is scaled by  $\frac{\tau}{\|\nabla\|_2}$ , otherwise the update is not changed. The final aggregate is an average of all the updates, scaled or otherwise.

**Krum** Blanchard et al. [30] propose Krum AGR based on the intuition that the malicious gradients need to be far from the benign gradients in order to poison the

global model. Hence, Krum selects the gradient from the set of its input gradients that is closest to its  $n - m - 2$  neighboring gradients in the squared Euclidean norm space; here,  $m$  is an upper bound on the number malicious clients in FL.

**Multi-krum** Blanchard et al. [30] modify Krum AGR to Multi-krum AGR in order to effectively utilize the knowledge shared by the clients in each FL epoch [30]. Multi-Krum selects a gradient using Krum from a remaining-set (initialized to the set of all the received gradients), adds it to a selection-set (initialized to an empty set), and removes it from the remaining-set. This way, Multi-krum selects  $c$  gradients such that  $n - c > 2m + 2$ . Finally, Multi-krum averages the gradients in the selection-set. Multi-krum significantly outperforms Krum in terms of the global model accuracy.

**Bulyan** Mhamdi et al. [108] show that a malicious gradient can remain close to benign gradients while having a *single* gradient dimension with a very large value (on order of  $\Omega(\sqrt[p]{d})$ ) and prevent convergence of the global model. As a remedy, they propose Bulyan AGR, which requires  $n \geq 4m + 3$  for its robustness guarantees to hold. Bulyan first selects  $\theta(\theta \leq n - 2m)$  gradients in the same fashion as Multi-krum, and then computes Trimmed-mean of the selected gradients; please refer to [108] for more details.

**Trimmed-mean** Trimmed-mean [168, 165] is a coordinate-wise AGR which aggregates each dimension of input gradients separately. Specifically, for a given dimension  $j$ , it sorts the values of  $j^{\text{th}}$ -dimension of all gradients, i.e., sorts  $\nabla_{\{i \in [n]\}}^j$ . Then it removes  $\beta$  largest and smallest values and computes average of the rest of the values as its aggregate of dimension  $j$ . We use Trimmed-mean where  $\beta$  equals  $m$ , the number of malicious clients. Yin et al. [168] show that Trimmed-mean achieves order-optimal error rates when  $m \leq \beta \leq \frac{n}{2}$  for strongly convex objective function.

**Median** Median [168, 165] is an another coordinate-wise AGR which aggregates its input gradients by computing median of the values of each of the dimensions of the gradients. Yin et al. [168] give theoretical guarantees on the robustness of Median

AGR, while Fang et al. [64] empirically show that Median AGR has better robustness than the more sophisticated Krum AGR.

**Adaptive federated average (AFA)** AFA [113] removes malicious gradients based on their cosine-similarities with a benign gradient. More specifically, in each FL round, AFA computes a weighted average of collected gradients and computes cosine similarities between the weighted average and each of the collected gradients. Then, AFA discards the gradients whose similarities are out of a range; this range is a simple function of mean, median and standard deviation of the similarities.

**Fang defenses** Fang et al. [64] propose defenses that are meta-AGRs and rely on existing robust AGRs to detect malicious gradients. More specifically, consider a robust AGR  $A$ . Given a set of gradients  $G$ , the corresponding Fang defense, called *Fang-A*, computes a score for each gradient in  $\nabla_i \in G$  as follows. Fang-A computes two aggregates using  $A$ , one with  $\nabla_i$  and one without  $\nabla_i$  in  $G$ , i.e.,  $A(G)$  and  $A(G - \nabla_i)$ . Fang-A then computes losses and/or errors of the models obtained by updating the global model using the two aggregates. Then Fang-A assigns a score to each  $\nabla_i$  such that the lower the negative impact of  $\nabla_i$  on the loss and/or error of the corresponding model, the higher the score. Finally, Fang-A discards the gradients with the lowest scores. For any given AGR, [64] proposes three defenses, one based on the loss of model, one based on error of model, and one based on both the loss and error. The combination of loss and error works strictly better than either loss or error, hence, we consider the defense based on the combination of loss and error.

## 2.3 Related works

Section 2.2 discusses the literature related to untargeted poisoning attacks in detail. Below, we discuss existing works on targeted and backdoor attacks.

**Targeted attacks** [27, 148, 151] aim to make the global model misclassify a specific set of samples at test time. Bhagoji et al. [27] aimed to misclassify a *single sample* and

proposed a model poisoning attack based on alternate minimization to make poisoned update look similar to benign updates. [27] shows that their attack, with a single attacker, can misclassify a single sample with 100% success against the non-robust Average AGR. try Sun et al. [148] investigated constrain-and-scale attack [18] with the aim to misclassify all samples of a few victim FL clients. Tolpegin et al. [151, 64] investigated targeted data poisoning attacks when compromised clients compute their updates by mislabeling the target samples.

**Backdoor attacks** [18, 155, 164] aim to make the global model misclassify the samples with adversary-chosen backdoor trigger. Backdoor attacks are *semantic*, if the trigger is naturally present in samples [18, 155] and *artificial* if the trigger needs to manually added at test time [164]. Bagdasaryan et al. [18] demonstrate a constrain-and-scale attack against simple Average AGR to inject semantic backdoor in the global model. They show that their attacks achieve accuracy of >90% on backdoor task in a next word prediction model. Wang et al. [155] propose data and model poisoning attacks to inject backdoor to misclassify out-of-distribution samples. Xie et al. [164] show how multiple colluding clients can distribute backdoor trigger to improve the stealth of poisoned updates. Backdoor (as well as targeted) attacks can be further divided in *specific-label* and *arbitrary-label* attacks. For a backdoored test sample, specific-label attack aims to misclassify it to a specific target class, while arbitrary-label attack aims to misclassify it to any class.

Note that, trivial extensions of the targeted and backdoor attack algorithms to mount untargeted attacks cannot succeed, because untargeted attacks aim at affecting almost *all* FL clients and test inputs. For instance, a simple label flipping based data poisoning [155] can insert a backdoor in FL with state-of-the-art defenses. However, such label flipping based untargeted poisoning attacks have no effect even on unprotected FL (Section 6.3.2).



**Existing Defenses Against Targeted and Backdoor Attacks** In Section 2.2.2, we discuss the defenses against untargeted poisoning in detail. Here, we review existing defenses against targeted and backdoor attacks. FoolsGold [66] identifies clients with similar updates as attackers, but incur very high losses in performances as noted in [65]. Sun et al. [148] investigate efficacy of norm-bounding to counter targeted poisoning and, as we will show, is also effective against untargeted poisoning. CRFL [163] counters backdoor attacks by providing certified accuracy for a given test input, but incurs large losses in FL performance (Table 2.1). Defenses based on pruning techniques [159, 154] remove parts of model that are affected by targeted/backdoor attacks, and hence cannot be used against untargeted attacks which affect the entire model.

## CHAPTER 3

# SYSTEMATIZATION OF POISONING ATTACKS AGAINST FEDERATED LEARNING

In the previous chapter, we introduced various types of poisoning attacks and defenses, i.e., robust aggregation rules( AGRs), in federated learning (FL). But, not all of these attacks, and especially, defenses are generally applicable to all FL settings. For instance, Sun et al. [147] introduce an attack that is only applicable to multi-task learning based FL, while Cao et al. [38] introduce a defense that is only applicable in cross-silo FL settings. Furthermore, different works make different assumptions about the FL setting and adversary they consider while designing their attacks/defenses. But there is no comprehensive taxonomy of attacks or threat models<sup>1</sup> that can help practitioners compare these various attacks/defenses. To this end, *in this chapter, we provide a systemization of the threat models of FL poisoning attacks.*

In Chapter 1, we introduced FL framework and the threat of poisoning to FL. In this chapter, we give a very detailed view of the poisoning threat to FL by providing a comprehensive systematization of poisoning attacks on FL. Specifically, we discuss three key dimensions of the poisoning threat to FL: The adversary’s objective, knowledge, and capability, and in Section 3.1.1.1.1, justify in detail why we choose to focus on untargeted poisoning attacks on FL in this thesis.

---

<sup>1</sup>In a nutshell, threat model specifies the setting, e.g., what is the FL setting under consideration, who is the adversary, what knowledge they have about the FL, etc.

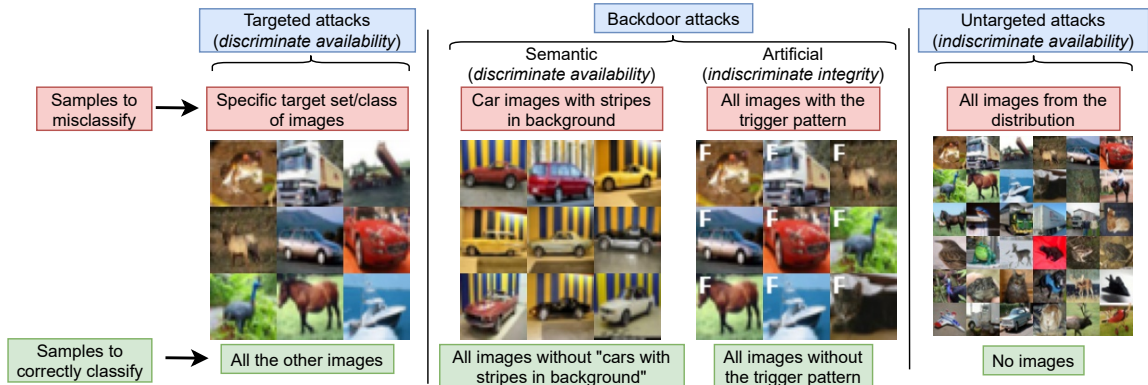


Figure 3.1: Classes of FL poisoning attacks and their objectives defined using the taxonomy in Section 3.1.1.1: *Targeted* attacks [148, 27] aim to misclassify only a specific set/classes of inputs (e.g., certain 10 samples from CIFAR10), *semantic backdoor* attacks [18, 155] aim to misclassify inputs with specific properties (e.g., cars with stripes in background), *artificial backdoor* attacks [164] aim to misclassify inputs with an artificial (visible or invisible) trigger pattern (e.g., shape of letter "F"), and *untargeted* attacks [132, 64] aim to reduce model accuracy on *arbitrary* inputs (e.g., the entire CIFAR10 distribution).

### 3.1 Systematization of FL Poisoning Threat Models

#### 3.1.1 Dimensions of Poisoning Threat to FL

In this section, we build on previous systemization efforts for adversarial ML [21, 76, 112, 29] and present three key dimensions for the threat model of FL poisoning, as shown in Table 3.1.

##### 3.1.1.1 Adversary’s Objective

Inspired by [29], we define three attributes of the adversary’s objectives.

**Security violation:** The adversary may aim to cause an *integrity* violation, i.e., to evade detection without disrupting normal service operations, or an *availability* violation, i.e., to compromise the service for legitimate users.

**Attack specificity:** The attack is *discriminate* if it aims to cause misclassification of a specific set/class of samples; it is *indiscriminate* otherwise.

**Error specificity:** This attribute is especially relevant in multi-class classification settings. It is *specific* if the attacker’s goal is to have a sample misclassified as a

specific class; the attack is *generic* if the attacker does not care about the wrong label assigned to the misclassified samples.

**Adversary objectives in different classes of poisoning:** Here, based on the above taxonomy, we discuss the adversary’s objective for different types of poisoning attacks (Figure 3.1).

*Targeted attacks* [27, 151] aim to misclassify specific sets/classes of input, hence they are “discriminate.” Such discriminate attacks can be either used for “integrity” or “availability” violations, depending on how the poisoned data is used.

*Semantic backdoor attacks* [18, 155] have the same goal as the targeted attacks, but the targeted inputs should have specific properties, e.g., a pixel pattern or a word sequence. Hence, these are “discriminate,” “availability” or “integrity” attacks.

*Artificial backdoor attacks* [164] aim to misclassify *any* input containing a backdoor *trigger*, hence these attacks are “indiscriminate” attacks. Note that, such test inputs should be modified to have the backdoor trigger and only the adversary or a malicious client know the trigger. Hence, these attacks aim to evade the detection, i.e., cause an integrity violation. Hence, these are “integrity indiscriminate” attacks.

*Untargeted attacks* [132, 64, 22] aim to misclassify any test input, i.e., they are “indiscriminate” attacks. But, test inputs need not be modified in order to misclassify. Hence, these are “availability” attacks.

Finally, we note that the error specificity of each of these attacks can be either “specific” or “generic.”

**3.1.1.1.1 Focus of this thesis:** In this thesis, *we focus on untargeted attacks*, i.e., indiscriminate availability attacks with generic error specificity, for the following reasons.

(1) *Untargeted attacks pose a great threat to production FL:* Untargeted attacks are designed to impact all clients and all test inputs. For instance, FL on FEMNIST

achieves an 85% [127] accuracy in a benign setting, and untargeted attacks reduce the accuracy to, e.g., [78, 82]% depending on the percentages of malicious clients. Such an accuracy drop is significant for production FL, as *a malicious service provider can gain advantage over their competitors by causing such small, yet noticeable, accuracy reductions in the competing services and such small accuracy reductions can impact most clients and data from all classes in arbitrary fashion.*

(2) *Untargeted attacks can go undetected for long duration:* As discussed above, the untargeted attack aims at reducing the overall accuracy of the global model, even by only a few percentage points. Such a small reduction in accuracy is hard to detect in practical settings due to the absence of reliable benchmarks for the target application. For instance, the affected service provider will never know that they could have achieved an 85% accuracy and will believe that [78, 82]% is the highest achievable accuracy.

(3) *Constructing untargeted attacks is more challenging:* Untargeted attacks aim to solve a more challenging problem, which is affecting arbitrary test inputs. However, while there exist several defenses to protect FL against untargeted poisoning [30, 108, 132, 168], *these attacks are not studied under production FL environments* (as discussed later on).

### 3.1.1.2 Adversary’s Knowledge

Below we elaborate on two dimensions of adversary’s knowledge: knowledge of the global model and knowledge of the data from the benign distribution.

***Knowledge of the global model:*** This can be *nobox* or *whitebox*. In the *nobox* case, the adversary does not know the model architecture, parameters, or outputs, and is the most practical setting in FL [82], e.g., the data poisoning adversary has *nobox* knowledge of the global model. In the *whitebox* case, the adversary knows the global model parameters and outputs, whenever the server selects at least one

compromised client. The model poisoning adversary always has whitebox knowledge of the global model. As we will explain in Section 6.1.3, this is a relatively less practical setting in FL, as it assumes complete control of the compromised devices.

***Knowledge of the data from benign distribution:*** This can be *full* or *partial*. In full knowledge case, the adversary can access the benign local data of compromised as well as benign clients. In partial knowledge case, the adversary can access the benign local data only of the malicious clients. We only consider the partial knowledge case, because accessing the data of all the clients is impractical in production FL.

### 3.1.1.3 Adversary’s Capability

Below, we elaborate on the the adversary’s capability in terms of *access to client devices* and *frequency of attack*, i.e., the *attack mode*.

***Capability in terms of access to client devices:*** Based on the FL stages (part of FL pipeline on client device) that the adversary poisons, there can be a *model poisoning adversary* or a *data poisoning adversary*. The model poisoning adversary can break into a compromised device (e.g., by circumventing the security protocols of operating systems such as Android) and can *directly manipulate* the poisoned updates [64, 22, 132, 30, 108, 123]. This adversary can craft highly effective poisoned updates, but due to unreasonable amount of access to client devices, it can compromise very small percentages of FL clients [82, 6].

On the other hand, a data poisoning adversary cannot break into a compromised device and can only poison its local dataset. The malicious clients use their local poisoned datasets to compute their poisoned updates, hence this adversary *indirectly manipulates* the poisoned updates. Due to the indirect manipulation, these updates may have less poisoning impact than the model poisoning updates. But, due to the limited access required to the malicious clients, this adversary can compromise relatively large percentages of FL clients [82, 6].


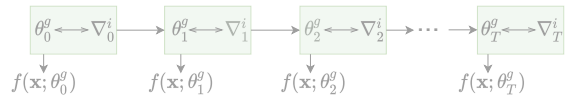


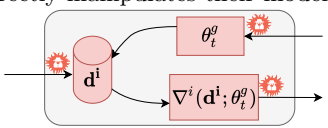
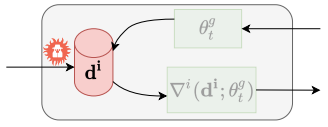


**Capability in terms of attack frequency (Attack mode):** The mode of poisoning attacks on FL can be either *offline* or *online*. In the offline mode, the adversary poisons the malicious clients only once before the start of FL training, e.g., the baseline label flip attack [64] flips the labels of data of malicious clients once before the FL training starts. In the online mode, the adversary *repeatedly* and *adaptively* poisons the malicious clients, e.g., existing model poisoning attacks [132, 22] repeatedly poison the updates of malicious clients selected by the server.

Finally, we assume that the malicious clients can collude to exchange their local data and model updates in order to increase impacts of their attacks.

### 3.2 Takeaways

We provided a comprehensive systemization of FL poisoning threat models. We hope that this systemization will help FL practitioners to classify existing threat models and understand how they apply to the FL setting of their interest. A very important use of the systemization is that it helps us understand which threat models are practically relevant, e.g., in Section 6.1 of Chapter 6 we discuss how only two threat models, out of many that the literature has considered, are truly relevant in practice. Finally, our systemization will help build new threat model for potential future threats and also will serve as a foundation to build more comprehensive systemization.

Table 3.1: The key dimensions of the threat models of poisoning attacks on FL. Each combination of these dimensions constitutes a threat model (Table 6.2). However, we argue in Section 6.1.2 that only two of these combinations are practical threat models.

Dimension	Attribute	Values	Description
Objective of the adversary	Security violation	Integrity	Misclassify a (adversarially crafted) test input in order to evade detection.
		Availability	Misclassify an unmodified test input to cause service disruption for benign users.
	Attack specificity	Discriminate	Misclassify a small and/or specific set of inputs at the test time.
		Indiscriminate	Misclassify all or most of inputs at the test time.
	Error specificity	Specific	Misclassify a given modified/pristine test input to a specific class.
		Generic	Misclassify a given modified/pristine test input to any class.
Knowledge of the adversary	Knowledge of the global model	Whitebox	Adversary can access the global model parameters as well as its predictions, e.g., in the model poisoning case. 
		Nobox	Adversary cannot access parameters or predictions of global model, e.g., in the data poisoning case. 
	Knowledge of the data from the distribution of benign clients' data	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p><b>Full knowledge</b></p>  <p>Adversary can access the local data of all of the collaborating clients, i.e., benign and compromised clients, in FL.</p> </div> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p><b>Partial knowledge</b></p>  <p>Adversary can access local data only of the compromised clients, but not of the benign clients.</p> </div> </div>	
Capabilities of the adversary	Capabilities in terms of access to client devices	Model poison	Adversary breaks into the malicious clients (e.g., by circumventing security protocols of operating systems such as Android) and directly manipulates their model updates. 
		Data poison	Adversary can only manipulate local data of the malicious clients; the clients use this data to compute their updates. Adversary does not break into the malicious clients. 
	Capabilities in terms of frequency of the attack (Attack mode)	Online	Adversary repeatedly and adaptively poisons the malicious clients during FL, e.g., model poisoning attacks [27, 64, 132]. Impacts of these attacks can persist over the entire FL training. 
		Offline	Adversary poisons the malicious clients only once at the beginning of FL, e.g., baseline label flipping attacks [64, 155]. Impact of such attacks may quickly fade away. 



## CHAPTER 4

# GENERAL UNTARGETED POISONING FRAMEWORK AGAINST FEDERATED LEARNING

In the previous chapter, we justified that the untargeted poisoning attacks on FL are a severe threat in practice. To this end, multiple previous works have proposed defenses, also called *robust aggregation rules (AGRs)*, to mitigate the untargeted poisoning by reducing the impact of malicious clients on FL. For instance, Krum AGR [30] removes malicious-looking updates, while Trimmed-mean AGR [168] removes values of each of the dimensions of model updates. Section 2.2.2 systematizes the types of existing robust AGRs and gives representative AGRs of each type. This chapter provides a general framework to empirically evaluate the robustness of such robust AGRs in FL setting.

**Necessity of general FL poisoning attacks framework:** Many of the existing robust AGRs [168, 30, 108] provide theoretical robustness guarantees against untargeted poisoning in FL. For a given percentage of malicious clients, the theoretical robustness guarantees provide upper bounds on the error rates of resulting ML model for convex ML settings. On the other hand, in case of non-convex ML settings, the robustness guarantees can only claim that the global model will eventually converge, but they cannot prove specific bounds on error rates.

Unfortunately, convergence of non-convex ML model does not mean that the model will have low error rate [132, 108], because in the non-convex optimization space there are a large number of local minimas. As non-convex ML is the most widely used ML type in FL [104, 34], it is necessary to *empirically* evaluate the robustness of robust AGRs to understand the true efficacy of the AGRs.

Note that, most of the existing robust AGRs [30, 108, 45, 168, 165] aim to defend against poisoning attacks that arbitrarily reduce the performances of the global model. In other words, these robust AGRs aim to defend against untargeted poisoning in FL. Furthermore, *as we are concerned with evaluating the robustness of AGRs, we consider the stronger (although theoretical) threat model of model poisoning, and not of the data poisoning.* In Chapter 6, we will assess the robustness of FL in more practical threat models of poisoning. There are two major works that propose untargeted model poisoning attacks: *little is enough (LIE)* [22] and *Static Optimization (STAT-OPT)* [64]; Section 2.2.1 details these two attacks. Unfortunately, as we will show, these two model poisoning attacks are not optimal and may give a false sense of security. This may lead to FL practitioners or academics using these AGRs and designing FL applications that are vulnerable to poisoning.

These shortcomings of existing FL poisoning attacks motivates our work in this chapter. We propose a general framework for model poisoning attacks on FL. Unlike prior works [64, 22, 108, 165], we consider a comprehensive set of possible threat models for model poisoning attacks along two dimensions of the adversary’s knowledge: the knowledge of the updates shared by benign clients, and the knowledge of the AGR algorithm that the server uses. We demonstrate that the model poisoning attacks launched using our framework outperform state-of-the-art model poisoning attacks in defeating *all* existing Byzantine-robust FL algorithms. Our attacks lead to significant conclusions, e.g., **(1)** robust AGRs are far more susceptible to poisoning than what their theoretical robustness guarantees claim, **(2)** poisoning adversary can mount a strong poisoning attack even without the knowledge of the server’s robust AGR.

We provide concrete lessons we learn from our empirical assessments that, we hope, will guide future designs of robust AGRs. Finally, based on these lessons we design a new robust AGR called *Divide and Conquer (DnC)* and demonstrate its

state-of-the-art protection against our poisoning attacks. Full details of our general poisoning attacks framework and proposed defense are in [132].

## 4.1 Threat Models of Untargeted Poisoning We Use

Here, we detail the threat models of untargeted poisoning that we consider for investigations in this chapter.

**Adversary’s objective:** The goal of the adversary is to craft malicious gradients such that when the malicious clients share the malicious gradients with the central server, the accuracy of the resulting global model reduces indiscriminately, i.e., on any test input. This is also known as *untargeted model poisoning attack*.

**Adversary’s capabilities:** We assume that the adversary controls up to  $m$  out of  $n$  total clients, called *malicious clients*. We assume that the number of malicious clients is less than the number of benign clients, i.e.,  $(m/n) < 0.5$ ; otherwise, no Byzantine-robust AGR will be able to defeat poisoning attacks. Following the previous works [27, 22, 18, 64, 165, 73], we assume that the adversary can access the global model parameters broadcast in each epoch and can directly manipulate the gradients on malicious devices.

**Adversary’s knowledge:** We consider two important dimensions of FL setting: knowledge of the gradient updates (simply gradients) shared by the benign devices and knowledge of the AGR algorithm of the server. More specifically, we consider four adversaries as shown in Table 4.1. **agr-updates** adversary is the strongest adversary who knows both the gradients of benign devices and the server’s AGR. Although **agr-updates** adversary has limited practical significance, it has been commonly used in previous works [64, 165, 22] to understand the severity of the model poisoning threat. Furthermore, it allows the service provider (the server in this case) to evaluate the robustness of its AGR algorithms. **agr-only** adversary knows the server’s AGR, but does not have the gradients of benign devices. To compute malicious gra-

Table 4.1: Knowledge-based classification of model poisoning adversaries in FL.

Type	Gradients of benign devices	Server’s AGR algorithm
<code>agr-updates</code>	✓	✓
<code>agr-only</code>	✗	✓
<code>updates-only</code>	✓	✗
<code>agnostic</code>	✗	✗

dients, `agr-only` adversary uses benign gradients computed using the benign data on malicious devices. `updates-only` adversary has the gradients of benign devices, but does not know the server’s AGR. We consider this adversary in order to demonstrate *the empirical upper bound* of the severity of our AGR-agnostic attacks. Finally, the `agnostic` adversary does not have the gradients on benign devices or the server’s AGR, and is *the weakest possible adversary in FL*.

Note that, none of the state-of-the-art untargeted model poisoning attacks thoroughly consider these two dimensions: Fang attacks [64] assume the complete knowledge of the server’s AGR algorithm, while LIE attacks [22] assumes the complete knowledge of the gradients of benign devices.

## 4.2 Our Generic Framework for Model Poisoning

In this section, we describe our generic framework to mount model poisoning attacks on FL, followed by specific optimizations for different AGRs and threat models, and finally give an algorithm to solve the optimizations.

### 4.2.1 General optimization formulation

In each FL training epoch, the malicious and benign clients share malicious and benign gradients, respectively, and then the server updates the global model using an aggregate of all of the gradients. To successfully mount an untargeted attack, our

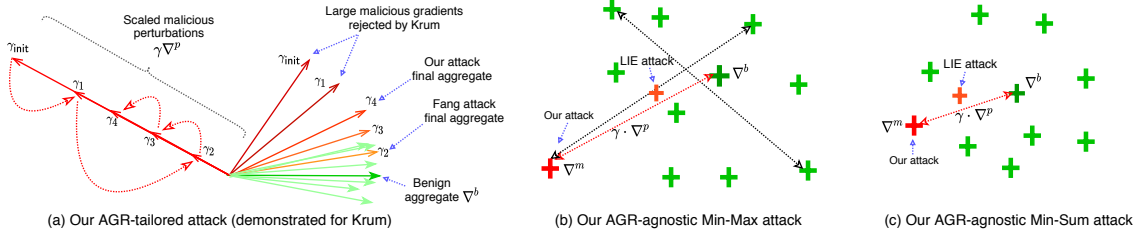


Figure 4.1: **Schematics of our attacks:** (a) Our AGR-tailored attack, unlike Fang attack, fine tunes the malicious gradient ( $\nabla^b + \gamma\nabla^P$ ), using optimal  $\gamma$  and dataset-optimized  $\nabla^P$ . (b) Our AGR-agnostic Min-Max attack finds its malicious gradient  $\nabla^m$  (red cross) whose maximum distance from any other gradient is less than the maximum distance between any two benign updates (black arrows). (c) Our AGR-agnostic Min-Sum attack finds  $\nabla^m$  (red cross) whose sum of distances from the other updates is less than the sum of distances of any benign gradient from the other benign updates. Due to stricter constraints,  $\nabla^m$  of Min-Sum attack is closer to the benign aggregate,  $\nabla^b$ , than  $\nabla^m$  of Min-Max attack. LIE attack computes very suboptimal  $\nabla^m$  due to extremely small amounts of noise additions.

general optimization problem aims to maximize the damage to the global model in each FL epoch.

In order to maximize the damage to the global model, we craft the malicious gradients, denoted by  $\nabla_{\{i \in [m]\}}^m$ , such that the aggregate computed by the server is far from a *reference benign aggregate*, denoted by  $\nabla^b$ . A possible  $\nabla^b$  is the average of the benign gradients that the adversary knows. For instance, the **agr-only** adversary can compute  $m$  benign gradients using the benign data on malicious devices. The final malicious gradient  $\nabla^m$  is a perturbed version of the benign aggregate  $\nabla^b$ , i.e.,  $\nabla^m = \nabla^b + \gamma\nabla^P$ , where  $\nabla^P$  is a *perturbation vector* and  $\gamma$  is a *scaling coefficient*. Therefore, the objective of the full knowledge **agr-updates** adversary is given by (4.1).

$$\operatorname{argmax}_{\gamma, \nabla^P} \|\nabla^b - f_{\text{agr}}(\nabla_{\{i \in [m]\}}^m \cup \nabla_{\{i \in [m+1, n]\}})\|_2 \quad (4.1)$$

$$\nabla_{i \in [m]}^m = \nabla^b + \gamma\nabla^P; \quad \nabla^b = f_{\text{avg}}(\nabla_{\{i \in [n]\}})$$

where  $\nabla_{\{i \in [n]\}}$  are the benign gradients that the adversary knows. Note that state-of-the-art robust AGRs [30, 168, 108] are generally not differentiable. Hence, solving (4.1), i.e., finding the optimal  $\gamma$  and  $\nabla^P$ , using gradient descent based optimizations is not trivial. Our idea to overcome this challenge is to fix the perturbation vector

$\nabla^p$  and find the optimal  $\gamma$ , i.e., solve the modified objective in (4.2). Algorithm 1 (Section 4.2.4) describes our algorithm to optimize  $\gamma$ .

$$\begin{aligned} \operatorname{argmax}_{\gamma} \quad & \|\nabla^b - f_{\text{agr}}(\nabla_{\{i \in [m]\}}^m \cup \nabla_{\{i \in [m+1, n]\}})\|_2 \\ & \nabla_{i \in [m]}^m = \nabla^b + \gamma \nabla^p; \quad \nabla^b = f_{\text{avg}}(\nabla_{\{i \in [n]\}}) \end{aligned} \quad (4.2)$$

#### 4.2.1.1 Introducing perturbation vectors

A perturbation vector is any malicious direction in the space of gradients that the adversary can use to perturb  $\nabla^b$  and find the malicious gradients  $\nabla_{\{i \in [m]\}}^m$ . In this work, we experiment with the following three types of  $\nabla^p$ 's.

*Inverse unit vector* ( $\nabla_{\text{uv}}^p$ ). The intuition here is to compute the malicious gradient by perturbing  $\nabla^b$  by a scaled unit vector that points in the opposite direction of  $\nabla^b$ . Hence, we compute  $\nabla_{\text{uv}}^p$  as  $-\frac{\nabla^b}{\|\nabla^b\|_2}$ .

*Inverse standard deviation* ( $\nabla_{\text{std}}^p$ ). The intuition here is that the higher the variance of a dimension of benign gradients, the higher the perturbation that the adversary can introduce along the dimension. Hence, we compute  $\nabla_{\text{std}}^p$  as  $-\text{std}(\nabla_{i \in [n]})$ .

*Inverse sign* ( $\nabla_{\text{sgn}}^p$ ). We compute  $\nabla_{\text{sgn}}^p$  as  $-\text{sign}(f_{\text{avg}}(\nabla_{i \in [n]}))$ . The intuition here is similar to that of ( $\nabla_{\text{uv}}^p$ ), but we observe that ( $\nabla_{\text{sgn}}^p$ ) is more effective for some classification tasks, e.g., MNIST.

We show in [132] that the appropriate choice of perturbation vector  $\nabla^p$  is the key to an effective attack. For instance, for Krum AGR, the attack using  $\nabla_{\text{uv}}^p$  increases the accuracy of global model of MNIST, while the attack using  $\nabla_{\text{uv}}^p$  reduces the accuracy to random guessing for Purchase. Finally, we note that our experiments show that our attacks destroy the global model accuracy and significantly outperform the existing model poisoning attacks using one of these  $\nabla^p$ 's. Hence, we leave investigating the optimal  $\nabla^p$  to future work.

### 4.2.2 AGR-tailored attacks

In this section, we consider `agr-updates` and `agr-only` adversaries, who know the server’s AGR algorithm and tailor the general attack objective in (4.2) to the known AGR. We consider the seven robust AGRs described in Section 2.2.3. For the clarity of presentation, we provide the AGR-tailored optimizations for `agr-updates` adversary with all the benign gradients  $\nabla_{\{i \in [n]\}}$ . The only change in optimizations for `agr-only` adversary is to compute  $\nabla^b$  using the benign gradients computed using the benign data of the  $m$  malicious devices, i.e.,  $\nabla_{\{i \in [m]\}}$ .

#### 4.2.2.1 Krum

Krum<sup>1</sup> selects a single gradient from its inputs as its aggregate. Hence, a successful attack requires Krum to select one of its malicious gradients, i.e.,  $\nabla_{i \in [m]}^m = f_{\text{krum}}(\nabla_{\{i \in [m]\}}^m \cup \nabla_{\{i \in [m+1, n]\}})$ . Therefore, we modify (4.2) to (4.3) for Krum. For each of the input gradients, Krum computes a score that is the sum of distances of  $n - m - 2$  nearest neighbors of the gradient. Therefore, to maximize the chances of Krum selecting a malicious gradient, we keep all the malicious gradients the same.

$$\begin{aligned} \operatorname{argmax}_{\gamma} \nabla_{i \in [m]}^m &= f_{\text{krum}}(\nabla_{\{i \in [m]\}}^m \cup \nabla_{\{i \in [m+1, n]\}}) \\ \nabla_{i \in [m]}^m &= f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^p \end{aligned} \tag{4.3}$$

#### 4.2.2.2 Multi-krum

Multi-krum uses Krum iteratively to construct a selection set  $\mathcal{S}$  and computes average of the gradients in the selection set as its aggregate. Our attack on Multi-krum ensures that all of the malicious gradients are in selected  $\mathcal{S}$ , while maximizing the perturbation  $\gamma \nabla^p$  used to compute the malicious gradients. This strategy minimizes the number of benign gradients in  $\mathcal{S}$ , while maximizing  $\gamma \nabla^p$  increases the poisoning

---

<sup>1</sup>We omit suffix AGR, when it is clear from the context.

impact of malicious gradients on the final aggregate. Therefore, we modify (4.2) to (4.4) for Multi-krum; here  $|A|$  is the cardinality of  $A$ .

$$\begin{aligned} \operatorname{argmax}_{\gamma} m &= |\{\nabla \in \nabla_{\{i \in [m]\}}^m \mid \nabla \in \mathcal{S}\}| & (4.4) \\ \nabla_{i \in [m]}^m &= f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^{\text{P}} \end{aligned}$$

#### 4.2.2.3 Trimmed-mean

For Trimmed-mean, we directly solve the optimization described by (4.2), by fixing the perturbation  $\nabla^{\text{P}}$  and keeping all the malicious updates the same. Hence, our objective is to maximize the  $L_2$ -norm of the distance between the reference benign update  $\nabla^b$  and the aggregate computed using Trimmed-mean on the set of benign and malicious updates. This is formalized in (4.5).

$$\begin{aligned} \operatorname{argmax}_{\gamma} \|\nabla^b - f_{\text{trmean}}(\nabla_{\{i \in [m]\}}^m \cup \nabla_{\{i \in [m+1, n]\}})\|_2 & & (4.5) \\ \nabla_{i \in [m]}^m &= f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^{\text{P}} \end{aligned}$$

Note that in (4.5), we aim to compute  $\gamma$  that maximizes the required  $L_2$ -norm distance. As we demonstrate in our evaluations, this extremely simple approach of crafting malicious updates outperforms the complex approaches proposed by Fang et al. [64] attacks by very large margins for all the datasets.

#### 4.2.3 AGR-agnostic attacks

Now, we consider the AGR-agnostic adversaries, `updates-only` and `agnostic`, who do not know the server’s AGR algorithm. This is an important practical consideration, because the FL platforms can conceal the details and/or parameters of their robust AGRs to protect the security of the proprietary global models. Below, we first provide intuition behind our attacks and then propose two AGR-agnostic attacks to craft malicious gradients.



### 4.2.3.1 Intuition

All the robust AGRs for FL tend to remove/attenuate malicious gradients based on one or more of the following criteria: 1) distances from the benign gradients [30, 27, 108, 13, 168], 2) distributional differences with the benign gradients [27, 148], 3) difference in  $L_p$ -norms of the benign and malicious gradients [148]. Figures 4.1-(b, c) visualize the intuition behind our attacks based on the above criteria. The intuition is as follows. The distance based defenses work by removing the gradients that lie outside of the clique formed by the benign gradients. Therefore, our attacks maximize the distance of malicious gradient from a reference benign gradient, while ensuring that the malicious gradients lie within the clique of benign gradients. This also ensures that  $L_p$ -norms of the malicious and benign gradients are similar. To ensure distributional similarity, we use perturbations  $\gamma \nabla^p$  with the similar distributions as the benign gradients.

Next, we present optimization for two novel AGR-agnostic attacks based on the intuition. We present the optimizations for **updates-only** adversary, who has all the benign gradients  $\nabla_{\{i \in [n]\}}$ . The extension to **agnostic** adversary is similar to that explained at the beginning of Section 4.2.2 for **agr-only** adversary.

### 4.2.3.2 Attack-1 (Min-Max): Minimize maximum distance attack

Our first attack ensures that the malicious gradients lie close to the clique of the benign gradients (Figure 4.1-(b)). Hence, we compute the malicious gradient such that its maximum distance from any other gradient is upper bounded by the maximum distance between any two benign gradients. (4.6) formalizes the corresponding optimization. Note that in order to maximize the impact of our attack, we keep all the malicious gradients the same. Hence, we formulate our attack objective in (4.6) for a single malicious gradient.

$$\operatorname{argmax}_{\gamma} \max_{i \in [n]} \|\nabla^m - \nabla_i\|_2 \leq \max_{i, j \in [n]} \|\nabla_i - \nabla_j\|_2 \quad (4.6)$$

$$\nabla^m = f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^p$$

#### 4.2.3.3 Attack-2 (Min-Sum): Minimize sum of distances attack

Our second AGR-agnostic Min-Sum attack ensures that the sum of squared distances of the malicious gradient from *all* the benign gradients is upper bounded by the sum of squared distances of any benign gradient from the other benign gradients (Figure 4.1-(c)). (4.7) formalizes the corresponding optimization. We keep all malicious gradients the same for maximum attack impact. Hence, we formulate our objective in (4.7) for a single malicious gradient.

$$\operatorname{argmax}_{\gamma} \sum_{i \in [n]} \|\nabla^m - \nabla_i\|_2^2 \leq \max_{i \in [n]} \sum_{j \in [n]} \|\nabla_i - \nabla_j\|_2^2 \quad (4.7)$$

$$\nabla^m = f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^p$$

#### 4.2.4 Solving for the most effective scaling factor $\gamma$

In previous sections, we formulated optimizations for various adversarial settings such that the final objective is to search for the optimal scaling coefficient,  $\gamma$ . Algorithm 1 describes our algorithm to optimize  $\gamma$  for any of the optimizations.

For clarity of presentation of Algorithm 1, we assume an oracle  $\mathcal{O}$  that takes the set of benign gradients,  $\nabla_{\{i \in [n]\}}$  and  $\gamma$  as inputs. Then,  $\mathcal{O}$  computes malicious gradients as  $\nabla_{\{i \in [m]\}}^m = \nabla^b + \gamma \nabla^p$ , and outputs **True** if they satisfy the adversarial objective, otherwise outputs **False**. For instance, for our AGR-tailored attack on Krum,  $\mathcal{O}$  outputs **True** if a malicious gradient is selected by  $f_{\text{krum}}$ , i.e., if (4.3) is satisfied. For our Min-Max attack (Section 4.2.3.2),  $\mathcal{O}$  outputs **True** if the maximum distance of malicious gradient from any benign gradient is lower than the maximum distance between any two benign gradients, i.e., if (4.6) is satisfied.

Now, we describe Algorithm 1. The core idea of our algorithm is as follows: We start with a large  $\gamma$  value. We reduce  $\gamma$  in steps of size `step` until  $\mathcal{O}$  returns `True`, e.g., for Krum, we reduce  $\gamma$  until a malicious gradient is selected by  $f_{\text{krum}}$ , i.e., (4.3) is satisfied for the first time. Our final  $\gamma$  is always *greater* than this *minimum*  $\gamma$  value that satisfies the objective. We halve the step size each time we update  $\gamma$  in order to make the search finer. From the minimum  $\gamma$  value, we increase  $\gamma$  using updated step sizes `step`, until  $\mathcal{O}$  returns `False`, i.e., for Krum, we increase  $\gamma$  until  $f_{\text{krum}}$  *does not* select any malicious gradient, i.e., (4.3) is no more satisfied. Our final  $\gamma$  is always *lower* than this *maximum*  $\gamma$  value that satisfies the objective. Then we modify  $\gamma$  repeatedly and oscillate between the minimum and maximum  $\gamma$  values until the change in  $\gamma$  is below a threshold  $\tau$ .

---

**Algorithm 1** Algorithm to optimize  $\gamma$

---

```

1: Input:  $\gamma_{\text{init}}, \tau, \mathcal{O}, \nabla_{\{i \in [n]\}}$ 
2: step  $\leftarrow \gamma_{\text{init}}/2, \gamma \leftarrow \gamma_{\text{init}}$ 
3: while  $|\gamma_{\text{succ}} - \gamma| > \tau$  do
4:   if  $\mathcal{O}(\nabla_{\{i \in [n]\}}, \gamma) == \text{True}$  then
5:      $\gamma_{\text{succ}} \leftarrow \gamma$ 
6:      $\gamma \leftarrow (\gamma + \text{step}/2)$ 
7:   else
8:      $\gamma \leftarrow (\gamma - \text{step}/2)$ 
9:   end if
10:  step = step/2
11: end while
12: Output  $\gamma_{\text{succ}}$ 

```

---

## 4.3 Experimental setup

### 4.3.1 Datasets and model architectures

*CIFAR10* [88] is a 10-class *class-balanced* classification task with 60,000 RGB images, each of size  $32 \times 32$ . ‘Class-balanced’ datasets have the same number of samples per class, e.g., each class of CIFAR10 has 6,000 images. We use 50 clients each with 1,000

samples and use validation and test data of sizes 5,000 each. We use Alexnet [89] and VGG11 [139] as the global model architectures.

*Purchase* [4] is a 100-class class-imbalanced classification task with 197,324 binary feature vectors, each of length 600. We use 80 clients each with 2,000 training samples and use validation and test data of sizes 5,000 each. We use a fully connected network with layer sizes {600, 1024, 100}.

*FEMNIST* [36, 51] is a character recognition classification task with 3,400 clients, 62 classes, and a total of 671,585 grayscale images. Each of the 3,400 clients has her own data made of her own handwritten digits or letters (62 classes: 52 for upper and lower case letters and 10 for digits). The mean and standard deviation of the number of samples per client are 226.83 and 88.94, respectively. In each FL epoch, we randomly select 60 out of 3400 clients for FL training. FEMNIST is a non-iid, class-imbalanced dataset commonly encountered in cross-device FL settings [82], while the previous datasets are more common in cross-silo FL settings.

### 4.3.2 Learning and attacks settings

We train CIFAR10 with Alexnet using batch size of 250 and SGD optimizer with learning rates of 0.5 from epochs 0-1000 and 0.05 from 1000-1200. We train CIFAR10 with VGG11 using batch size of 200 and SGD optimizer with learning rates of 0.1 from epochs 0-1000 and 0.01 from 1000-1200. We train MNIST for 500 epochs using Adam optimizer with 0.001 learning rate and batch size of 100. We train Purchase for 1000 epochs using SGD with learning rate of 0.5 and batch size of 500. We train FEMNIST for 1500 epochs using Adam optimizer with learning rate of 0.001 and use client’s entire data in a each batch.

Unless specified otherwise, we assume 20% malicious clients for all adversarial settings, e.g., 20 malicious clients for MNIST. For most of our evaluation, we use inde-

pendently and identically distributed (iid) CIFAR10, MNIST, and Purchase datasets, because poisoning FL with iid data is the hardest [64].

**Measurement metrics.** For a given FL setting,  $A_\theta$  denotes the accuracy of the best global model, over all the FL training epochs, in the benign setting without any attack, while  $A_\theta^*$  denotes the accuracy under the given attack. We define *attack impact*,  $I_\theta$ , as *the reduction in the accuracy of the global model due to the attack*, hence for a given attack,  $I_\theta = A_\theta - A_\theta^*$ .

**Baseline Model Poisoning Attacks:** We use two state-of-the-art model poisoning attacks, LIE [22] and STAT-OPT [64] described in Section 2.2.1 to compare with our attacks.

## 4.4 Evaluation of Our Attacks

### 4.4.1 Comparison with the state-of-the-art attacks

In this section, we compare our attacks with state-of-the-art model poisoning attacks, Fang [64] and LIE<sup>2</sup> [22], for all the adversaries from Table 4.1. The results are given in Table 4.2; ‘No attack’ column shows accuracy  $A_\theta$  of the global model in the benign setting, while the rest of the columns show the ‘attack impact’  $I_\theta$ , as defined in Section 4.3.2.

For a fair comparison, we compare the attacks that use the knowledge of AGR, i.e., our AGR-tailored and Fang attacks under `agr-updates` and `agr-only` adversaries. We separately compare the attacks that do not use the knowledge of AGR, i.e., our AGR-agnostic and LIE attacks under `updates-only` and `agnostic` adversaries.

#### 4.4.1.1 Comparing AGR-tailored attacks

Table 4.2 shows that, *our AGR-tailored attacks outperform Fang attacks for all the combinations of threat model, AGR, dataset, and model archi-*

---

<sup>2</sup>We omit the suffix ‘attack’ when it is clear from the context.

Table 4.2: Comparing state-of-the-art model poisoning attacks and our attacks under various threat models from Table 4.1, when cross-silo FL is used. In all the settings, the impact of our AGR-tailored attacks is significantly higher than that of AGR-tailored Fang attacks. While both of our AGR-agnostic attacks outperform AGR-agnostic LIE attacks in most cases. We assume 20% malicious clients and, except for ‘No attack’ column, report *the attack impact*  $I_\theta$  (Section 4.3.2). For each adversary, we bold  $I_\theta$  of the strongest attack. MMax and MSum are our Min-Max and Min-Sum attacks, respectively.

Dataset (Model)	AGR	No attack ( $A_\theta$ )	Gradients of benign devices are known				Gradients of benign devices are unknown			
			AGR tailored (agr-updates)		AGR agnostic (updates-only)		AGR tailored (agr-only)		AGR agnostic (agnostic)	
			Fang	Ours	LIE	Ours MMax (MSum)	Fang	Ours	LIE	Ours MMax (MSum)
CIFAR10 (Alexnet)	Krum	53.5	21.8	<b>43.6</b>	9.9 (17.4)	<b>30.1</b>	19.8	<b>43.1</b>	18.1	13.7 ( <b>30.2</b> )
	MKrum	67.6	12.6	<b>36.8</b>	20.5 (27.8)	<b>30.8</b>	11.2	<b>35.3</b>	19.7	<b>31.7</b> (30.4)
	Bulyan	66.9	12.3	<b>45.6</b>	33.8 (35.5)	<b>44.5</b>	11.8	<b>34.6</b>	30.0	40.6 ( <b>41.1</b> )
	TrMean	67.7	15.8	<b>45.8</b>	22.8 ( <b>41.6</b> )	33.5	12.9	<b>45.0</b>	19.4	<b>38.7</b> (27.9)
	Median	65.5	12.9	<b>40.9</b>	20.7 (34.7)	<b>39.6</b>	12.6	<b>39.1</b>	19.7	34.1 ( <b>39.5</b> )
	AFA	66.8	7.0	<b>47.0</b>	5.9 ( <b>31.5</b> )	16.9	6.1	<b>46.8</b>	5.5	<b>22.2</b> 16.0
	Fang	66.8	8.9	<b>56.3</b>	6.5 ( <b>42.9</b> )	21.5	8.5	<b>56.0</b>	6.3	<b>42.1</b> (19.9)
Purchase (FC)	Krum	62.1	6.0	<b>61.3</b>	-15.8 ( <b>60.6</b> )	59.1	4.4	<b>60.8</b>	-17.7	<b>61.1</b> (61.0)
	MKrum	91.9	13.7	<b>21.4</b>	1.5 ( <b>20.4</b> )	18.4	12.2	<b>18.2</b>	1.7	<b>19.8</b> (16.4)
	Bulyan	91.3	14.7	<b>28.7</b>	10.9 (23.4)	<b>30.0</b>	20.9	<b>28.4</b>	8.4	28.0 ( <b>30.3</b> )
	TrMean	92.0	1.8	<b>23.4</b>	2.3 ( <b>16.9</b> )	5.4	1.6	<b>22.2</b>	1.9	<b>26.8</b> (14.6)
	Median	87.4	0.2	<b>11.0</b>	0.5 ( <b>11.6</b> )	11.3	-1.0	<b>14.1</b>	-1.6	<b>13.4</b> (12.6)
	AFA	91.7	1.5	<b>3.4</b>	0.7 ( <b>1.4</b> )	0.7	1.4	<b>2.8</b>	0.2	<b>1.3</b> (0.5)
	Fang	91.9	1.2	<b>89.2</b>	0.5 ( <b>18.9</b> )	8.4	0.9	<b>89.2</b>	0.5	<b>8.5</b> (7.8)
FEMNIST (CNN)	Krum	69.3	18.3	<b>30.0</b>	0.9 (0.1)	<b>9.8</b>	1.9	<b>2.9</b>	0.2	1.1 ( <b>8.0</b> )
	MKrum	86.6	34.5	<b>78.8</b>	15.7 ( <b>79.5</b> )	61.7	30.8	<b>57.1</b>	10.2	<b>79.5</b> (61.4)
	Bulyan	86.1	38.9	<b>41.0</b>	32.0 (20.1)	<b>40.0</b>	35.6	<b>40.5</b>	20.5	18.7 ( <b>30.4</b> )
	TrMean	86.7	7.2	<b>24.3</b>	19.1 ( <b>29.7</b> )	26.8	7.9	<b>20.1</b>	14.4	24.7 ( <b>25.2</b> )
	Median	77.1	2.7	<b>30.2</b>	12.0 ( <b>26.7</b> )	17.1	0.8	<b>18.2</b>	5.8	<b>19.8</b> (16.6)
	AFA	84.6	6.2	<b>77.0</b>	7.4 ( <b>74.4</b> )	50.0	2.1	<b>75.3</b>	4.6	<b>74.0</b> (46.0)
	Fang	86.0	7.6	<b>83.1</b>	1.8 ( <b>81.6</b> )	62.3	2.8	<b>83.0</b>	1.7	<b>78.3</b> (60.1)

*ecture by large margins.* For CIFAR10 with `agr-updates` adversary, our attacks are  $2\times$  more impactful than Fang. While with `agr-only` adversary, our attacks are  $2.5\times$  and  $4.5\times$  more impactful than Fang for Alexnet and VGG11 models, respectively. For the rest of the AGRs, our attacks are  $3\times$  to  $7\times$  ( $2\times$  to  $4\times$ ) more impactful than Fang attacks on CIFAR10 with Alexnet (VGG11) for both `agr-updates` and `agr-only` adversaries.

Under `agr-updates` (`agr-only`) adversary, Fang and our attacks on Krum with MNIST have impacts of 20.5% (17.4%) and 33.9% (24.1%), respectively, i.e., our attack is  $1.7\times$  ( $1.5\times$ ) more effective than Fang. The impact of Fang attack on Trimmed-mean (Median) with MNIST is just 1.2% (1.7%), while that of our attack is 11.0% (4.4%), i.e., our attack is  $10\times$  ( $2.5\times$ ) more impactful. Even for AFA, which is the empirically most robust AGR for MNIST, our attack is  $3\times$  more impactful than Fang.

For Purchase, our attacks reduce the accuracy of Krum to the random guessing, i.e., close to 1% for all the adversaries and, except for AFA, our attacks are at least  $10\times$  more impactful than Fang attacks. Similarly, with `agr-updates` adversary, impacts of Fang attacks on Trimmed-mean and Median are 1.8% and 0.2%, respectively, while impacts of our attacks are 23.4% and 11.0%. We note similarly higher impacts of our attacks with `agr-only` adversary. For Multi-krum, Bulyan, and AFA, our attacks are  $2\times$  more impactful than Fang attacks. For Fang-Trmean, the Fang defense that uses Trimmed-Mean to discard malicious gradients, our attacks reduce the global model accuracy to random guessing for all combinations of datasets and models; this is expected as discussed in Section 2.2.3.

For FEMNIST the impacts of our attacks with `agr-updates` adversary on AFA, Multi-krum, Trimmed-mean, and Median are respectively  $12\times$ ,  $2\times$ ,  $3\times$ , and  $15\times$ , that of Fang attack. For Krum and Bulyan also, the impacts of our attacks are moderately higher than that of Fang attacks.

**Why our attacks are superior?** For Krum AGR, although ours and Fang attacks have similar attack objectives, they differ in two instrumental aspects: First, instead of generalizing a single perturbation type across all datasets, our attacks tailor the perturbation to the given dataset as explained in detail in [132]. Next, as Figure 4.1-(a) demonstrates, our Algorithm 1 carefully fine tunes  $\gamma$  of our objective (4.3), while Fang attack simply finds the first  $\gamma$  that satisfies its objective. Our attacks on AFA, Bulyan, and Multi-krum AGRs are also carefully tailored to the AGRs, while Fang attack uses the same objective as Krum for these AGRs.

Fang proposes the same attack for Trimmed-mean and Median AGRs, which crafts the values of each dimension of malicious gradients using the available benign gradients. But our attacks have more tailored and impactful objectives of diverging the final aggregate as far away from a benign aggregate as possible using the most malicious perturbation direction.

#### 4.4.1.2 Comparing AGR-agnostic attacks

Table 4.2 shows that, *both of our AGR-agnostic attacks significantly outperform LIE, the state-of-the-art AGR-agnostic attack* for most of the FL settings that we evaluate. For MNIST with Krum, the impact of Min-Sum attack (simply Min-Sum) is  $3\times$  that of LIE, for both `updates-only` and `agnostic` adversaries. Except for CIFAR10 with VGG11, we note significantly higher impacts of Min-Sum on Krum than that of LIE. Note that *LIE, due to its small noise addition, regularizes and increases accuracy of the global model trained on Purchase using Krum*. For Bulyan, with `agnostic` and `updates-only` adversaries, Min-Sum significantly outperforms LIE by amounts varying from 1.8% (for MNIST) to 22.1% (for Purchase) depending on the classification task.

On the other hand, Min-Max is more effective against Multi-krum and outperforms LIE by amounts varying from 10.2% (MNIST) to 18.1% (Purchase) depending on the



classification task. Min-Max is more effective against AFA, which also computes an average of gradients in a selection set. On AFA, Min-Max outperforms LIE for all datasets but MNIST dataset; for MNIST the two attacks have almost the same impacts. Min-Max is also more effective than LIE and Min-Sum attacks against Trimmed-mean, Median, and Fang-Trmean AGRs. For instance, depending on the classification task, Min-Max is almost  $1.2\times$  (for CIFAR10 + VGG11) to  $8\times$  (for Purchase) more impactful than LIE against Trimmed-mean, while it is almost  $1.2\times$  (for CIFAR10 + VGG11) to  $20\times$  (for Purchase) more impactful than LIE against Median.

LIE attack is ineffective, because it adds very small amounts of noises to compute its malicious gradients, while our AGR-agnostic attacks are much more impactful as they find the most malicious gradient within a ball formed by the benign gradients (Figure 4.1-(b,c)). For the same reason, for all the considered scenarios, except for the combination of Krum and FEMNIST dataset *one or more of our AGR-agnostic attacks also outperform AGR-tailored Fang attacks*. Due to the extreme non-iid nature of FEMNIST, the malicious gradients of our AGR-agnostic attacks can be arbitrarily far from benign gradients, which Krum can easily discard.

**Reasons for the differences in the impacts of Min-Max and Min-Sum attacks:** Min-Max finds the malicious gradient whose maximum distance from a benign gradient is less than the maximum distance between any two benign gradient. While, Min-Sum finds the malicious gradient such that the sum of its distances from all the other gradients is less than the sum of distances of any benign gradient from other benign gradients. Therefore, *as Figures 4.1-(b, c) demonstrate, the radius of search of malicious gradients of Min-Max is much larger than that of Min-Sum*. Therefore, the malicious gradients of Min-Sum more effectively circumvent the filtering of Krum and Bulyan AGRs, and therefore, are more impactful against these AGRs. For the same reason, Multi-krum selects a lesser number of malicious gradients of Min-Max

than that of Min-Sum. But, as Multi-krum averages the selected gradients, Min-Max, with significantly more malicious gradients, damages the Multi-krum aggregate more effectively than Min-Sum.

Finally, we note that for AGR-agnostic attacks, we observe a few cases in Table 4.2 where `agnostic` adversary has slightly more impact than `updates-only` adversary. For example, Min-Max attack on (CIFAR10 + Alexnet + Multi-krum) with `agnostic` adversary has 3.9% more impact than with `updates-only` adversary. The reason for this are various sources of randomness in our experiments. More specifically, we do not use the exact same set of gradients to compute malicious gradients under the two adversaries. Instead, we instantiate the whole FL training every time we compute the attack impact. Therefore, empirical randomness, e.g., random initial model parameters, in running the two different instantiations may cause this behavior. Our experimental results are the average of three such instantiations for each of the presented result, and such empirical anomalies can be mitigated in various ways, including setting the seed for different random number generators and averaging over multiple runs of experiments.

#### 4.4.2 Impact of our attacks on cross-device FL setting

In this section, we evaluate the impact of our attacks when cross-device FL is used to learn on CIFAR10 dataset. More specifically, in each FL epoch, instead of processing all of the 50 clients, we process only 10 clients. As before, we evaluate for two model architectures, Alexnet and VGG11. The attack procedures for different AGRs do not change.

Table 4.3 shows the results. Similar to cross-silo setting, our AGR-tailored attacks outperform the state-of-the-art Fang attacks for both Alexnet and VGG11 architectures. For Alexnet with `agr-updates` adversary, our attack is 2× (Trimmed-mean) to

Table 4.3: Comparing the state-of-the-art model poisoning attacks and our attacks under all threat models in Table 4.1 when *cross-device FL* is used. Our AGR-tailored attacks significantly outperform Fang attacks, while at least one of our AGR-agnostic attacks significantly outperforms LIE attack in most cases. Experimental setup is exactly the same as that of Table 4.2. MMax and MSum are our Min-Max and Min-Sum attacks, respectively.

Dataset (Model)	AGR	No attack ( $A_\theta$ )	Gradients of benign devices are known				Gradients of benign devices are unknown			
			AGR tailored (agr-updates)		AGR agnostic (updates-only)		AGR tailored (agr-only)		AGR agnostic (agnostic)	
			Fang	Ours	LIE	Ours MMax (MSum)	Fang	Ours	LIE	Ours MMax (MSum)
CIFAR10 (Alexnet)	Krum	53.9	11.0	<b>34.0</b>	<b>19.9</b>	8.7 (19.7)	4.5	<b>15.8</b>	<b>14.9</b>	8.7 (13.7)
	MKrum	64.5	2.2	<b>22.2</b>	11.5	15.9 ( <b>17.2</b> )	1.2	<b>14.4</b>	9.2	14.5 ( <b>15.5</b> )
	Bulyan	63.9	2.0	<b>28.8</b>	13.0	<b>28.4</b> (26.4)	1.9	<b>27.9</b>	9.9	<b>22.3 (8.6)</b>
	TrMean	64.9	8.3	<b>14.0</b>	9.3	<b>9.4</b> (7.3)	3.2	<b>9.8</b>	4.9	<b>6.5</b> (4.2)
	Median	62.4	1.8	<b>20.3</b>	4.1	<b>20.3</b> (17.8)	0.2	<b>16.4</b>	-1.6	<b>15.8</b> (10.3)
	AFA	66.2	1.6	<b>41.4</b>	3.8	<b>3.8</b> (2.6)	1.0	<b>36.7</b>	3.4	<b>3.6</b> (1.9)
	Fang	64.5	7.2	<b>54.3</b>	3.6	<b>9.1</b> (6.5)	4.3	<b>54.3</b>	2.3	<b>5.7</b> (5.3)
CIFAR10 (VGG11)	Krum	59.3	3.8	<b>26.3</b>	15.0	6.7 ( <b>20.1</b> )	1.2	<b>12.7</b>	<b>11.8</b>	1.8 (10.7)
	MKrum	72.0	1.4	<b>13.2</b>	<b>9.8</b>	9.0 (8.5)	1.0	<b>9.4</b>	<b>8.8</b>	7.3 (8.3)
	Bulyan	72.0	2.8	<b>18.8</b>	9.2	<b>24.0</b> (14.9)	2.6	<b>17.6</b>	6.7	<b>16.6</b> (12.3)
	TrMean	72.1	5.9	<b>8.7</b>	4.1	<b>6.0</b> (3.6)	3.6	<b>8.1</b>	4.0	<b>5.2</b> (3.0)
	Median	70.2	0.3	<b>11.2</b>	1.0	<b>12.8</b> (11.3)	0.1	<b>10.7</b>	0.8	<b>9.9</b> (7.2)
	AFA	71.8	2.1	<b>15.0</b>	<b>2.3</b>	1.8 (1.5)	2.0	<b>14.7</b>	<b>2.1</b>	1.7 (1.1)
	Fang	71.9	1.9	<b>60.9</b>	3.6	<b>8.9</b> (4.4)	0.6	<b>58.1</b>	2.7	<b>7.1</b> (4.0)

11× (Median) more impactful than Fang attack. We note similar results for `agr-only` adversary as well as VGG11 architecture in Table 4.3.

For AGR-agnostic adversaries with Alexnet, we note that at least one of our Min-Sum and Min-Max attacks has up to 5× more attack impact than the state-of-the-art LIE attack, for all but Krum AGR. For Krum AGR, LIE outperforms our attack by 0.2% and 1.2% under `updates-only` and `agnostic` adversaries, respectively. We note similar results for Alexnet with `agnostic adversary`. In case of VGG11 as well, at least one of our AGR-agnostic attacks has up to 10× more impact than LIE, for all but Multi-krum and AFA AGRs. For Multi-krum and AFA, the LIE and Min-Max have almost equal attack impacts.

Finally we note that, overall the attack impacts are lower in cross-device setting than in cross-silo setting; the reduction in impacts varies widely based on AGR and model architecture. For instance, for Alexnet with Krum, Multi-krum, and Trimmed-mean, the impacts reduce by 9.5%, 14.6%, and 31.8%, respectively. The reason for this is that, in cross-device FL, the adversary cannot constantly corrupt the global model. Because, in many cross-device FL epochs, the number of malicious clients that the server selects can be negligible.

## 4.5 Takeaways

We provide a general framework to design *tailored* untargeted poisoning attacks against robust AGRs that aim to defend FL against poisoning. We also demonstrated the simplicity of using the framework by designing poisoning attacks against seven robust AGRs. Our framework also facilitates designing *AGR-agnostic* attacks that do not require the knowledge of the server’s AGR. Our experiments clearly show that both AGR-tailored and AGR-agnostic attacks designed using our framework outperform previous attacks.

Finally, the main goal of our framework is to understand the flaws in existing AGRs and enable designing more robust AGRs for FL. To this end, below we summarize three lessons we learned from our attacks that will guide the design of future robust FL designs.

#### 4.5.1 Lessons learned from our attacks

**L1: The curse of dimensionality.** The theoretical error bounds provided by previous robust AGRs [30, 108, 168, 59, 95, 13] depend on the dimensionality of their inputs. Hence, the theoretical as well as empirical errors of these defenses explode for high dimensional gradients of neural networks [42] in FL. Therefore, *reducing the dimensionality of input gradients is necessary to improve robustness against poisoning.*

**L2: Convergence is necessary but not sufficient.** All prior robust AGRs [30, 108, 168] give provable convergence guarantees for non-convex FL. However, for non-convex optimizations, such guarantees are meaningless due to large number of suboptimal local optima. Our attacks exploit this and force the global model to converge to a suboptimal local optimum. Therefore, providing convergence guarantees is not enough and *robust AGRs should provide guarantees on how well they detect and remove outliers.*

**L3: Distance- or dimension-wise pruning is insufficient.** Krum, Multi-krum, and Bulyan use  $\ell_p$  distance-based filtering, which, as [108, 22] point out and we show in our work, allows malicious gradients to be close enough to benign gradients while far enough to effectively poison the global model. Dimension-wise pruning in Trimmed-mean and Median allows an adversary to craft gradients which significantly shift the aggregate in bad directions as our and Fang [64] attacks show. Therefore, *robust AGRs need to go beyond just using dimension/distance-based filtering.*

## CHAPTER 5

# DEFENDING FEDERATED LEARNING AGAINST POISONING ATTACKS

Previous chapters highlight the threat of poisoning attacks, and specifically that of untargeted poisoning attacks, to federated learning (FL) and discuss the robust aggregation rules (AGRs) proposed in prior literature that claim to defend federated learning from such poisoning. However, in 4, we demonstrate the fallacy of these claims and motivate the need for more sophisticated FL poisoning defenses.

To this end, in this chapter, we present two defense mechanisms (robust AGRs) that defend FL against untargeted poisoning: *Divide-n-Conquer (DnC)* and *Federated Rank Learning (FRL)*. The intuition behind our defenses stems from the lessons (Section 4.5.1) learned from our poisoning framework, e.g., both defenses address the curse of dimensionality in certain way:

**Curse of dimensionality in FL:** As explained by the Lesson-1 in Section 4.5.1, the theoretical error bounds of previous robust AGRs [30, 108, 168, 59, 95, 13] depend on the dimensionality of their inputs. Hence, the theoretical as well as empirical errors of these defenses explode for high dimensional gradients of neural networks [42] in FL. Therefore, *reducing the dimensionality of input gradients is necessary to improve robustness against poisoning.*

Similarly, to satisfy the constraint of Lesson-2, our defenses provide theoretical robustness guarantees in terms of removal of malicious clients (for DnC) and optimality of model updates (for FRL), as opposed to convergence based robustness guarantees of prior defenses. None of our defenses are based on distance-wise filtering/pruning

(Lesson-3) Next, we discuss each of the three defenses in details. In particular, for each of the defenses, we provide an overview, intuition, methodology, and finally present empirical robustness evaluations.

## 5.1 Divide-n-Conquer (DnC)

### 5.1.1 Intuition

Our intuition behind DnC is based on the lessons from Section 4.5.1. To address L3, DnC leverages singular value decomposition (SVD) based spectral methods to detect and remove outliers. Previous works [153, 57, 28] have demonstrated the theoretical and empirical performance of these methods in mitigating data poisoning against centralized learning. To address L2, we provide theoretical analysis of our defense in Section 5.1.3, giving guarantees on the removal of malicious gradients under certain conditions. Furthermore, we also construct adaptive attacks against DnC in Section 5.1.4 to provide empirical evidence of its robustness.

Note that SVD-based defenses require  $\mathcal{O}(d^3)$  memory and computational cost, hence, performing SVD directly on high dimensional gradients in common FL settings [104] is prohibitively expensive. To address this issue and the curse of dimensionality (L1), DnC reduces dimensionality through random sampling on its input gradients.

### 5.1.2 Our DnC algorithm

Algorithm 2 describes the algorithm of our DnC AGR. First, DnC randomly picks a sorted set  $r$  of indices less than the dimensionality  $d$  of its input gradients (Line-4) and constructs a subsampled set  $\tilde{\nabla}$  of gradients using  $r$  (Line-5). For instance, if  $d = 5$  and  $r = [0, 3]$ , a subsample of gradient  $\nabla_i = \{\nabla_0, \dots, \nabla_4\}$  is  $\tilde{\nabla}_i = \{\nabla_0, \nabla_3\}$ . Next, DnC computes a centered subsampled set  $\nabla^c$  of  $\tilde{\nabla}$  using dimension-wise mean  $\mu$  of  $\tilde{\nabla}$  (Lines 6-7). Then DnC computes projections of centered gradients along their

---

**Algorithm 2** Our Divide-and-Conquer AGR Algorithm

---

- 1: **Input:** Input gradients  $\nabla_{\{i \in [n]\}}$ , filtering fraction  $c$ , number of malicious clients  $m$ , niters, dimension of subsamples  $b$ , input gradients dimension  $d$
  - 2:  $\mathcal{I}_{\text{good}} \leftarrow \emptyset$
  - 3: **while**  $i < \text{niters}$  **do**
  - 4:    $r \leftarrow$  sorted set of size  $b$  of random dimensions  $\leq d$
  - 5:    $\tilde{\nabla}_{\{i \in [n]\}} \leftarrow$  set of gradients subsampled using indices in  $r$
  - 6:    $\boldsymbol{\mu} = \frac{1}{n} \sum_{i \in [n]} \tilde{\nabla}_i$  {Compute mean of input gradients}
  - 7:    $\nabla^c = \tilde{\nabla}_{\{i \in [n]\}} - \boldsymbol{\mu}$  { $\nabla^c$  is a  $n \times b$  matrix of centered input gradients}
  - 8:   Compute  $v$ , the top right singular eigenvector of  $\nabla^c$
  - 9:   Compute *outlier scores* defined as  $s_i = (\langle \nabla_i - \boldsymbol{\mu}, v \rangle)^2$
  - 10:    $\mathcal{I} \leftarrow$  Set of  $(n - c \cdot m)$  indices of the gradients with lowest outlier scores from  $s$
  - 11:   Append  $\mathcal{I}$  to  $\mathcal{I}_{\text{good}}$
  - 12:    $i = i + 1$
  - 13: **end while**
  - 14:  $\mathcal{I}_{\text{final}} \leftarrow \cap \mathcal{I}_{\text{good}}$  {Compute intersection of sets in  $\mathcal{I}_{\text{good}}$  as the final set of indices}
  - 15:  $\nabla_a = \frac{1}{|\mathcal{I}_{\text{final}}|} \sum_{i \in \mathcal{I}_{\text{final}}} \nabla_i$
  - 16: **Output**  $\nabla_a$
- 

top right singular eigenvector  $v$ , computes a vector of outlier scores  $s$ , and removes  $c \cdot m$  gradients with the highest scores (Lines 8-10). The remaining gradients are added to the set of good gradients. Such niters number of good sets are computed by randomizing  $r$  to reduce dependence on a single  $r$ . Finally, DnC computes its aggregate  $\nabla_a$  as the average of the common gradients in all of the niters good sets (Lines 14-16).

### 5.1.3 Theoretical analysis

Our theoretical analysis of DnC provides guarantees on the removal of malicious gradients and leverages the analysis of SVD based defenses against data poisoning in the centralized settings [57, 153, 59, 97]. Definition 1 from [153, 97] defines a condition  *$\epsilon$ -spectral separability* under which two distributions can be separated using spectral methods, e.g., SVD.

**Definition 1. ( *$\epsilon$ -spectral separability*)** Consider  $0 < \epsilon < 0.5$  and two finite covariance distributions,  $B$  and  $M$ . Let  $U = (1 - \epsilon)B + \epsilon M$  be a mixture of samples



from  $B$  and  $M$ , and denote the top right singular eigenvector of  $U$  by  $v$ . Then  $B$  and  $M$  are  $\epsilon$ -spectrally separable if there exists  $t$  such that

$$\begin{aligned} Pr_{X \sim B} [|\langle X - \mu_U, v \rangle| > t] &< \epsilon \\ Pr_{X \sim M} [|\langle X - \mu_U, v \rangle| < t] &< \epsilon \end{aligned}$$

If we consider that  $B$  and  $M$  (the distributions of benign and malicious gradients, respectively) are  $\epsilon$ -spectrally separable, then by removing  $\epsilon$ -fraction of gradients with maximum projections along the top eigenvector direction we can remove malicious gradients from a set of benign and malicious gradients. Lemma 1 formalizes the theoretical filtering guarantees of DnC.

**Lemma 1.** Consider  $0 < \epsilon < 0.5$  and two distributions  $B, M$  with means  $\mu_B, \mu_M$  and covariances  $\Sigma_B, \Sigma_M \preceq \sigma^2 I$ . Let  $U = (1 - \epsilon)B + \epsilon M$  be a mixture of samples from  $B$  and  $M$ . Then  $B$  and  $M$  are  $\epsilon$ -spectrally separable if  $\|\mu_B - \mu_M\|_2^2 \geq \frac{6\sigma^2}{\epsilon}$ .

For our FL poisoning setting, Lemma 1 implies that if the means of poisoned and benign gradients are sufficiently separated, then the two types of gradients can be reliably separated using spectral methods. Figure 5.1, demonstrates this exactly: the means of malicious gradients which effectively poison FL are sufficiently far from the means of benign gradients, and therefore, spectral methods can filter them. On the other hand, the malicious gradients which circumvent the criterion given in Lemma 1 have no impact on the accuracy of global model. We note that the result in Lemma 1 is common to SVD based outliers detection [57, 153, 59, 97, 43]; we provide it here for completeness and to give the intuition about the efficiency of DnC. Please check [132] for complete proof of Lemma 1.

### 5.1.4 An adaptive attack against DnC

DnC provides provable theoretical guarantees on detection of malicious gradients. However, to provide empirical evidence on the robustness guarantees of DnC, we propose an adaptive attack by against the strongest `agr-updates` adversary who has the complete knowledge of the gradients of benign devices and of DnC.

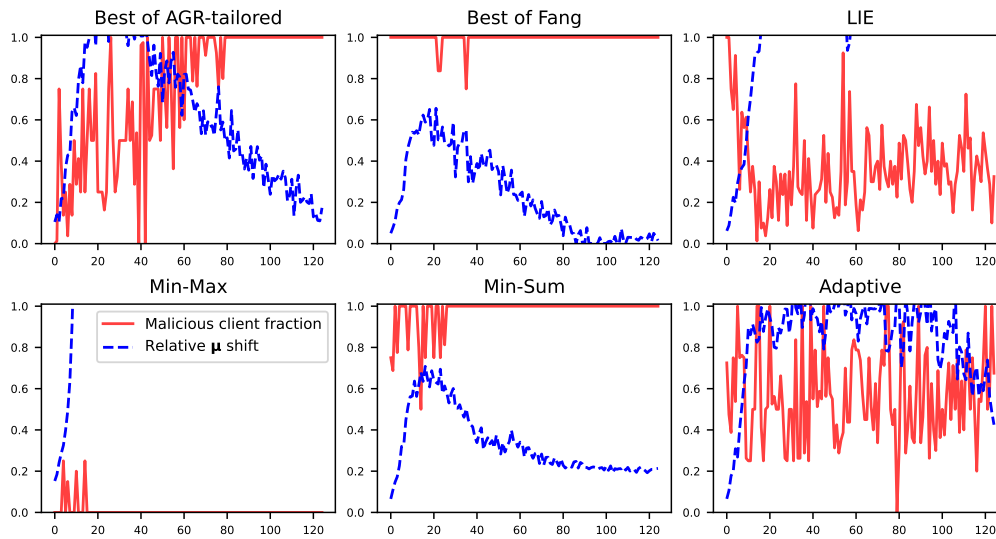
Our adaptive attack is based on the general optimization framework proposed in Section 4.2.1. The attack is inspired by our AGR-tailored attack on Multi-krum AGR, because both DnC and Multi-krum compute a selection set and average the gradients in the final selection set. The intuition of the attack is to maximize the number of malicious gradients selected by DnC to maximize the bad impact on the final aggregate. This also ensures that the number of benign gradients selected and their good impact on the final aggregate are minimized. Hence, our adaptive attack’s optimization is:

$$\begin{aligned} \operatorname{argmax}_{\gamma} m &= |\{\nabla \in \nabla_{\{i \in [m]\}}^m \mid \nabla \in \nabla_{\{i \in \mathcal{I}_{\text{final}}\}}^m\}| & (5.1) \\ \nabla_{i \in [m]}^m &= f_{\text{avg}}(\nabla_{\{i \in [n]\}}) + \gamma \nabla^p \end{aligned}$$

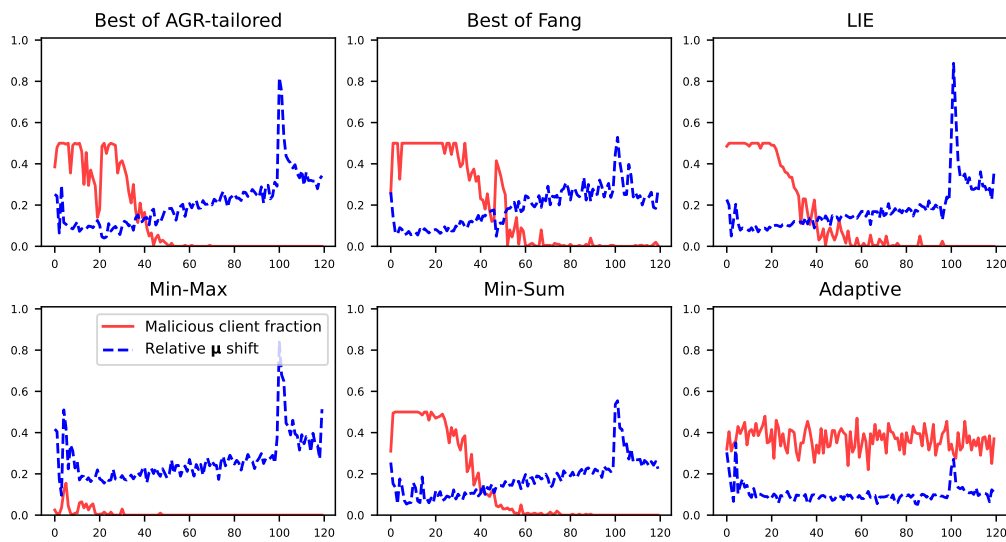
where  $m$  is the number of malicious clients,  $\mathcal{I}_{\text{final}}$  is the final set of candidate indices selected by DnC,  $\nabla^p$  is perturbation, and  $\gamma$  is scaling factor. Note that, it is reasonable to assume that although the adversary knows DnC algorithm thoroughly, she cannot know the exact random indices  $r$  from Algorithm 2. Finally, we solve the optimization in (5.1) by finding the most impactful  $\gamma$  using Algorithm 1.

### 5.1.5 Evaluation of Our Defense

In this section, we first demonstrate the robustness of our DnC AGR against state-of-the-art [64, 22] and our model poisoning attacks from Sections 4.2.1 and 5.1 for iid datasets. We also analyze the effectiveness of spectral separability, and therefore of



(a) MNIST with fully connected network



(b) CIFAR10 with Alexnet

Figure 5.1: DnC selects high fractions of malicious gradients (red plots) iff the distances between  $\mu_B$  and  $\mu_M$ , the means of benign and malicious gradients, are low (blue plots), i.e., poisoning impact of the malicious gradients is low. Upper row is for MNIST and lower row is for CIFAR10 + Alexnet. We use the strongest full knowledge agr-updates adversary.

Table 5.1: Our robust DnC AGR defends against all the existing model poisoning attacks for independently and identically distributed datasets. We consider the adversaries with complete knowledge of gradients of benign clients with 20% malicious clients. For each attack, we report its attack impact on DnC and on the existing defense with the highest global model accuracy  $A_\theta^*$ , computed as  $(A_\theta - I_\theta)$  from Table 4.2.

Dataset (Model)	No attack ( $A_\theta$ )	Fang	LIE	Best of our AGR-tailored attacks	Our AGR-agnostic attacks		Adaptive attack
					Min-Max	Min-Sum	
CIFAR10 (Alexnet)	67.6	<b>3.2</b> (7.0)	<b>3.0</b> (5.9)	<b>4.3</b> (36.8)	<b>3.5</b> (27.8)	<b>2.0</b> (16.9)	<b>6.1</b>
CIFAR10 (VGG11)	75.5	<b>3.3</b> (8.5)	<b>1.7</b> (6.8)	<b>3.4</b> (32.5)	<b>2.5</b> (21.9)	<b>2.2</b> (10.4)	<b>6.3</b>
Purchase (FC)	92.0	0.8 ( <b>0.2</b> )	<b>0.5</b> (0.5)	<b>0.9</b> (3.4)	<b>0.6</b> (1.4)	<b>0.8</b> (0.7)	<b>1.8</b>
MNIST (FC)	96.2	<b>0.1</b> (0.3)	<b>0.2</b> (0.5)	<b>1.8</b> (2.5)	<b>0.2</b> (1.2)	<b>1.2</b> (2.2)	<b>1.9</b>

Table 5.2: Results of empirical robustness analysis of DnC for *cross-device FL* setting. We consider the adversaries with complete knowledge of gradients of benign clients with 20% malicious clients, and report  $A_\theta^*$  as described in Table 4.2.

Dataset (Model)	No attack ( $A_\theta$ )	Fang	LIE	Best of our AGR-tailored attacks	Our AGR-agnostic attacks		Adaptive attack
					Min-Max	Min-Sum	
CIFAR10 (Alexnet)	64.6	<b>0.6</b> (1.6)	<b>0.3</b> (3.8)	<b>0.2</b> (14.0)	<b>0.3</b> (3.8)	<b>0.0</b> (2.6)	<b>3.4</b>
CIFAR10 (VGG11)	72.1	<b>0.8</b> (1.4)	<b>0.3</b> (2.3)	<b>2.0</b> (8.7)	<b>0.4</b> (1.8)	<b>0.4</b> (1.5)	<b>4.1</b>

Table 5.3: For non-iid FEMNIST dataset, DnC cannot mitigate our attacks in the worst case settings when the adversary knows gradients of the benign devices. But, mitigates all the attacks in the more practical settings when the gradients of benign devices are unknown. We report  $I_\theta$  on DnC of all adversaries in Table 4.1 with 20% malicious clients. ‘No attack’ accuracy  $A_\theta$  for FEMNIST with DnC is 86.6%. MMax and MSum refer to Min-Max and Min-Sum attacks, respectively.

AGR	Gradients of benign devices are known				Gradients of benign devices are unknown			
	Best of AGR-tailored (agr-updates)	AGR-agnostic (updates-only)		Adaptive attack	Best of AGR-tailored (agr-only)	AGR-agnostic (agnostic)		Adaptive attack
		MMax	MSum			MMax	MSum	
DnC	48.1	13.8	79.3	78.6	12.7	9.3	11.7	10.2
DnC + resampling	79.3	80.5	45.9	77.6	77.5	79.1	43.4	70.6

DnC, in defending against model poisoning on FL. Finally, we discuss the effectiveness of DnC for non-iid FEMNIST dataset.

### 5.1.5.1 Robustness of DnC for iid data

For iid datasets, i.e., MNIST, CIFAR10, and Purchase, we evaluate DnC against a strong adversarial setting with 20% malicious clients and the adversaries with complete knowledge of the gradients of benign clients, i.e., `agr-updates` when AGR is known and `updates-only` when AGR is unknown. We evaluate DnC using Fang and LIE, and our stronger AGR-tailored and AGR-agnostic attacks. For all these datasets, we set `niters`, `c`, and `b` in Algorithm 2 to 1, 1, and 10,000, respectively.

**Robustness comparison with previous AGRs.** Table 5.1 shows, for each of the attacks, the attack impact on DnC; in parentheses, we show the impact of the attack on the most of existing AGRs, e.g., for Fang attack on CIFAR10 + Alexnet, Bulyan is the most robust AGR, hence, for CIFAR10 + Alexnet, we show the impact of Fang attack on Bulyan.

Below, *we analyze the AGRs based on the increase in accuracy of the global model under the strongest of the attacks*, i.e., based on the minimum  $A_\theta^*$  (Section 4.3.2) for the AGR. For an AGR, the minimum  $A_\theta^*$  is obtained by subtracting the impact of the strongest attack,  $I_\theta$ , from ‘No attack’ accuracy,  $A_\theta$ . For instance, for CIFAR10 + Alexnet, our adaptive attack is the strongest attack against DnC and the corresponding minimum  $A_\theta^*$  is 61.5% (as  $A_\theta$  is 67.6% and the maximum  $I_\theta$  is 6.1%). While our AGR-tailored attack is the strongest attack against the best of the existing AGRs, thus the minimum  $A_\theta^*$  is 30.8% (as  $A_\theta$  is 67.6% and the maximum  $I_\theta$  is 32.5%). Hence, **for CIFAR10 + Alexnet, DnC increases  $A_\theta^*$  from 30.8% to 61.5% ( $\sim 100\%$  increase)**. For CIFAR10 + VGG11, DnC increases the minimum  $A_\theta^*$  from 43.0% to 69.2% ( $\sim 150\%$  increase). For Purchase, DnC increase the minimum  $A_\theta^*$  from 88.6% to 90.2%.

DnC increases the minimum  $A_\theta^*$  for MNIST from 90.7% (93.2% – 2.5%) to 94.3% (96.2% – 1.9%). Although, the absolute increase due to DnC is small for MNIST, it is significant due to the simplicity of the tasks.

**Robustness comparisons under cross-device FL setting.** Now we compare robustness of previous AGRs and our DnC when cross-device FL is used. We use CIFAR10 dataset with Alexnet and VGG11 architectures; Table 5.2 shows the results in similar fashion as Table 5.1. As before, we analyze robustness of an AGR based on the minimum  $A_\theta^*$  for the AGR. We note that the impact of attacks on DnC reduces in cross-device FL, as for the other AGRs. **For CIFAR10 + Alexnet, with no attack accuracy,  $A_\theta$ , of 64.6%, DnC increases  $A_\theta^*$  from 50.6% to 61.2%:** for best of existing AGRs, our AGR-tailored attack is the strongest attack with  $I_\theta$  of 14.0%, i.e.,  $A_\theta^*$  ( $A_\theta - I_\theta$ ) of 50.6%. While for DnC, our adaptive attack is the strongest with  $I_\theta$  of 3.4%, i.e.,  $A_\theta^*$  of 61.2%. Similarly, for CIFAR10 + VGG11, DnC increases  $A_\theta^*$  from 63.4% to 68.0%. The increase in  $A_\theta^*$  due to DnC in cross-device FL is lower, because the impact of attacks on previous AGRs is lower, which leaves smaller room for improvements.

**Why DnC is superior?** *The strong robustness of DnC stems from the effective filtering guarantees of Lemma 1*, which we empirically confirm in Figure 5.1: Here, in each epoch of FL + DnC training, we compute the fraction of malicious clients DnC selects and the norm of the difference between means of benign and malicious gradients relative to the norm of mean of the benign gradients, i.e.  $\mu_{\text{shift}} = \frac{\|\mu_B - \mu_M\|_2^2}{\|\mu_B\|_2^2}$ . We then average these entities over a few epochs for presentation clarity.

We observe in Figure 5.1 that for MNIST (upper row), Fang and Min-Sum attacks evade DnC’s detection, but have very small  $\mu_{\text{shift}}$  which leads to high accuracy of global model  $A_\theta^*$ . DnC mitigates LIE even when LIE evades DnC’s detection to some extent and introduces large  $\mu_{\text{shift}}$ . We suspect that, this is because LIE uses

ineffective perturbation  $\nabla_{\text{std}}^b$  for MNIST as explained in [132]. Our AGR-tailored and adaptive attacks evade DnC’s detection to some extent by maintaining low  $\mu_{\text{shift}}$ . Hence, MNIST, due to its simplicity, withstands their poisoning impact.

For CIFAR10 + Alexnet, we observe that DnC effectively filters malicious gradients of all but our adaptive attack. However, the adaptive attack manages to evade DnC’s detection only due to low  $\mu_{\text{shift}}$ , which is insufficient to poison DnC based FL.

### 5.1.5.2 Robustness of DnC for non-iid data

Table 5.3 shows the evaluation of DnC for FEMNIST, an imbalanced and non-iid datasets. We set `niters`, `c`, and `b` in Algorithm 2 to 1,  $(n - 1)/n$ , and 10,000, respectively. We note that, ***DnC cannot defend at least one of our attacks by the strongest adversaries with complete knowledge of the gradients of benign clients***, i.e., `agr-updates` and `updates-only` adversaries. Min-sum has attack impact of 79.3%, i.e., it reduces the accuracy from 86.6% in the benign setting to 7.3%. Here, We omit evaluation of DnC against Fang and LIE attacks, as they are strictly weaker than all of our attacks against FEMNIST.

Furthermore, resampling [73], a mechanism proposed to reduce non-iid nature of the input gradients, exacerbates DnC’s robustness (also of existing AGRs as shown in Table 5.4).

Even the benign gradients of FEMNIST with highly non-iid nature, do not point in a single direction, and therefore, DnC cannot reliably detect malicious gradients. This allows adversaries to easily circumvent DnC’s detection and mount strong attacks. ***However, DnC mitigates all of the model poisoning on FL by more practical adversaries who do not know the gradients on benign devices***, i.e., `agr-only` and `agnostic` adversaries. The maximum attack impact is only 12.7%, i.e., the maximum accuracy of the global model due to DnC is 73.9%. The most robust of existing AGRs is Krum and the corresponding maximum accuracy is 66.4%.

Table 5.4: Resampling [73] significantly reduces the robustness of existing defenses against our attacks (20% malicious clients) for all the adversaries from Table 4.1. Because, resampling increases the number of malicious updates processed, and therefore, their poisoning impacts.

AGR	No attack ( $A_\theta$ )	Updates of benign devices are known			Updates of benign devices are unknown		
		AGR-tailored	AGR-agnostic		AGR-tailored	AGR-agnostic	
			MMax	MSum		MMax	MSum
Krum	69.3	30.0	0.1	9.8	2.9	1.1	8.0
Krum-RS		<b>47.8</b>	<b>64.0</b>	43.6	<b>45.2</b>	<b>63.6</b>	58.0
Bulyan	86.1	41.0	20.1	40.0	40.5	18.7	<b>30.4</b>
Bulyan-RS		<b>76.8</b>	<b>81.7</b>	53.9	<b>49.6</b>	10.0	8.1
Trmean	86.7	24.3	29.7	26.8	20.1	24.7	25.2
Trmean-RS		<b>70.3</b>	<b>81.2</b>	46.0	<b>46.4</b>	<b>80.6</b>	68.5

We note that, defending the real-world non-iid FL settings from the worst case model poisoning attacks is a well-known challenging task [64, 73, 68] and a limitation of DnC in its current form. We leave investigating further to improve DnC to make it robust to the non-iid settings to future work.

## 5.2 Federated Rank Learning (FRL)

In this section, we present federated rank learning (FRL) a novel defense against FL poisoning. We first present a brief overview of FRL followed by our intuition behind FRL. Then we detail FRL method and its evaluation.

**High level overview of FRL:** Figure 5.2 shows how the poisoning adversary searches for malicious updates in the space of possible updates to maximize the distance between benign and malicious aggregates (Section 4.2). When the server’s AGR is not robust, e.g., dimension-wise average [104], there is no limitation on the adversary’s choices, so they can maximize their goal using a malicious update arbitrarily far from benign updates; (Figure 5.2-(a)). Therefore, even a single malicious client can jeopardize the accuracy of the global model trained using FedAvg [30]. Current robust AGRs, such as Multi-krum [30] or Trimmed-mean [168] limit the space of acceptable updates, i.e., the safe zone shown in Figure 5.2-(b). These robust AGRs



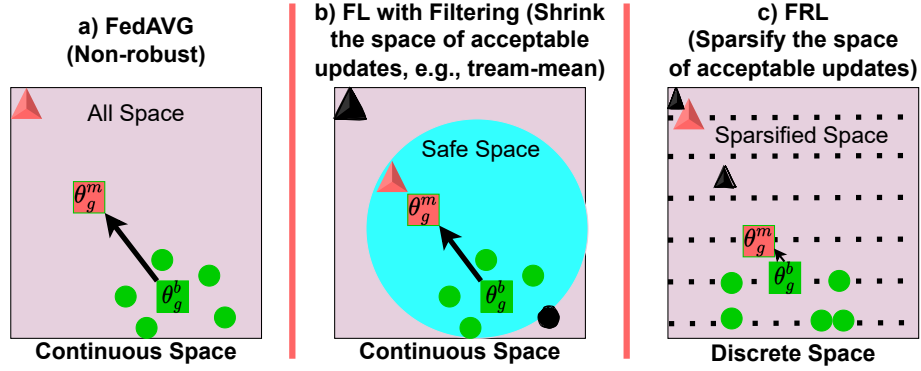


Figure 5.2: The space of client updates. Green circles represent benign updates and red triangles represent malicious updates. To defend against poisoning, existing robust AGRs filter the updates by creating a safe space (continuous  $\in \mathbb{R}^d$ ). On the other hand, FRL limits the choices of clients by enforcing a discrete space of updates (a permutation of integers  $\in [1, d]$ ).  $\theta_g^b$  (green square) demonstrates the aggregated model for benign users, and  $\theta_g^m$  (red square) demonstrates the aggregated model considering malicious updates. Black objects are updates that are ruled out by the server.

only consider the updates that are in the safe zone and thereby reduce the adversary’s choices.

Figure 5.2-(c) shows how FRL limits the poisoning adversary’s choices of malicious updates by making the space of acceptable updates *discrete*. FRL uses a novel learning paradigm called *supermasks* training [173, 126] to create edge rankings, which, as we will show, allows FRL to reduce communication costs while achieving significantly stronger robustness. Specifically, in FRL, clients collaborate to *find a subnetwork* within a *randomly initialized* neural network which we call the *supernet* (this is in contrast to conventional FL where clients collaborate to *train a neural network*). The goal of training in FRL is to collaboratively rank the supernet’s edges based on the importance of each edge and find a *global ranking*. The global ranking can be converted to a supermask, which is a binary mask of 1’s and 0’s, that is superimposed on the random neural network (the supernet) to obtain the final subnetwork. For example, in our experiments, the final subnetwork is constructed using the top

50% of all edges. The subnetwork is then used for downstream tasks, e.g., image classification, hence it is equivalent to the global model in conventional FL. Note that in entire FRL training, weights of the supernetwork *do not* change.

More specifically, each FRL client computes the importance of the edges of the supernetwork based on their local data. The importance of the edges is represented as a ranking vector. Each FRL client will use the *edge popup* algorithm [126] and their data to compute their local rankings (the edge popup algorithm aims at learning which edges in a supernetwork are more important over the other edges by minimizing the loss of the subnetwork on their local data). Each client then will send their local edge ranking to the server. Finally, the FRL server uses a novel *voting* mechanism to aggregate client rankings into a global ranking vector, which represents which edges of the random neural network (the supernetwork) will form the global subnetwork.

***Intuitions on FRL’s robustness:*** In traditional FL algorithms, clients send high dimensional model updates  $\in \mathbb{R}^d$  (real numbers) to the server, providing malicious clients significant flexibility in fabricating malicious updates. By contrast, FRL clients merely share the rankings of the edges of the supernetwork, i.e., integers  $\in [1, d]$ , where  $d$  is the size of the supernetwork. This allows the FRL server to use a voting mechanism to aggregate client updates (i.e., ranks), therefore, providing high resistance to malicious ranks submitted by poisoning clients, since each client can only cast a single vote! Therefore, as we will show both theoretically and empirically, FRL provides robustness by design and reduces the impact of untargeted poisoning attacks. Furthermore, *unlike most existing robust FL frameworks, FRL does not require any knowledge about the percentages of malicious clients.*

## 5.2.1 Preliminaries for FRL

### 5.2.1.1 Edge-popup algorithm

The edge-popup (EP) algorithm [126] is an optimization to find supermasks within a large, randomly initialized neural network, i.e., called *supernetwork*, with performances close to the *fully trained* supernetwork. EP algorithm does not train the weights ( $\theta^w$ ) of the network, instead only decides the set of edges to keep and removes (pops) the rest of the edges. Specifically, EP algorithm assigns a positive score to each of the edges in the supernetwork ( $\theta^s$ ). On forward pass, it selects top  $k\%$  edges with highest scores, where  $k$  is the percentage of the total number of edges in the supernetwork that will remain in the final subnetwork. On the backward pass, it updates the scores with the straight-through gradient estimator [24].

Algorithm 3 presents EP algorithm. Suppose in a fully connected neural network, there are  $L$  layers and layer  $\ell \in [1, L]$  has  $n_\ell$  neurons, denoted by  $V^\ell = \{V_1^\ell, \dots, V_{n_\ell}^\ell\}$ . If  $I_v$  and  $Z_v$  denote the input and output for neuron  $v$  respectively, then the input of the node  $v$  is the weighted sum of all nodes in previous layer, i.e.,  $I_v = \sum_{u \in V^{\ell-1}} W_{uv} Z_u$ . Here,  $W_{uv}$  is the weight of the edge connecting  $u$  to  $v$ . Edge-popup algorithm tries to find subnetwork  $E$ , so the input for neuron  $v$  would be:  $I_v = \sum_{(u,v) \in E} W_{uv} Z_u$ .

**Updating scores:** Consider an edge  $E_{uv}$  that connects two neurons  $u$  and  $v$ ,  $W_{uv}$  be the weight of  $E_{uv}$ , and  $s_{uv}$  be the score assigned to the edge  $E_{uv}$  by Edge-popup algorithm. Then the edge-popup algorithm removes edge  $E_{uv}$  from the supermask if its score  $s_{uv}$  is not high enough. Each iteration of supermask training updates the scores of all edges such that, if having an edge  $E_{uv}$  in subnetwork reduces loss (e.g., cross-entropy loss) over training data, the score  $s_{uv}$  increases.

---

**Algorithm 3** Edge-popup (EP) algorithm

---

```
1: Input: number of local epochs  $E$ , training data  $D$ , initial weights  $\theta^w$  and scores  $\theta^s$ , subnetwork size  $k\%$ , learning rate  $\eta$ 
2: for  $e \in [E]$  do
3:    $\mathcal{B} \leftarrow$  Split  $D$  in  $B$  batches
4:   for batch  $b \in [B]$  do
5:     EP FORWARD ( $\theta^w, \theta^s, k, b$ )
6:      $\theta^s = \theta^s - \eta \nabla \ell(\theta^s; b)$ 
7:   end for
8: end for
9: return  $\theta^s$ 
10: Function EP forward( $\theta^w, \theta^s, k, b$ )
11:  $\mathbf{m} \leftarrow$  sort( $\theta^s$ )
12:  $t \leftarrow$  int( $(1 - k) * \text{len}(\mathbf{m})$ )
13:  $\theta^p = \theta^w \odot \mathbf{m}$ , where  $\mathbf{m}[t] = 0$ ;  $\mathbf{m}[t:] = 1$ 
14: return  $\theta^p(b)$ 
15: End Function
```

---

The algorithm selects top  $k\%$  edges (i.e., finds a subnetwork with sparsity of  $k\%$ ) with highest scores, so  $I_v$  reduces to  $I_v = \sum_{u \in V^{\ell-1}} W_{uv} Z_u h(s_{uv})$  where  $h(\cdot)$  returns 1 if the edge exists in top- $k\%$  highest score edges and 0 otherwise. Because of existence of  $h(\cdot)$ , which is not differentiable, it is impossible to compute the gradient of loss with respect to  $s_{uv}$ . Recall that, the Edge-popup algorithm uses straight-through gradient estimator [24] to compute gradients. In this approach,  $h(\cdot)$  will be treated as the identity in the backward pass meaning that the upstream gradient (i.e.,  $\frac{\partial L}{\partial I_v}$ ) goes straight-through  $h(\cdot)$ . Now using chain rule, we can derive  $\frac{\partial L}{\partial I_v} \frac{\partial I_v}{\partial s_{uv}} = \frac{\partial L}{\partial I_v} W_{uv} Z_u$

where  $L$  is the loss to minimize. Then we can SGD with step size  $\eta$  to update scores as  $s_{uv} \leftarrow s_{uv} - \eta \frac{\partial L}{\partial I_v} Z_u W_{uv}$ .

### 5.2.2 Details of federated rank learning (FRL)

Algorithm (Algorithm 4) details FRL algorithm. FRL clients collaborate (without sharing their data) to *find a subnetwork* within a *randomly initialized, untrained supernetwork*, with scores  $\theta^s$  and weights  $\theta^w$ . In each round, FRL first finds a unanimous (global) ranking of the supernetwork edges and then uses the subnetwork of the top ranked edges as the global model.

---

**Algorithm 4** Federated Ranking Learning (FRL)

---

- 1: **Input:** number of rounds  $T$ , number of local epochs  $E$ , number of users per round  $n$ , seed  $SEED$ , learning rate  $\eta$ , subnetwork size  $k\%$
  - 2: **Server: Initialization**
  - 3:  $\theta^s, \theta^w \leftarrow$  Initialize random scores and weights using  $SEED$
  - 4:  $R_g^1 \leftarrow \text{ARGSORT}(\theta^s)$  {Sort the initial scores and obtain initial rankings}
  - 5: **for**  $t \in [1, T]$  **do**
  - 6:    $U \leftarrow$  set of  $n$  randomly selected clients out of  $N$  total clients
  - 7:   **for**  $u$  in  $U$  **do**
  - 8:     **Clients: Calculating the ranks**
  - 9:      $\theta^s, \theta^w \leftarrow$  Initialize scores and weights using  $SEED$
  - 10:      $\theta^s[R_g^t] \leftarrow \text{SORT}(\theta^s)$  {sort the scores based on the global ranking}
  - 11:      $S \leftarrow \text{Edge-PopUp}(E, D_u^{tr}, \theta^w, \theta^s, k, \eta)$  {Client  $u$  uses Algorithm3 to train a supermask on its local training data}
  - 12:      $R_u^t \leftarrow \text{ARGSORT}(S)$  {Ranking of the client}
  - 13:   **end for**
  - 14:   **Server: Majority Vote**
  - 15:    $R_g^{t+1} \leftarrow \text{VOTE}(R_{\{u \in U\}}^t)$  {Majority vote aggregation}
  - 16: **end for**
  - 17: **Function**  $\text{Vote}(R_{\{u \in U\}})$ :
  - 18:    $V \leftarrow \text{SUM}(\text{ARGSORT}(R_{\{u \in U\}}))$ ,  $A \leftarrow \text{SUM}(V)$
  - 19:   **return**  $\text{ARGSORT}(A)$
  - 20: **End Function**
- 

The objective of FRL is to find a global ranking  $R_g$  and convert it to a global binary mask,  $\mathbf{m}$ , such that resulting subnetwork,  $\theta^w \odot \mathbf{m}$ , minimizes the average loss of all clients. FRL optimization can be formalized as follows:

$$\begin{aligned} \min_{R_g} F(\theta^w, R_g) &= \min_{R_g} \sum_{i=1}^N \lambda_i L_i(\theta^w \odot \mathbf{m}) & (5.2) \\ \text{s.t. } \mathbf{m}[R_g < k] &= 0 \text{ and } \mathbf{m}[R_g \geq k] = 1 \end{aligned}$$

where  $N$  is the total number of FRL clients,  $L_i$  is the loss function for the  $i^{\text{th}}$  client,  $\lambda_i$  is the importance, e.g., weight, of the  $i^{\text{th}}$  client; we use  $\lambda_i = \frac{1}{N}$ , i.e., all clients have the same weight.  $\mathbf{m}$  is the final binary mask, where edges with top  $k$  ranks (layer-wise) get '1' while others get '0'. We use  $\mathbf{m}$  to compute final global model by superimposing  $\mathbf{m}$  on  $\theta$ , i.e., the we use the subnetwork  $\theta \odot \mathbf{m}$  as the final global model. In [111], we

show that probability of encountering a disconnected subnetwork is very negligibly small, and also demonstrate what happens if the subnetwork becomes disconnected for a small sparsity  $k$ . In Figure 5.3, we demonstrate a single FRL round using a supernetwork with six edges  $e_{i \in [0,5]}$  and three clients  $C_{j \in [1,3]}$  who aim to find a subnetwork of size  $k=50\%$  of the original supernetwork. In Appendix B of [111], we show how FRL minimizes its objective and is independent of the downstream task.

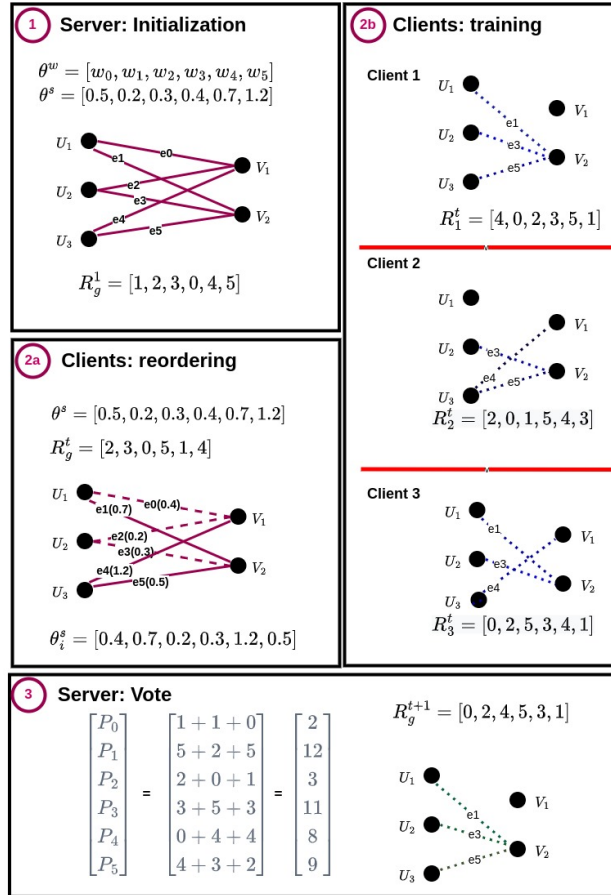


Figure 5.3: A single FRL round with three clients and supernetwork of 6 edges.

### 5.2.2.1 Server: Initialization (only for round $t = 1$ )

In the first round, the FRL server chooses a random seed SEED to generate initial random weights  $\theta^w$  and scores  $\theta^s$  for the global supernetwork  $\theta$ ; note that,  $\theta^w$ ,  $\theta^s$ , and SEED remain constant during the entire FRL training. Next, the FRL server shares

SEED with FRL clients, who can then locally reconstruct the initial weights  $\theta^w$  and scores  $\theta^s$  using SEED. Figure 5.3-① depicts this step.

Recall that, the goal of FRL training is to find the most important edges in  $\theta^w$  without changing the weights. Unless specified otherwise, both server and clients use the Signed Kaiming Constant algorithm [126] to generate random weights and the Kaiming Uniform algorithm [72] to generate random scores. However, in [111], we also explore the impacts of different weight initialization algorithms on the performance of FRL. We use the same seed to initialize weights and scores.

At the beginning, the FRL server finds the global rankings of the initial random scores (Algorithm 4 line 4), i.e.,  $R_g^1 = \text{ARGSORT}(\theta^s)$ . We define *rankings of a vector* as the indices of elements of vector when the vector is sorted from low to high, which is computed using ARGSORT function.

### 5.2.2.2 Clients: Calculating the ranks (for each round $t$ )

In the  $t^{\text{th}}$  round, FRL server randomly selects  $n$  clients among total  $N$  clients, and shares the global rankings  $R_g^t$  with them. Each of the selected  $n$  clients locally reconstructs the weights  $\theta^w$ 's and scores  $\theta^s$ 's using SEED (Algorithm 4 line 9). Then, each FRL client reorders the random scores based on the global rankings,  $R_g^t$  (Algorithm 4 line 10); we depict this in Figure 5.3-②a).

Next, each of the  $n$  clients uses reordered  $\theta^s$  and finds a subnetwork within  $\theta^w$  using Algorithm 3; to find a subnetwork, they use their local data and  $E$  local epochs (Algorithm 4 line 11). Note that, each iteration of Algorithm 3 updates the scores  $S$  starting from  $\theta^s$ . Then client  $u$  computes their local rankings  $R_u^t$  using the final updated scores ( $S$ ) and  $\text{ARGSORT}(\cdot)$ , and sends  $R_u^t$  to the server. Figure 5.3-②a) shows how each of the selected  $n$  clients reorders the random scores using global rankings. For instance, the initial global rankings for this round are  $R_g^t = [2, 3, 0, 5, 1, 4]$ , mean-

ing that edge  $e_4$  should get the highest score ( $s_4 = 1.2$ ), and edge  $e_2$  should get the lowest score ( $s_2 = 0.2$ ).

Figure 5.3-(2b) shows, for each client, the scores and rankings they obtained after finding their local subnetwork. For example, rankings of client  $C_1$  are  $R_1^t = [4, 0, 2, 3, 5, 1]$ , i.e.,  $e_4$  is the least important and  $e_1$  is the most important edge for  $C_1$ . Considering desired subnetwork size to be 50%,  $C_1$  uses edges  $\{3, 5, 1\}$  in their final subnetwork.

### 5.2.2.3 Server: Majority vote (for each round $t$ )

The server receives all the local rankings of the selected  $n$  clients, i.e.,  $R_{\{u \in U\}}^t$ . Then, it performs a majority vote over all the local rankings using  $\text{VOTE}(\cdot)$  function. Note that, for client  $u$ , the index  $i$  represents the importance of the edge  $R_u^t[i]$  for  $C_u$ . For instance, in Figure 5.3-(2b), rankings of  $C_1$  are  $R_1^t = [4, 0, 2, 3, 5, 1]$ , hence the edge  $e_4$  at index=0 is the least important edge for  $C_1$ , while the edge  $e_1$  at index=5 is the most important edge. Consequently,  $\text{VOTE}(\cdot)$  function assigns reputation=0 to edge  $e_4$ , reputation=1 to  $e_0$ , reputation=2 to  $e_2$ , and so on. In other words, for rankings  $R_u^t$  of  $C_u$  and edge  $e_i$ ,  $\text{VOTE}(\cdot)$  computes the reputation of  $e_i$  as its index in  $R_u^t$ . Finally,  $\text{VOTE}(\cdot)$  computes the total reputation of  $e_i$  as the sum of reputations from each of the local rankings. In Figure 5.3-(2b), reputations of  $e_0$  are 1 in  $R_1^t$ , 1 in  $R_2^t$ , and 0 in  $R_3^t$ , hence, the total reputation of  $e_0$  is 2. We depict this in Figure 5.3-(3); here, the final total reputations for edges  $e_{\{i \in [0, 5]\}}$  are  $A = [2, 12, 3, 11, 8, 9]$ . Finally, the server computes global rankings  $R_g^{t+1}$  to use for round  $t + 1$  by sorting the final total reputations of all edges, i.e.,  $R_g^{t+1} = \text{ARGSORT}(A)$ .

Note that, *all FRL operations that involve sorting, reordering, and voting are performed in a layer-wise manner*. For instance, the server computes global rankings  $R_g^t$  in round  $t$  for each layer separately, and consequently, the clients selected in round  $t$  reorder their local randomly generated scores  $\theta^s$  for each layer separately.



### 5.2.3 Robustness of FRL to poisoning

FRL and FL are distributed learning algorithms with mutually untrusting clients. Hence, a *poisoning adversary* may own or compromise some of FRL (FL) clients, called *malicious clients*, and mount a *targeted* or *untargeted* poisoning attack. As discussed in Section 1, we mainly focus on the more severe untargeted attacks and show that FRL is significantly more robust by design to such poisoning attacks. However, for completeness we also evaluate robustness of FRL against targeted attacks in [111].

#### 5.2.3.1 Intuition behind robustness of FRL

Existing FL algorithms, including robust algorithms, are shown to be vulnerable to various poisoning attacks [134]. One of the key reasons behind the susceptibility of existing algorithms is that their model updates can have a large continuous space of values. For instance, to manipulate vanilla FedAvg, malicious clients send very large updates [30], and to manipulate Multi-krum and Trimmed-mean, [64, 132] propose to perturb a benign update in a specific malicious direction. On the other hand, in FRL, clients must send a permutation of indices  $\in [1, n_\ell]$  for each layer. Hence, FRL significantly reduces the space of the possible malicious updates that an adversary can craft. Majority voting in FRL further reduces the chances of successful attack. Intuitively, this makes FRL design robust to poisoning attacks. Below, we make this intuition more concrete.

#### 5.2.3.2 The worst-case untargeted poisoning attack on FRL

Here, the poisoning adversary aims to reduce the accuracy of the final global FRL subnetwork on most test inputs. To achieve this, the adversary should replace the high ranked edges with low ranked edges in the final subnetwork. For the worst-case analysis of FRL, we assume a very strong adversary (i.e., threat model): 1) each of the malicious clients has some data from benign distribution; 2) malicious clients

know the entire FRL algorithm and its parameters; 3) malicious clients can collude. Under this threat model we design a worst case attack on FRL (Algorithm 5), which executes as follows: First, malicious clients compute rankings on their benign data and use VOTE(.) algorithm to compute an aggregate rankings. Finally, each of the malicious clients uses the reverse of the aggregate rankings to share with the FRL server in given round. The adversary should invert the rankings layer-wise as the FRL server will aggregate the local rankings per layer too, and it is not possible to mount a model-wise attack.

---

**Algorithm 5** FRL Poisoning

---

- 1: **Input:** number of malicious clients  $M$ , number of malicious local epochs  $E'$ , seed SEED, global ranking  $R_g^t$ , learning rate  $\eta$ , subnetwork size  $k\%$
  - 2: **Function** Maliciousupdate( $M$ , SEED,  $R_g^t$ ,  $E'$ ,  $\eta$ ,  $k$ ):
  - 3: **for**  $mu \in [M]$  **do**
  - 4:   Malicious Client Executes:
  - 5:    $\theta^s, \theta^w \leftarrow$  Initialize random scores and weights using SEED
  - 6:    $\theta^s[R_g^t] \leftarrow$  SORT( $\theta^s$ )
  - 7:    $S \leftarrow$  Edge-PopUp( $E'$ ,  $D_u^{tr}$ ,  $\theta^w$ ,  $\theta^s$ ,  $k$ ,  $\eta$ )
  - 8:    $R_{mu}^t \leftarrow$  ARGSORT( $S$ ) {Ranking of the malicious client}
  - 9: **end for**
  - 10: Aggregation:
  - 11:  $R_m^t \leftarrow$  VOTE( $R_{\{mu \in [M]\}}^t$ ) {Majority vote aggregation}
  - 12: **return** REVERSE( $R_m^t$ ) {reverse the ranking}
  - 13: **End Function**
- 

Now we justify why the attack in Algorithm 5 is the worst case attack on FRL for the strong threat model. Note that, FRL aggregation, i.e., VOTE(.), computes the reputations using clients' rankings and sums the reputations of each network edge. Therefore, the strongest poisoning attack would want to reduce the reputation of good edges. This can be achieved following the aforementioned procedure of Algorithm 5 to reverse the rankings computed using benign data.

### 5.2.3.3 Theoretical analysis of robustness of FRL algorithm

In this section, we prove an upper bound on the failure probability of robustness of FRL, i.e., the probability that a good edge will be removed from the final subnetwork when malicious clients mount the worst case attack.

Following the work of [25], we make two assumptions in order to facilitate a concrete robustness analysis of FRL: a) each malicious client has access only to its own data, and b) we consider a simpler VOTE(.) function, where the FRL server puts an edge  $e_i$  in the final subnetwork if more than half of the clients have  $e_i$  (a good edge) in their local subnetworks. In other words, the rankings that each client sends to the server is just a bit mask showing that each edge should or should not be in the final subnetwork. The server makes a majority vote on the bit masks, and if an edge has more than half votes, it will be in the global subnetwork. Our VOTE(.) mechanism has more strict robustness criterion, as it uses more nuanced reputations of edges instead of bit masks. Hence, the upper bound on failure probability in this section also applies to the FRL VOTE(.) function.

The probability that our voting system fails is the probability that more than half of the votes do not include  $e_i$  in their subnetworks. The upper bound on the probability of failure would be  $1/2\sqrt{\frac{np(1-p)}{(n(p+\alpha(1-2p)-1/2))^2}}$ , where  $n$  is the number of clients being processed,  $p$  is the probability that a benign client puts  $e_i$  in their top ranks, and  $\alpha$  is the fraction of malicious clients. Please check [111] for the detailed proof. Figure 5.4 shows the upper bound on the failure of VOTE(.) for different values of  $\alpha$  and  $p$ . As can be seen, the higher the probability  $p$ , the higher the robustness of FRL.

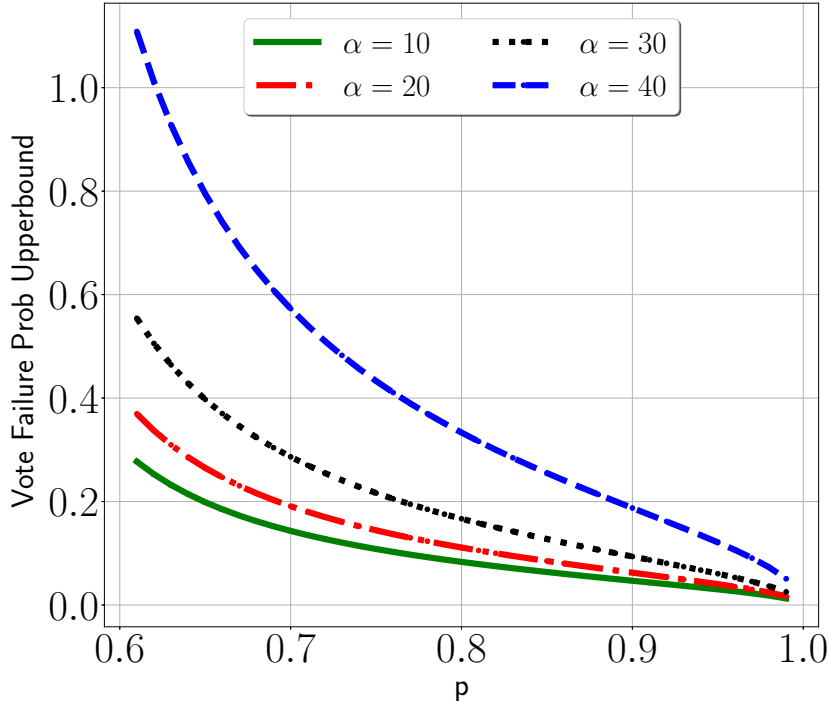


Figure 5.4: Upper bound on the failure probability of  $\text{VOTE}(\cdot)$  function in FRL.  $\alpha$  is the percentages of malicious clients and  $p$  is the probability that a benign client puts a good edge in its top  $k$  ranks.

## 5.2.4 Robustness evaluation of FRL

### 5.2.4.1 Experimental setup

**Datasets and their distribution:** We use MNIST, CIFAR10, and FEMNIST datasets. Most real-world FL settings have heterogeneous client data, hence following previous works [127, 75], we distribute MNIST and CIFAR10 datasets among 1,000 clients in non-iid fashion using *Dirichlet* distribution with parameter  $\beta = 1$ . Note that, increasing  $\beta$  results in more iid datasets. FEMNIST is naturally distributed non-iid among 3,400 clients. We further split the datasets of each client into training (80%) and test (20%).

We run all the experiments for 2000 global rounds of FRL and FL, while selecting 25 clients in each round. At the end of the training, we calculate the test accuracy

Table 5.5: State-of-the-art model architectures that we use for FRL experiments.

Architecture	Layer Name	Number of parameters
LeNet [158] (MNIST and FEMNIST)	Conv(32)	288
	Conv(64)	18432
	FC(128)	1605632
	FC(10) or FC(62)	1280 or 7936
Conv8 [126] (CIFAR10)	Conv(64), Conv(64)	38592
	Conv(128), Conv(128)	221184
	Conv(256), Conv(256)	884736
	Conv(512), Conv(512)	3538944
	FC(256), FC(256), FC(10)	592384

of all the clients on the final global model, and we report the mean and standard deviation of all clients’ test accuracies in our experiments. For MNIST and FEMNIST we use LeNet architecture and for CIFAR10 we use Conv8 architecture shown in Table 5.5. For complete hyperparameter settings, please refer to Appendix C.1 of [111]. We independently tune the hyperparameters for FRL and other baselines (Section 5.2.4.1).

**Baseline FL algorithms:** We compare the FRL with FedAvg (Section 2.1) Multi-Krum (Section 2.2.3), Trimmed-mean (Section 2.2.3) and the following two additional FL baselines:

*SignSGD* [25] is a quantization method used in distributed learning to compress each dimension of gradient updates into 1 bit instead of 32 or 64 bits. To achieve this, in SignSGD the clients only send the sign of their gradient updates to the server, and the server runs a majority vote on them. SignSGD is designed for distributed learning where all the clients participate in each round, so all the clients are aware of the most updated weight parameters of the global model. However, SignSGD only reduces upload communication (clients→server). But, does not reduce download communication (server→clients), i.e., to achieve good performance of the global model, the server sends all the weight parameters (each of 32 bits) to the newly selected clients in each round. Hence, SignSGD is as inefficient as FedAvg in download communication.

*TopK* [15, 12] is a sparsification method used in distributed learning that transmits only a few dimensions of each model update to the server. In TopK, the clients first sort the absolute values of their local model updates, and send the Top K% largest model update dimensions to the server for aggregation. TopK suffers from the same problem as SignSGD: for performance reasons, the server should send the entire updated model weights to the new selected clients.

Table 5.6: Comparing the robustness of various FL algorithms: FRL and Sparse-FRL (SFRL) (in **bold**) outperform the state-of-the-art robust AGRs and SignSGD against our strong untargeted poisoning attacks from Section 4.2.

Dataset	AGR	No malicious	10% malicious	20% malicious
MNIST + LeNet 1000 clients	FedAvg	98.8 (3.2)	10.0 (10.0)	10.0 (10.0)
	Trimmed-mean	98.8 (3.2)	95.1 (7.7)	87.6 (9.5)
	Multi-krum	98.8 (3.2)	98.6 (3.3)	97.9 (4.1)
	SignSGD	97.2 (4.6)	96.6 (5.0)	96.2 (5.6)
	<b>FRL</b>	<b>98.8 (3.1)</b>	<b>98.8 (3.1)</b>	<b>98.7 (3.3)</b>
	<b>SFRL Top 50%</b>	<b>98.2 (3.8)</b>	<b>97.04 (4.4)</b>	<b>95.1 (7.8)</b>
CIFAR10 + Conv8 1000 clients	FedAvg	85.4 (11.2)	10.0 (10.1)	10.0 (10.1)
	Trimmed-mean	84.9 (11.0)	56.3 (16.0)	20.5 (13.2)
	Multi-krum	84.7 (11.3)	58.8 (15.8)	25.6 (14.4)
	SignSGD	79.1 (12.8)	39.7 (15.9)	10.0 (10.1)
	<b>FRL</b>	<b>85.3 (11.3)</b>	<b>79.0 (12.4)</b>	<b>69.5 (14.8)</b>
	<b>SFRL Top 50%</b>	<b>77.6 (13.0)</b>	<b>41.7 (15.4)</b>	<b>39.7 (15.2)</b>
FEMNIST + LeNet 3400 clients	FedAvg	85.8 (10.2)	6.3 (5.8)	6.3 (5.8)
	Trimmed-mean	85.2 (11.0)	72.7 (15.7)	56.2 (20.3)
	Multi-krum	85.2 (10.9)	80.9 (12.2)	23.7 (12.8)
	SignSGD	79.3 (12.4)	76.7 (13.2)	55.1 (14.9)
	<b>FRL</b>	<b>84.2 (10.7)</b>	<b>83.0 (10.9)</b>	<b>65.8 (17.8)</b>
	<b>SFRL Top 50%</b>	<b>75.2 (12.7)</b>	<b>70.5 (14.4)</b>	<b>60.39 (14.8)</b>

#### 5.2.4.2 Robustness of FRL against untargeted poisoning

Here, we compare FRL with Multi-krum (Mkrum) and Trimmed-mean. Table 5.6 gives the performances of robust AGRs, SignSGD, and FRL with different percentages of malicious clients using our strong model poisoning attacks (Section 4.2), and the attacks in [25] and Algorithm 5, respectively. Here, we make a rather impractical assumption in favor of the *previous* robust AGRs: we assume that the server knows the exact % of malicious clients in each FL round. Note that, *FRL does not require this knowledge*.

**FRL achieves higher robustness than state-of-the-art robust AGRs:** We note from Table 5.6 that, FRL is more robust to the presence of malicious clients who mount untargeted poisoning attacks, compared to Multi-Krum and Trimmed-mean, when percentages of malicious clients are 10% and 20%. For instance, on CIFAR10, 10% malicious clients can decrease the accuracy of FL models to 56.3% and 58.8% for Trimmed-mean and Multi-Krum respectively; 20% malicious clients can decrease the accuracy of the FL models to 20.5% and 25.6% for Trimmed-mean and Multi-Krum respectively. On the other hand, FRL performance decreases to 79.0% and 69.5% for 10% and 20% attacking ratio, respectively.

We make similar observations for MNIST and FEMNIST datasets: for FEMNIST, 10% (20%) malicious clients reduce accuracy of the global model from 85.8% to 72.7% (56.2%) for Trimmed-Mean, and to 80.9% (23.7%) for Multi-krum, while FRL accuracy decreases to 83.0% (65.8%).

**FRL is more accurate than SignSGD:** First, we note that, in the absence of malicious clients, FRL is significantly more accurate than SignSGD. For instance, on CIFAR10 distributed in non-iid fashion among 1000 clients, FRL achieves 85.3% while SignSGD achieves 79.1% , or on FEMNIST, FRL achieves 84.2% while SignSGD achieves 79.3%. This is because, FRL clients send more nuanced information via rankings of their subnetworks compared to SignSGD, where clients just send the signs of their model updates.

**FRL is more robust than SignSGD:** Next, we note from Table 5.6 that, FRL is more robust against untargeted poisoning attacks compared to SignSGD when percentages of malicious clients are 10% and 20%. For instance, on CIFAR10, 10% (20%) malicious clients can decrease the accuracy of SignSGD model to 39.8% (10.0%). On the other hand, FRL performance decreases to 79.0% and 69.5% for 10% and 20% attacking ratio respectively. We make similar observations for MNIST and FEMNIST datasets: for FEMNIST, 10% (20%) malicious clients reduce accuracy of the global

model from 85.8% to 76.7% (55.1%) for SignSGD, while FRL accuracy decreases to 83.0% (65.8%).

**Robustness of Sparse-FRL:** We evaluate robustness of SFRL Top 50% against 10% and 20% malicious clients. As we can see from Table 5.6, by sending only top half of the local rankings, the accuracy goes from 85.3% (FRL) to 77.6% (SFRL). SFRL also can provide robustness to some extent, but adversary has more influence on the global ranking since half of the rankings are missing. For instance, on CIFAR10, 10% (20%) malicious clients can decrease the accuracy of global ranking to 41.7% (39.7%) from 77.6%. Also for FEMNIST, 10% (20%) malicious clients can decrease the accuracy of global ranking to 70.5% (60.39%) from 75.2%. We can see when malicious clients’ percentages are higher, SFRL can perform better compared to existing robust AGR.

**FRL versus FedAvg and TopK:** We omit the results of non-robust aggregations, FedAvg and TopK, because even a single malicious client [30] can jeopardize their performances.

### 5.2.4.3 Robustness of FRL against targeted poisoning

So far, we evaluated the robustness of FRL against untargeted attacks. In this section, we evaluate the robustness against targeted poisoning, and specifically against *backdoor poisoning attacks*. We consider state-of-the-art backdoor attacks of three types: *semantic* [18], *artificial* [164], and *edge-case*[155] backdoor attacks.

**Evaluation setup:** Below we detail evaluation setup we use for evaluating robustness of FRL against targeted poisoning.

**Training hyperparameters:** We compare the performances of the FRL and FedAvg against the above backdoor attacks. We evaluate the accuracy of the (poisoned) global model on the main (inputs without trigger) and backdoor (inputs with trigger) tasks. Following [155, 18], for all the experiments, we start from a pre-trained model



with 80.0% test accuracy and train it for 1,000 more FL rounds when malicious clients are present. We use  $N = 1000$  clients, distribute CIFAR10 as in Section 5.2.4.1, and randomly select  $n = 25$  clients in each round. We report the final test accuracy and the average of backdoor test accuracies over the 1000 FL rounds when percentages of malicious clients is in  $\{1, 2, 5, 10\}$ . We assume that each malicious client has some benign data (per the distribution scheme from Section 5.2.4.1) and all the backdoored data. We use the same hyperparameters discussed in Section 5.2.4.1 for training CIFAR10 on Conv8 model.

**Backdoor attack hyperparameters:** Here we provide hyperparameter details.

(1) *Model Replacement in FL backdoor Attacks:* FL backdoor poisoning attacks use a strategy called model replacement where the malicious client first finds a malicious update that contains the backdoor, then it scales the model parameters to cancel the contributions from the other honest clients. For example, if there are  $m$  malicious clients selected in FL round  $t$ , each malicious client  $u$  calculates its backdoored update  $\theta_u^t$ , and re-scaled it to  $\lambda\theta_u^t$  where  $\lambda = \frac{m}{n}$  where  $n$  is the number of selected clients in each FL round. Model replacement strategy requires that the global model is close to convergence, so the malicious clients can replace the global model with their backdoored model, which performs well on the main task. Following this strategy, the backdoor accuracy would be very high in FedAvg, even if one of the malicious clients is chosen in one round. However, in FRL, re-scaling is not possible as the clients are sending their local rankings where each ranking are a permutation of the indices of the edges in each layer, i.e., of  $[0, n_\ell - 1] \forall \ell \in [L]$  where  $L$  is the number of layers and  $n_\ell$  is the number of parameters in  $\ell$ th layer. To have a fair comparison with FedAvg, we did not use a model replacement strategy for FedAvg too.

(2) *Semantic backdoors:* We choose images with vertically striped walls in the background (Figure 5.5 (a)) as the backdoors. Of these 12 images with this trigger in the CIFAR10 dataset, we use 9 of them for training the backdoors while keeping the other

three for testing the backdoor accuracy. The malicious clients want the global model to predict these images as the bird (class label=2). We measure the backdoor accuracy on 1000 randomly rotated and cropped versions of the three backdoor images held out of the adversary’s training.

(3) *Artificial backdoors*: We add a particular pixel pattern to the top left corner of the first nine (not bird) images of the CIFAR10 dataset (Figure 5.5 (b)) and change their labels to bird (label=2). To evaluate these attacks, we pick 256 random (not bird) images from CIFAR10 and add this pattern to them with the label of class bird.

(4) *Edge-Case backdoors*: We collect 980 images from public web by searching for Southwest airplanes (similar to what [155] did) and resize the images to  $32 \times 32$  (Figure 5.5 (c)). We set their target labels as truck (class label=9). We use 784 of these images for training and keep 196 of them for the evaluation of the backdoor.

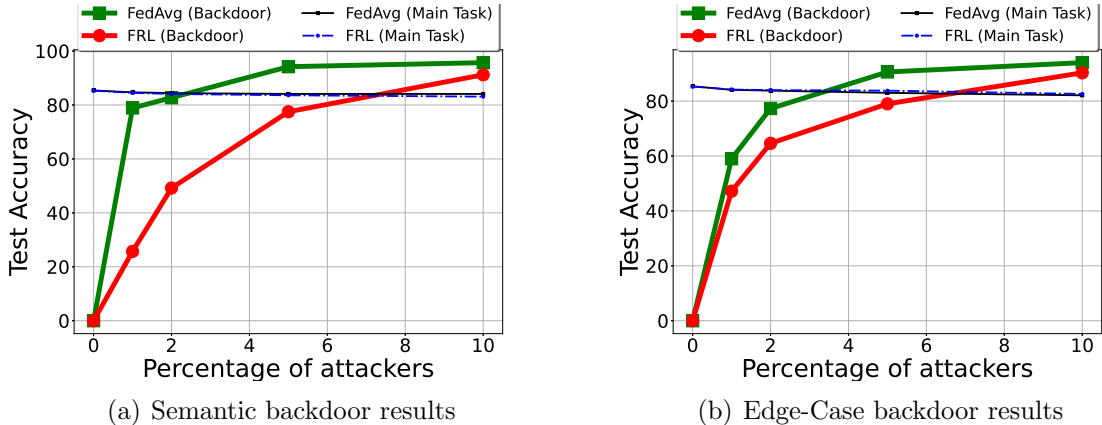


Figure 5.5: FL backdoor poisoning attacks on CIFAR10 distributed over 1000 clients with Dirichlet ( $\beta = 1.0$ ) for presence of adversary in 1000 FL rounds.

**Evaluation results:**

**Semantic backdoor attacks:** Figure 5.5 (a) shows the performance of FedAvg and FRL on the main task and the backdoor task when different percentages of malicious clients want to put a semantic backdoor in the global model. This figure shows that FRL is more robust against semantic backdoor attacks for different percentages of

malicious clients. For example, with 2% of malicious clients, training FedAvg results in 84.4% final test accuracy with 82.7% average backdoor accuracy, while training FRL results in 84.1% and 49.2% accuracy on the main task and backdoor task, respectively. The existence of a more significant number of malicious clients (e.g., 10%) results in higher backdoor accuracy for both FedAvg and FRL as the malicious clients have more influence on the global model to introduce their backdoor; with existence of 10% of malicious clients, training FedAvg and FRL achieves 95.7% and 91.2% average backdoor accuracy respectively.

**Artificial backdoor attacks:** These attacks are ineffective when the adversary cannot use model replacement strategy (i.e., cannot re-scale their parameters). In FRL, malicious clients cannot scale their updates, as they submit a local ranking (from a discrete space of updates). To be fair, we also did not use re-scaling in our experiments for FedAvg. We did not report the results for this attack, as the backdoor accuracy would be 0% for both FRL and FedAvg with no parameter re-scaling. It means that the global model always predict the right label (not the adversary target label "bird") for the test backdoor images.

**Edge-Case backdoor attacks:** Figure 5.5 (b) shows that FRL is more robust against Edge-case backdoor attacks for different percentages of malicious clients. For example, with 2% of malicious clients, training FedAvg results in 83.7% final test accuracy with 77.3% average backdoor accuracy, while training FRL results in 84.0% and 64.6% accuracy on the main task and backdoor task, respectively. Similar to semantic backdoors, a larger number of malicious clients (e.g., 10%) results in higher backdoor accuracy for both FedAvg and FRL; with 10% of malicious clients, training FedAvg and FRL achieves 94.0% and 90.3% average backdoor accuracy respectively.

## CHAPTER 6

### A CRITICAL EVALUATION POISONING ATTACKS ON PRODUCTION FEDERATED LEARNING

In Chapter 4, we demonstrated that the our state-of-the-art model poisoning attacks can significantly reduce performances of the global models trained using FL based on state-of-the-art robust aggregation rules (AGRs). In other words, we showed that the theoretical robustness guarantees of the major robust AGRs overestimate their robustness against poisoning attacks.

**The gap between the literature and practice:** In order to test the theoretical robustness of AGRs, in Chapter 4 we assumed a strong poisoning adversary who introduces and controls up to 25% malicious clients, because most of the existing robust AGRs claim that they can defend against up to 25% malicious clients. Majority of the existing literature on poisoning attacks and defenses for FL makes similarly *unrealistic assumptions* that do not hold in real-world FL deployments, e.g., assumptions about the percentages of malicious clients, total number of FL clients, and the types of FL systems [82]. Typically, the assumptions are about the convexity of optimization problem, percentages of malicious clients, total number of FL clients, and type of the FL setting. For instance, state-of-the-art attacks [64, 22, 132] (defenses [30, 168, 165, 45]) assume adversaries who can compromise up to 25% (50%) of FL clients. For an app like Gboard with  $\sim 1B$  installations [82], 25% malicious clients would mean an attacker controls *250 million Android devices!* We argue that, although interesting from theoretical perspectives, the assumptions in recent FL robustness works do not represent common real-world adversarial scenarios that account for the difficulty and cost of at-scale compromises.

Hence, in this chapter, we perform a critical analysis of the literature on FL robustness against (untargeted) poisoning under practical considerations. Our ultimate goal is to understand the significance of poisoning attacks and the need for sophisticated robust FL algorithms in production FL.

More specifically, following the systematization of threat models of poisoning attacks on FL introduced in Chapter 3, we discuss the practicality of all possible threat models obtained by combining the dimensions corresponding to the objective of untargeted poisoning. Note that, as clarified in Section 3.1.1.1.1, *this thesis focuses on untargeted poisoning on FL*. As we will discuss, out of all possible combinations, only two threat models, i.e., *nobox offline data poisoning* and *whitebox online model poisoning*, are of practical value to production FL. We believe that prior works [22, 30, 64, 132] have neglected the crucial constraints of production FL systems on the parameters relevant to FL robustness. To the best of our knowledge, *for the first time, we consider the production FL environments* [82, 32] and provide practical ranges for various parameters of poisoning threat models. As a result, *our evaluations lead to conclusions that contradict the common beliefs in the literature*, e.g., we show that production FL even with the non-robust Average AGR is significantly more robust than previously thought.

Next, we introduce improved poisoning attacks under the two aforementioned threat models. We build on the classic label flipping data poisoning attack [161, 162, 118] designed for centralized ML and present *the first* attacks that systematically consider the data poisoning threat model for FL. We also propose novel model poisoning attacks based on the idea of *stochastic gradient ascent* and show that they outperform the state-of-the-art.

Finally, we extensively evaluate all existing poisoning attacks as well as our own improved attacks across three benchmark datasets, for various FL parameters, and

Table 6.1: Practical ranges of FL parameters based on the literature and discussions on FL production systems [6, 82, 32] and the ranges used in *untargeted* FL poisoning and robust AGRs literature [64, 22, 132, 30, 108]. MPA means model poisoning attack and DPA means data poisoning attack. Red (green) cells denote impractical (practical) ranges.

Parameters/Settings	What we argue to be practical	Used in previous <i>untargeted</i> works
FL type + Attack type	Cross-silo + DPAs Cross-device + {MPAs, DPAs}	Cross-silo + MPAs
Total number of FL clients, $N$	Order of $[10^3, 10^{10}]$ for cross-device $[2, 100]$ for cross-silo	$[50, 100]$
Number of clients chosen per round, $n$	Small fraction of $N$ for cross-device All for cross-silo	All
% of compromised clients, $M$	$M \leq 0.1\%$ for DPAs $M \leq 0.01\%$ for MPAs	$[20, 50]\%$
Average size of benign clients' data, $ D _{\text{avg}}$	$[50, 1000]$ for cross-device Not applicable to cross-silo	Not studied for cross-device $[50, 1000]$ for cross-silo
Maximum size of local poisoning data	Up to $100 \times  D _{\text{avg}}$ for DPAs Not applicable to MPAs	$\sim  D _{\text{avg}}$

for different types of FL deployments. We make several significant deductions about the state of FL poisoning literature for production FL.

## 6.1 Practical Considerations for Poisoning Threat Models

In this section, we consider practical FL deployments and discuss their features relevant to understanding the robustness of such practical FL deployments.

### 6.1.1 Salient Features of Production Federated Learning

Production FL can be either **cross-device** or **cross-silo** [82]. In *cross-device FL*, the number of clients ( $N$ ) is large (from few thousands to billions) and only a small fraction of them is chosen in each FL training round, i.e.,  $n \ll N$ . In cross-device FL, clients' devices are highly resource constrained, and therefore, they can process only a limited amounts of data in an FL round. Also, as the devices have highly unreliable network connections, it is expected that a small fraction of the selected devices may drop out in any given FL round. Note that, this equally impacts both benign and malicious clients and does not affect the robustness; this is similar to how

Table 6.2: The eight possible threat models for *untargeted poisoning attacks* on FL. T3-T8 are valid, but only T4 and T5 represent practical FL deployments (Section 6.1.4).

	Capability $\in \{\text{MP}, \text{DP}\}$	Knowledge $\in \{\text{Nb}, \text{Wb}\}$	Attack mode $\in \{\text{Off}, \text{On}\}$
T1	Model poison	Nobox	Offline
T2	Model poison	Nobox	Online
T3	Model poison	Whitebox	Offline
T4	Model poison	Whitebox	Online
T5	Data poison	Nobox	Offline
T6	Data poison	Nobox	Online
T7	Data poison	Whitebox	Offline
T8	Data poison	Whitebox	Online

the choice of  $n$  has no impact on the robustness (Section 6.3.4.3). In *cross-silo FL*,  $N$  is moderate (up to 100) and all clients are selected in each round, i.e.,  $n = N$ . Clients are large corporations, e.g., banks, and have devices with ample resources. Hence, they can process very large amounts of data and client drop-outs do not happen.

In both FL types, the on-device model used for inference and the on-device model being trained are different. Hence, an adversary cannot gain any insight into the training-model by querying the inference-model, i.e., nobox access (Table 3.1), and must break into the device, i.e., get whitebox access (Table 3.1).

Finally, we assume that production systems are adequately protected against standard attack vectors and vulnerabilities such as Sybil attacks. For instance, if the adversary manages to operate millions of fake accounts [63], we argue that the service provider should prioritize improving their security attestation protocols instead of deploying FL. Section 6.1.3 also explains that the cost of operating a large scale, persistent botnet in modern operating systems, e.g., Android, is non-trivial. Please refer to [82] for more details on production FL.

### 6.1.2 Understanding the practicality of threat models

For our goal of untargeted poisoning with the partial knowledge of the benign data, we can combine the rest of the dimensions in Table 3.1 and obtain eight possible

threat models (Table 6.2). We argue that only T4 (nobox offline data poison) and T5 (whitebox online model poison) are of practical value, and below, justify why other models are less relevant in practice: **(1)** With model poisoning capability, the adversary has whitebox access by default, hence, T1 and T2 in Table 6.2 are not valid. **(2)** In cross-device FL, only a few selected clients get the most recent global model in each round. Hence, to gain whitebox access to the model, the adversary needs to control (i.e., break into) a large number of devices (so that in most FL rounds, the FL server picks at least one of them), which is impractical in practice as we explain in Section 6.1.3. With whitebox access, the adversary can mount the stronger online model poisoning attacks (MPAs) instead of data poisoning attacks (DPAs). Therefore, T3, T7, and T8 are not reasonable threat models, as they combine whitebox access with either offline attacks or DPAs. **(3)** Under T6 (nobox online data poison), the adversary mounts an online attack, i.e., they *adaptively* poison the local data of malicious clients. But, as the adversary has no knowledge of the (current) global model due to nobox access, they cannot generate new poisoning data adaptively. Hence, the combination of nobox and online is not practical.

### 6.1.3 Practical Ranges of FL Parameters

We argue that the literature on untargeted poisoning [22, 64, 132, 30, 108] rarely evaluates their proposed attacks/defenses for the production FL settings, primarily due to their motivation to perform worse-case analyses. But, we show that such analyses lead to conclusions that do not apply to production FL.

Table 6.1 demonstrates the stark differences between the parameter ranges used in the untargeted poisoning literature and their practical ranges, which we have obtained from recent surveys [82, 32] and discussion among FL experts [6]. This is due to the more challenging nature of untargeted poisoning in FL. We attribute this to the



difficulty of establishing successful untargeted attacks for practical settings, as we will also show in our evaluations.

Contrary to what production FL settings encounter, previous works commonly evaluate robustness using very high percentages of malicious clients and/or using model poisoning attacks on cross-silo FL (Table 6.1). However, we use small percentages of malicious clients  $M \leq 1$ , for cross-device FL, use large numbers of clients  $N \in [1,000, 34,000]$  and use  $n \in [25, 50] \ll N$  in each round; we use  $N=n=50$ .

In particular, consider the percentages of malicious clients; state-of-the-art attacks [64, 22, 132] (defenses [30, 168, 165, 45]) assume adversaries who can compromise up to 25% (50%) of FL clients. The cost of creating and operating a compromised client botnet at scale (which includes breaking into devices) is non-trivial. To create the botnet, the adversary would need to either buy many physical devices ( $\sim \$25$  each) and root them (for state-of-the-art model poisoning attacks [64, 22, 132]), pay for access to large but undetected botnets with remote administrative access, or develop an entirely new botnet via compromising a popular app/sdk to exploit unpatched security holes and gain persistence. To operate the botnet, the adversary must avoid detection by antimalware services [8] as well as dynamic anti-abuse services (such as Android’s SafetyNet [9]). With a botnet in place, the adversary may further need to pay for a skilled engineering team to keep malicious FL code in sync with the target FL-enabled app and to reverse-engineer frequently-shifting ML workloads. Such an engineering team could instead change apps’ behaviors to mimic the effect of a compromised FL-trained model, they might use their privileged access to steal login credentials for account hijacking, or they might participate in ad/click fraud or bank fraud or ransomware for financial gain. More plausible scenarios for an adversary reaching double-digit client percentages—such as an app insider—likely enable attacker-controlled FL servers, thereby removing them from the literature’s standard threat model.

For data poisoning attacks, we assume that malicious clients can have a limited amount of poisoned data  $D_p$ . Because, in cross-device FL, the devices with low processing powers (e.g., smart phones and watches) can process limited  $D_p$  in the short duration of FL rounds. However, in cross-silo FL, silos can inspect  $D_p$  and remove  $D_p$  with sizes much larger than the average size of clients’ data  $|D|_{\text{avg}}$ . Hence, we argue that  $|D_p|$  should be up to  $100 \times |D|_{\text{avg}}$ . We discuss rest of the parameters from Table 6.1 in the corresponding sections.

#### 6.1.4 Threat Models in Practice

Here we discuss the two threat models of practical interest.

##### 6.1.4.1 Nobox Offline Data Poisoning (T4)

In this setting, the adversary does not know the architecture, parameters, or outputs of the global model. The adversary knows the server’s AGR, but may or may not know the global model architecture; we evaluate both cases. We assume that the adversary knows the benign data of the malicious clients and mounts offline data poisoning attacks (DPAs).

This adversary does not require any access to the internals (e.g., FL binaries, memory) of compromised devices, and therefore, can compromise large percentages of production FL clients, e.g., on order of up to 0.1% [82, 6]. However, the poisoning impact of the corresponding poisoned updates is very limited. This is partly because arbitrarily poisoned updates (e.g., of model poisoning attacks (MPAs) [64, 22, 132]) need not map to the valid data domain. For instance, consider the standard *max* function:  $f(x, y) = \max(x, y)$ . Gradient of this function with respect to either  $x$  or  $y$  is always 0 or 1 [53]. Hence, a DPA cannot have a poisoned update with an arbitrary value for gradients of the parameters. But an MPA can, because it can directly assign any arbitrary value to the parameters’ gradients.

### 6.1.4.2 Whitebox Online Model Poisoning (T5)

The adversary knows the parameters and predictions of the global model whenever the server selects at least one compromised client. We assume that the adversary knows the server’s aggregation rule and the benign data on the compromised devices. The adversary mounts online MPAs.

Unlike data poisoning adversary, this adversary breaks into the compromised devices, which is extremely costly as discussed in Section 6.1.3. Hence, in practice, a model poisoning adversary can compromise very small percentages of FL clients, e.g., on order of up to 0.01% [82, 6]. However, due to their ability to directly manipulate the model updates, in theory, a model poisoning adversary can craft highly poisonous updates. We can justify this claim from the example of a zero-value parameter discussed in Section 6.1.4.1.

## 6.2 Exploring The Space of FL Poisoning Attacks

**Existing FL Poisoning Attacks:** Section 2.2.1 details the existing data and model poisoning attacks. In our evaluations, we also consider our model poisoning attacks from Chapter 4; we call them *dynamic optimization (DYN-OPT) attacks*.

### 6.2.1 Improved FL Poisoning Attacks

We use the same general optimization problem as in Section 4.2.1 of Chapter 4 to design improved data and model poisoning attacks. The optimization is reproduced below for reference:

$$\begin{aligned} \operatorname{argmax}_{\nabla' \in R^d} \quad & \|\nabla^b - \nabla^p\| & (6.1) \\ \dots \nabla^b = & f_{\text{avg}}(\nabla_{i \in \{[n']\}}), \quad \nabla^p = f_{\text{agr}}(\nabla'_{\{i \in [m]\}}, \nabla_{\{i \in [n']\}}) \end{aligned}$$

where,  $m$  is the number of malicious clients selected in the given round,  $f_{\text{agr}}$  is the target AGR,  $f_{\text{avg}}$  is the Average AGR,  $\nabla_{\{i \in [n']\}}$  are the benign updates available to

the adversary (e.g., updates computed using the benign data of malicious clients),  $\nabla^b$  is a reference benign aggregate, and  $\nabla'_{\{i \in [m]\}}$  are  $m$  replicas of the poisoned update,  $\nabla'$ , of our attack.  $\nabla^p$  is the final poisoned aggregate.

Although the optimization problem is the same as before, *two key differences are:* **(1)** We are the first to use (6.1) to construct systematic data poisoning attacks on FL. **(2)** Our model poisoning attacks not only tailor the optimization in (6.1) to the given AGR (as in Chapter 4), but also to the given dataset and global model, by using stochastic gradient ascent algorithm (Section 6.2.1.2); this boosts the efficacy of our attack.

### 6.2.1.1 Improved Data Poisoning Attacks (DPAs)

We formulate a general DPA optimization problem using (6.1) as follows:

$$\begin{aligned} \operatorname{argmax}_{D_p \subset \mathcal{D}} \quad & \|\nabla^b - \nabla^p\| & (6.2) \\ \dots \nabla^b \text{ and } \nabla^p \text{ as in (6.1) and } \nabla' = & A(D_p, \theta^g) - \theta^g \end{aligned}$$

where  $\mathcal{D}$  is the entire input space and  $D_p$  is the poisoning data used to compute the poisoned update  $\nabla'$  using a training algorithm  $A$ , e.g., mini-batch SGD, and global model  $\theta^g$ . The rest of the notations are the same as in (6.1). To solve (6.2), we find  $D_p$  such that when  $\theta^g$  is fine-tuned using  $D_p$ , the resulting model  $\theta'$  will have high cross-entropy loss on some benign data  $D_b$  (e.g., that of malicious clients), i.e., high  $L(D_b; \theta')$ , and the corresponding update  $\nabla' = \theta' - \theta^g$  will circumvent the target AGR. Our intuition is that, when the global model is updated using such  $\nabla'$ , it will have high loss on benign data [28, 78, 112].

Sun et al. [147] propose DPAs on federated multi-task learning where each client learns a different task. Hence, their attacks are orthogonal to our work. On the other hand, as [64] demonstrates, backgradient optimization based DPAs [112] are

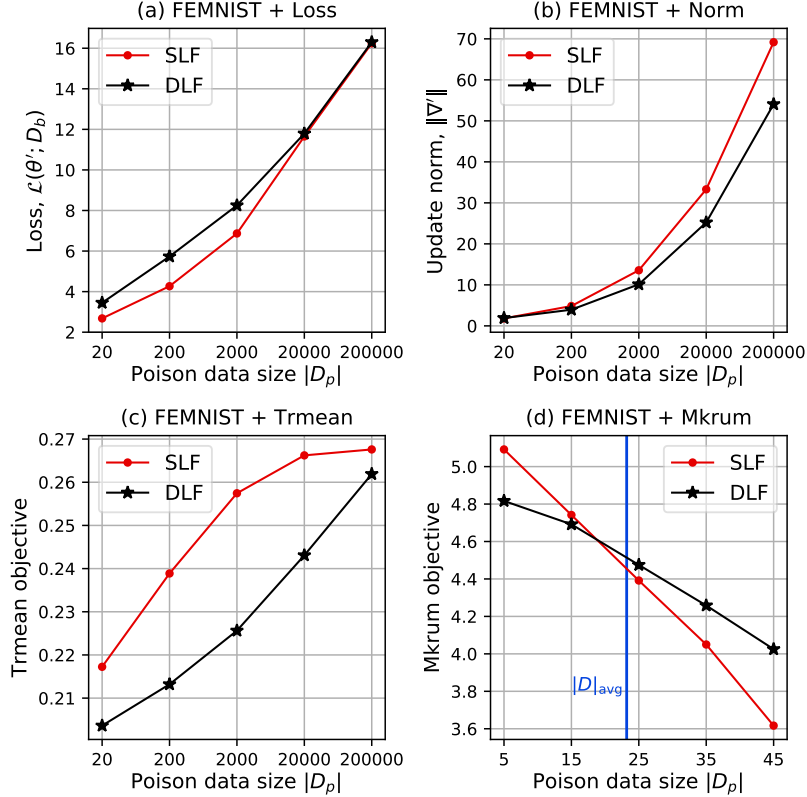


Figure 6.1: Effect of varying the sizes of poisoned data,  $D_p$ , on the objectives of DPAs (Section 6.2.1.1) on various AGRs. We compute  $D_p$  by flipping the labels of benign data.

computationally very expensive ( $\sim 10$  days to compute poison for a subset of MNIST task) yet ineffective.

Instead, because the central server has no visibility into the clients’ data or their sizes, we propose to use an appropriate amount of label flipped data as  $D_p$  for each of the malicious clients. Our intuition behind this approach is the same as before: the larger the amount of label flipped data used to compute  $\theta'$ , the larger the  $L(D_p; \theta')$  and  $\|\nabla'\|$ , and therefore, the higher the deviation in (6.2). We validate this intuition using FEMNIST dataset in Figure 6.1 for various AGRs. For instance, Figures 6.1 (a) and (b) show that increasing  $|D_p|$  monotonically increases update’s loss and norm, respectively, and hence, can effectively poison the Average AGR [30, 108].

In our work, we propose two label flipping (LF) strategies: **static LF (SLF)** and **dynamic LF (DLF)**. In SLF, for a sample  $(\mathbf{x}, y)$ , the adversary flips labels in a

static fashion as in Section 2.2.1. On the other hand, in DLF, the adversary computes a surrogate model  $\hat{\theta}$ , an estimate of  $\theta^g$ , e.g., using the available benign data, and flips  $y$  to the least probable label with respect to  $\hat{\theta}$ , i.e., to  $\text{argmin } \hat{\theta}(\mathbf{x})$ . We observe that the impacts of the two LF strategies are dataset dependent. Therefore, for each dataset, we experiment with both of the strategies and, when appropriate, present the best results. We now specify our DPA for Average, Norm-bounding, Multi-krum, and Trimmed-mean AGRs; these are described in Section 2.2.3.

**Average:** To satisfy the attack objective in (6.2) for Average AGR, we produce updates with large loss and norm [30, 108] using very large amounts of label flipped data (Figures 6.1-(a,b)).

To obtain large  $|D_p|$ , we combine the benign data of all malicious clients and flip their labels using either SLF or DLF strategy (simply SLF/DLF). To increase  $|D_p|$  further, we add Gaussian noise to existing feature vectors of  $|D_p|$  to obtain new feature vectors and flip their labels using SLF or DLF.

**Norm-bounding:** To attack Norm-bounding AGR, we use large  $|D_p|$  to generate poisoned updates that incur high losses on benign data (as we show in Figure 6.1-(b)). As our evaluations will show, even if their norms are bounded, such poisoned updates remain far from benign updates and have high poisoning impacts. This leads to effective attacks, but only at high percentages of malicious clients (e.g.,  $M=10\%$ ).

**Multi-krum:** Following [132], our attack aims to maximize the number of poisoned updates in the selection set ( $S$ ) of Multi-krum AGR (Section 2.2.3). As the size of  $S$  is fixed, maximizing the number of poisoned updates in  $S$  implicitly means minimizing the number of benign updates. This objective is formalized as:

$$\operatorname{argmax}_{D_p \subset D'_p} m' = |\{\nabla \in \nabla'_{\{i \in [m]\}} | \nabla \in S\}| \quad (6.3)$$

where  $D'_p$  is all the available labels flipped data and  $m'$  is the final number of poisoned updates in  $S$  of Multi-krum.

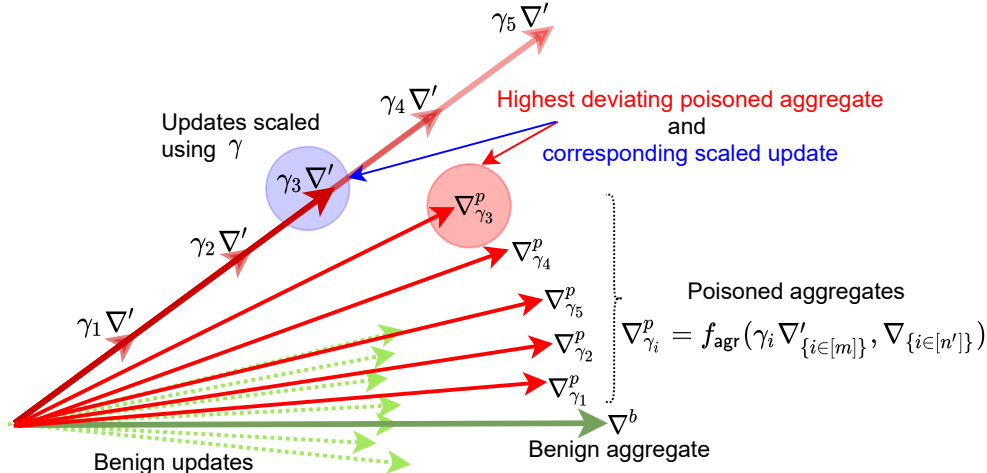


Figure 6.2: Schematic of our PGA attack: PGA first computes a *poisoned update*  $\nabla'$  using stochastic gradient ascent (SGA). Then,  $f_{\text{project}}$  finds the scaling factor  $\gamma$  that maximizes the deviation between benign aggregate  $\nabla^b$  and poisoned aggregate  $\nabla_{\gamma}^p$ . Robust aggregations easily discard the scaled poisoned updates,  $\gamma \nabla'$ , with very high  $\gamma$  (e.g.,  $\gamma_{\{4,5\}}$ ), while those with very small  $\gamma$  (e.g.,  $\gamma_{\{1,2\}}$ ) have no impact.

We solve (6.3) based on an observation: In Figure 6.1-(d) we vary  $|D_p|$  and plot the fraction of corresponding poisoned updates that Multi-krum selects. Let  $|D|_{\text{avg}}$  be the average dataset size of benign clients, e.g.,  $|D|_{\text{avg}}$  is 23.7 for FEMNIST. Note from Figure 6.1-(d) that, even for  $|D_p|$  slightly higher than  $|D|_{\text{avg}}$ , Multi-krum easily discards most of the poisoned updates. Only when  $|D_p|$  is small ( $\sim 10$ ), Multi-krum selects most of the poisoned updates. Hence, we sample  $D_p \subset D'_p$ , where we vary  $|D_p| \in [0.5 \cdot |D|_{\text{avg}}, 3 \cdot |D|_{\text{avg}}]$ , and check the poisoning impact of  $D_p$  on Multi-krum; to reduce variance, we repeat this 10 times for each  $|D_p|$ . We report the results for  $D_p$  with the maximum poisoning impact.

**Trimmed-mean:** For Trimmed-mean AGR, we use the objective in (6.2), but it is cumbersome to solve it directly. Hence, similar to our attacks on Average and Norm-bounding AGRs, we use large  $|D_p|$  for poisoned data on each of the malicious clients. Our approach is based on the observation in Figure 6.1-(c): The higher the  $|D_p|$  (obtained using DLF/SLF strategies), the higher the Trimmed-mean objective value, i.e.,  $\|\nabla^p - \nabla^b\|$ .

### 6.2.1.2 Improved Model Poisoning Attacks (MPAs)

We use (6.1) as the general optimization problem for our MPAs. To solve this optimization, we craft a poisoned model  $\theta'$  with high  $L(D_b; \theta')$  while ensuring that the corresponding poisoned update,  $\nabla'$ , circumvents the target AGR.

Model poisoning adversary can directly manipulate the malicious clients' updates (Section 6.1.4.2). Hence, first, our attack uses the stochastic gradient ascent (SGA) algorithm (instead of SGD) and fine-tunes  $\theta^g$  to increase (instead of decreasing) the loss on some benign data,  $D_b$ , to obtain a malicious  $\theta'$ . But, in order to ensure that the corresponding poisoned update, i.e.,  $\nabla' = \theta' - \theta^g$ , circumvents the target AGR, we *project* the update on a ball of radius  $\tau$  around origin, i.e., scale the update to have a norm  $\|\nabla'\| \leq \tau$ , where  $\tau$  is the average of norms of the available benign updates. Hence, we call our attack **projected gradient ascent (PGA)**. To perform stochastic gradient ascent, we increase the loss on batch  $b$  of data by *using the opposite of a benign gradient direction*, i.e.,  $-\nabla_{\theta}\mathcal{L}(\theta; b)$ .

Algorithm 6 gives the overview of our MPA. The adversary first computes  $\tau$  (line 2), an average of the norms of some benign updates available to her ( $\nabla_{\{i \in [n']\}}$ ). Then, the adversary fine-tunes  $\theta^g$  using  $D_p$  and SGA to compute a poisoned update  $\nabla'$ ; our attack computes  $\nabla'$  for any AGR in the same manner. Finally, the adversary uses  $f_{\text{project}}$  function to appropriately project  $\nabla'$  in order to circumvent the robustness criteria of the target AGR,  $f_{\text{agr}}$ .

Algorithm 7 describes  $f_{\text{project}}$ : It computes  $\nabla^b = f_{\text{avg}}(\nabla_{\{i \in [n']\}})$ . Then, it finds a scaling factor  $\gamma$  for  $\nabla'$  that maximizes the distance between the benign aggregate  $\nabla^b$  and the poisoned aggregate  $\nabla^p = f_{\text{agr}}(\gamma \nabla'_{\{i \in [m]\}}, \nabla_{\{i \in [n']\}})$ . Note that, there can be many ways to optimize  $\gamma$  [132], but we empirically observe that simply searching for  $\gamma$  in a pre-specified range (e.g.,  $[1, \Gamma]$  with  $\Gamma \in \mathbb{R}^+$ ) yields strong attacks (line 6). Figure 6.2 depicts the idea of  $f_{\text{project}}$  algorithm.



---

**Algorithm 6** Our PGA model poisoning attack algorithm

---

- 1: **Input:**  $\nabla_{\{i \in [n']\}}, \theta^g, f_{\text{agr}}, D_p$
  - 2:  $\tau = \frac{1}{n'} \sum_{i \in [n']} \|\nabla_i\|$  {Compute norm threshold}  
    { $\tau$  is given for norm-bounding AGR}
  - 3:  $\theta' \leftarrow A_{\text{SGA}}(\theta^g, D_p)$  {Update using stochastic gradient ascent}
  - 4:  $\nabla' = \theta' - \theta^g$  {Compute poisoned update}
  - 5:  $\nabla' = f_{\text{project}}(f_{\text{agr}}, \nabla', \tau, \nabla_{\{i \in [n']\}})$  {Scale  $\nabla'$  appropriately}
  - 6: **Output**  $\nabla'$
- 

---

**Algorithm 7** The projection function ( $f_{\text{project}}$ ) of our PGA from Section 6.2.1.2.

---

- 1: **Input:**  $f_{\text{agr}}, \nabla', \tau, \nabla_{\{i \in [n']\}}$
  - 2:  $d^* = 0$  {Initialize maximum deviation}
  - 3:  $\gamma^* = 1$  {Optimal scaling factor that maximizes deviation in (6.1)}
  - 4:  $\nabla' = \frac{\nabla' \times \tau}{\|\nabla'\|}$  {Scale  $\nabla'$  to have norm  $\tau$ }
  - 5:  $\nabla^b = f_{\text{avg}}(\nabla_{\{i \in [n']\}})$  {Compute reference benign update}
  - 6: **for**  $\gamma \in [1, \Gamma]$  **do**
  - 7:      $\nabla'' = \gamma \cdot \nabla'$
  - 8:      $d = \|f_{\text{agr}}(\nabla''_{\{i \in [m]\}}, \nabla_{\{i \in [n']\}}) - \nabla^b\|$
  - 9:      $\gamma^* = \gamma$  **if**  $d > d^*$  {Update optimal  $\gamma$ }
  - 10:     $\gamma = \gamma + \delta$  {Update  $\gamma$ }
  - 11: **end for**
  - 12: **Output**  $\gamma^* \cdot \nabla'$
- 

Due to the modular nature of our attacks, one can attack any given AGR by plugging its algorithm in Algorithm 7. This is unlike Sun et al. [148], who propose a similar *targeted* attack which only works against norm-bounding AGR.

Furthermore, to reduce computation, below we tailor  $f_{\text{project}}$  to some of the state-of-the-art AGRs from Section 2.2.3; note that, the adversary obtains a poisoned update,  $\nabla'$ , using Algorithm 6 before tailoring  $f_{\text{project}}$  to the target AGR.

**Average:** Average does not impose any robustness constraints, therefore, we simplify  $f_{\text{project}}$  by scaling  $\nabla'$  by an arbitrarily large constant, e.g.,  $10^{20}$ . If the server selects a compromised client, such poisoned update suffices to completely poison  $\theta^g$ .

**Norm-bounding:** Following the Kirchoff's law, we assume that the attacker knows the norm-bounding threshold,  $\tau$ , and therefore,  $f_{\text{project}}$  scales  $\nabla'$  by  $\frac{\tau}{\|\nabla'\|}$ , so that the norm of the final  $\nabla'$  will be  $\tau$ .

**Multi-krum:** Similar to our DPA (Section 6.2.1.1), the objective of our MPA on Multi-krum is to maximize the number of poisoned updates in the selection set  $S$ . We aim to find a scaling factor  $\gamma$  for  $\nabla'$  such that maximum number of  $\nabla'' = \gamma\nabla'$  are selected in  $S$ . This is formalized below:

$$\operatorname{argmax}_{\gamma^* \in \mathbb{R}} m = |\{\nabla \in \nabla''_{\{i \in [m]\}} | \nabla \in S\}| \quad (6.4)$$

To solve the optimization in (6.4), our  $f_{\text{project}}$  searches for the maximum  $\gamma$  in a pre-specified range  $[1, \Gamma]$  such that Multi-krum selects all the scaled poisoned updates. Specifically, in Algorithm 7, instead of computing the deviation (line-8), we compute the number of  $\nabla''$  selected in  $S$  and update  $\gamma^*$  if  $S$  has all of  $\nabla''$ s.

**Trimmed-mean:** Here, we directly plug Trimmed-mean algorithm in Algorithm 7 (line-8). Our attack is similar to that of [132], but instead of using one of several perturbation vectors,  $\omega$ 's, we use stochastic gradient ascent to tailor  $\omega$  to the entire FL setting ( e.g.,  $\theta^g$ , data, optimizer, etc.) to improve the attack impact.

### 6.3 Analysis of FL Robustness in Practice

In this section, we evaluate state-of-the-art data (DPAs) and model poisoning attacks (MPAs) against non-robust and robust FL algorithms (Section 2.2.3), under practical threat models from Section 6.1.4. We start by analyzing cross-device FL (Sections 6.3.2 to 6.3.4), as it is barely studied in previous works and is more susceptible to poisoning. Then, we will analyze cross-silo FL in Section 6.3.5.

#### 6.3.1 Experimental setup

Real-world FL datasets [3, 121] are proprietary and cannot be publicly accessed. Hence, we follow the literature on untargeted poisoning in FL [64, 132, 22, 151] and focus on image and categorical datasets. But, we ensure that our setup embodies

the production FL [82], e.g., by using large number of clients with extremely non-iid datasets.

### 6.3.1.1 Datasets and Model Architectures

**FEMNIST** [36, 51] is a character recognition classification task with 3,400 clients, 62 classes (52 for upper and lower case letters and 10 for digits), and 671,585 grayscale images. Each client has data of her own handwritten digits or letters. Considering the huge number of clients in real-world cross-device FL (up to  $10^{10}$ ), we further divide each of the clients' data in  $p \in \{2, 5, 10\}$  non-iid parts using Dirichlet distribution [110] with  $\alpha = 1$ . Increasing the Dirichlet distribution parameter,  $\alpha$ , generates more iid datasets. Unless specified otherwise, we set  $p = 10$ , i.e., the total number of clients is 34,000. We use LeNet [96] architecture.

**CIFAR10** [88] is a 10-class classification task with 60,000 RGB images (50,000 for training and 10,000 for testing), each of size  $32 \times 32$ . Unless specified otherwise, we consider 1,000 total FL clients and divide the 50,000 training data using Dirichlet distribution [110] with  $\alpha = 1$ . We use VGG9 architecture with batch normalization [139].

**Purchase** [4] is a classification task with 100 classes and 197,324 binary feature vectors each of length 600. We use 187,324 of total data for training and divide it among 5,000 clients using Dirichlet distribution with  $\alpha = 1$ . We use validation and test data of sizes 5,000 each. We use a fully connected network with layer sizes  $\{600, 1024, 100\}$ .

### 6.3.1.2 Details of Federated learning and attack parameters

For FEMNIST, we use 500 rounds, batch size,  $\beta = 10$ ,  $E = 5$  local training epochs, and in the  $e^{th}$  round use SGD optimizer with a learning rate  $\eta = 0.1 \times 0.995^e$  for local training; we select  $n = 50$  clients per round and achieve baseline accuracy  $A_\theta=82.4\%$  with  $N=34,000$  clients. For CIFAR10, we use 1,000 rounds,  $\beta = 8$ ,  $E = 2$ ,

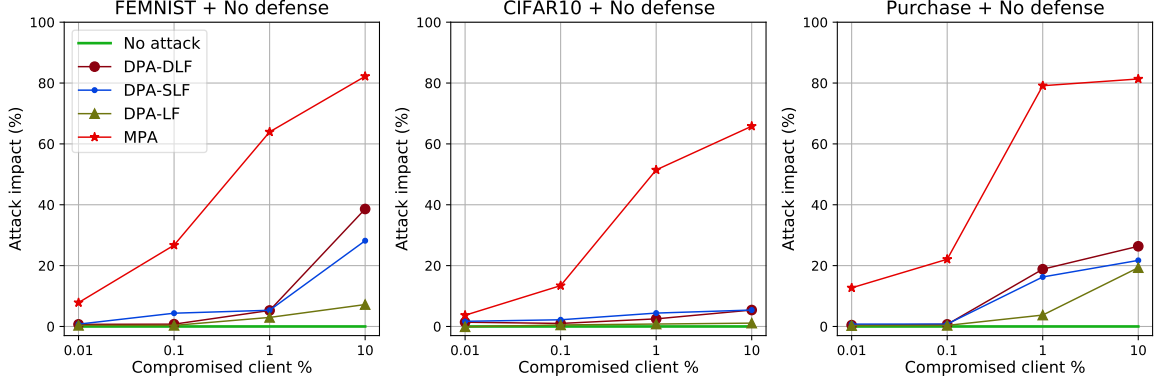


Figure 6.3: Attack impacts ( $I_\theta$ ) of state-of-the-art data (DPA-DLF/SLF) and model (MPA) poisoning attacks on cross-device FL with average AGR.  $I_\theta$ 's are significantly lower for practical percentages of compromised clients ( $\leq 0.1\%$ ) than previously thought.

and in the  $e^{th}$  round use SGD with momentum of 0.9 and  $\eta = 0.01 \times 0.9995^e$ ; we use  $n = 25$  and achieve  $A_\theta=86.6\%$  with  $N=1,000$ . For Purchase, we use 500 rounds,  $\beta = 10$ ,  $E = 5$ , and in the  $e^{th}$  round use SGD with  $\eta = 0.1 \times 0.999^e$ ; we use  $n = 25$  and achieve  $A_\theta=81.2\%$  with  $N=5,000$ .

We generate large poisoned data  $D_p$  required for our DPAs (Section 6.2.1.1) by combining the dataset of compromised clients and adding Gaussian noise to their features. We round the resulting feature for categorical Purchase dataset.

### 6.3.1.3 Attack impact metric

$A_\theta$  denotes the maximum accuracy that the global model achieves over all FL training rounds, without any attack.  $A_\theta^*$  for an attack denotes the maximum accuracy of the model under the given attack. We define *attack impact*,  $I_\theta$ , as *the reduction in the accuracy of the global model due to the attack*, hence for a given attack,  $I_\theta = A_\theta - A_\theta^*$ .

## 6.3.2 Evaluating Non-robust FL (Cross-device)

We study Average AGR due to its practical significance and widespread use. Previous works [30, 64, 132, 22, 108, 168] have argued that even a single compromised

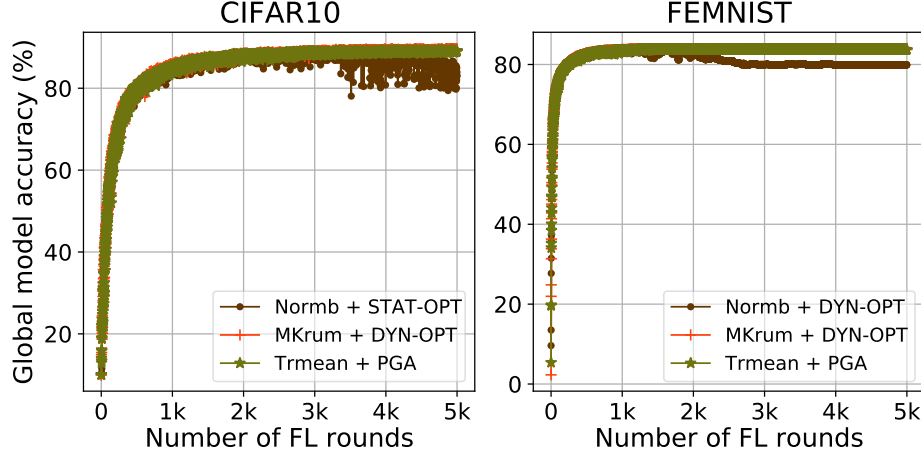


Figure 6.4: Even with a very large number of FL rounds (5,000), the state-of-the-art model poisoning attacks with  $M=0.1\%$  cannot break the robust AGRs (Section 6.3.3).

client can prevent the convergence of FL with Average AGR. However, our results contradict those of previous works: we show that this established belief about Average AGR is *incorrect* for production cross-device FL.

Figure 6.3 shows the attack impacts ( $I_\theta$ ) of various DPAs and MPAs. Note that, for the Average AGR, all MPAs [132, 64, 22], including ours, are the same and craft arbitrarily large updates in a malicious direction. Hence, we show a single line for MPAs in Figure 6.3.

We see that *for cross-device FL, when percentages of compromised clients ( $M$ ) are in practical ranges (Table 6.1),  $I_\theta$ 's of all the attacks are very low, i.e., the final  $\theta^g$  converges with high accuracy.* For FEMNIST,  $I_\theta$  of MPAs at  $M=0.01\%$  is  $\sim 2\%$  and  $I_\theta$  of DPAs at  $0.1\%$  is  $\sim 5\%$ . In other words, compared to the no attack accuracy (82.3%), the attacks reduce the accuracy by just 2% and 5%. Similarly, we observe very low  $I_\theta$ 's for the Purchase and CIFAR10 datasets.

Note that, here we use very large local poisoned data ( $D_p$ ) for our DPAs, as DPAs on Average AGR become stronger with higher  $|D_p|$  (Section 6.2.1.1);  $|D_p|$ 's are 20,000, 50,000, and 20,000 for FEMNIST, CIFAR10, and Purchase, respectively. However,

as we will show in Section 6.3.4.1, *under practical  $|D_p|$ ,  $I_\theta$ 's of DPAs are negligible even with  $M=10\%$ .*

*The inherent robustness of cross-device FL is due to its client sampling procedure.*

In an FL round, the server selects a very small fraction of all FL clients. Hence, in many FL rounds no compromised clients are chosen when  $M$  ( $< 1\%$ ) is in practical ranges.

**(Takeaway 6.3.2)** Contrary to the common belief, production cross-device FL with (the naive) Average AGR converges with high accuracy even in the presence of untargeted poisoning attacks.

### 6.3.3 Evaluating Robust FL (Cross-device)

In this section, contrary to previous works, we study the robustness of robust AGRs for cross-device FL when percentages of compromised clients ( $M$ ) are in practical ranges. Figure 6.8 shows the poisoning impact ( $I_\theta$ ) of DPAs and MPAs for Normbounding (Normb), Multi-krum (Mkrum), and Trimmed-mean (Trmean) AGRs. Below, we discuss three **key takeaways**:

#### 6.3.3.1 Cross-device FL with robust AGRs is highly robust in practice

$I_\theta$  of attacks on robust AGRs are negligible in practice, i.e., when  $M \leq 0.1\%$  for DPAs and  $M \leq 0.01\%$  for MPAs. For instance,  $I_\theta \leq 1\%$  for all of state-of-the-art attacks on all the three datasets, i.e., the attacks reduce the accuracy of  $\theta^g$  by less than 1 percent.

We also run FL with a robust AGR for a very large number (5,000) of rounds to investigate if the strongest of MPAs against the AGR with  $M = 0.1\%$  can break the AGR after long rounds of continuous and slow poisoning. Figure 6.4 shows the results: Mkrum and Trmean remain completely unaffected (in fact accuracy of the global model increases), while accuracy due to Normb reduces by  $<5\%$ .

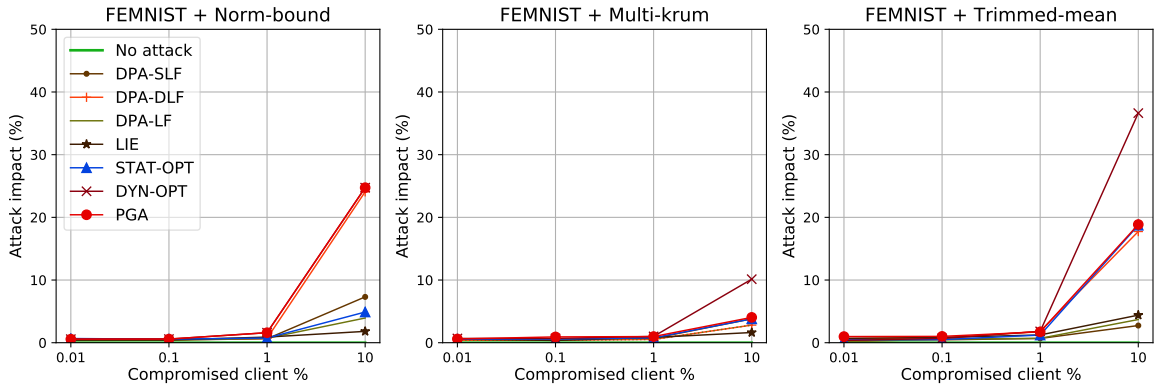


Figure 6.5: FEMNIST with CNN architecture

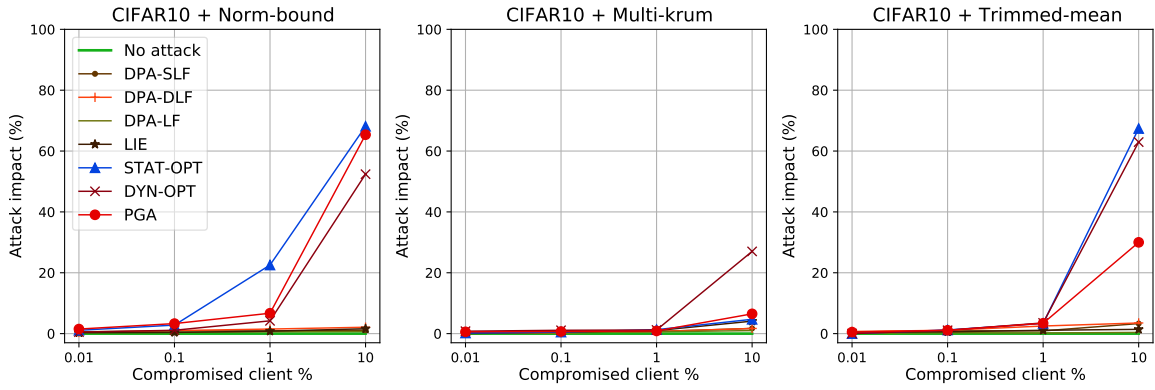


Figure 6.6: CIFAR10 with VGG9 architecture

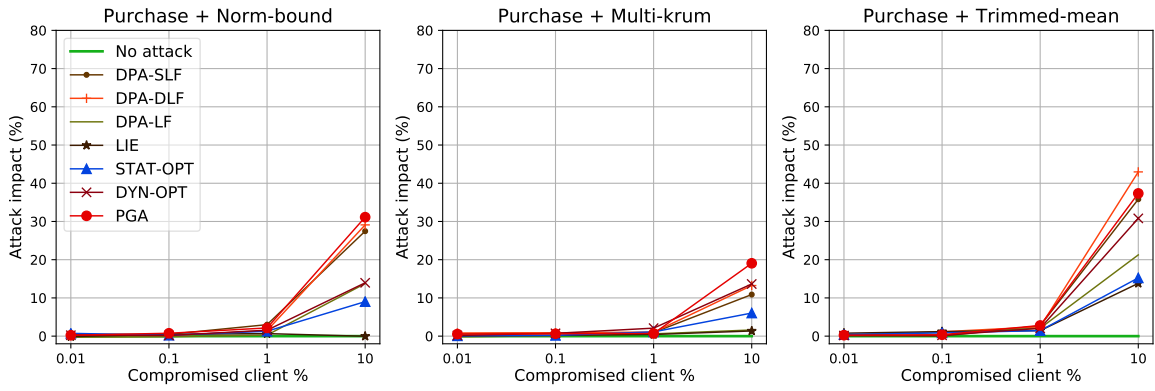


Figure 6.7: Purchase with Fully connected architecture

Figure 6.8: Attack impacts of various poisoning attacks (Section 6.2) on cross-device FL with robust AGRs (Section 2.2.3). *These AGRs are highly robust for practical percentages of compromised clients; for in-depth analysis, please check Section 6.3.3.*

In summary, state-of-the-art poisoning attacks [132, 22, 64] demonstrate that the robust AGRs are significantly less robust than their theoretical guarantees. On the other hand, our findings show that these AGRs are more than sufficient to protect, more practical, production cross-device FL against untargeted poisoning. This is due to the peculiar client sampling of cross-device FL, as discussed in Section 6.3.2.

**(Takeaway 6.3.3.1)** Cross-device FL with robust AGRs is highly robust to state-of-the-art poisoning attacks under production FL environments ( $M < 0.1\%$ ,  $n \ll N$ ).

### 6.3.3.2 Investigating simple and efficient robustness checks is necessary

Most of the state-of-the-art robust AGRs with strong theoretical guarantees [30, 108, 168, 165] have complex robustness checks on their inputs, which incur high computation and storage overheads. For instance, to process  $n$  updates of length  $d$ , the computational complexity of Mkrum is  $\mathcal{O}(dn^2)$  and that of Trmean is  $\mathcal{O}(dn \log n)$ . Therefore, in production FL systems where  $n$  can be up to 5,000 [82, 32], the computation cost prohibits the use of such robust AGRs.

On the other hand, Norm-bounding only checks for the norm of its inputs and has computation complexity of  $\mathcal{O}(d)$ , same as Average. Figure 6.8 shows that a simple and efficient AGR, Norm-bounding, protects cross-device FL against state-of-the-art poisoning attacks similarly to the theoretically robust (and expensive) AGRs, under practical  $M$ . For instance, for all the datasets with  $M \leq 1\%$ ,  $I_\theta < 1\%$  for all of the AGRs (Figure 6.8). Our evaluation highlights that simple robust AGRs, e.g., Norm-bounding, can effectively protect cross-device FL in practice, and calls for further investigation and invention of such low-cost robust AGRs.

**(Takeaway 6.3.3.2)** Even the simple, low-cost Norm-bounding AGR is enough to protect production FL against untargeted poisoning, questioning the need for the more sophisticated (and costlier) AGRs.



### 6.3.3.3 Thorough empirical assessment of robustness is inevitable

Theoretically robust AGRs claim robustness to poisoning attacks at high  $M$ 's, e.g., in theory, Mkrum [30] and Trmean [168] are robust for  $M \leq 25\%$ . But, we observe that, even at the theoretically claimed values of  $M$ , these robust AGRs do not exhibit high robustness; in fact, simple AGRs, e.g., Norm-bounding, are equally robust. Note in Figure 6.8 that, for FEMNIST at  $M=10\%$ ,  $I_\theta$ 's on Trmean are *higher* than on Norm-bounding. For CIFAR10 at  $M=10\%$ ,  $I_\theta$ 's for Norm-bounding and Trmean are almost similar.

Sections 6.3.3.2 and 6.3.3.3 show that, some of the sophisticated, theoretically robust AGRs do not outperform simpler robust AGRs at *any* ranges of  $M$ . More importantly they demonstrate the shortcomings of the methodology used to assess the robustness of AGRs in previous works [30, 108, 168, 165] (because these works use very preliminary attacks) and highlight that a thorough empirical assessment is necessary to understand the robustness of AGRs in production FL systems.

**(Takeaway 6.3.3.3)** Understanding the robustness of AGRs in production FL requires a thorough empirical assessment of AGRs, on top of their theoretical robustness analysis.

## 6.3.4 Effect of FL Parameters on Poisoning (Cross-device)

### 6.3.4.1 Effect of the Size of Local Poisoning Datasets ( $|D_p|$ ) on DPAs.

The success of our state-of-the-art data poisoning attacks depends on  $|D_p|$  of compromised clients (Section 6.2.1.1). In Sections 6.3.2 and 6.3.3, we use large  $|D_p|$  (e.g., 50,000 for CIFAR10) to find the highest impacts of DPAs. But, as argued in Section 6.1.3, in practice  $|D_p| \leq 100 \times |D|_{\text{avg}}$ ;  $|D|_{\text{avg}}$  is the average size of local datasets of benign clients and it is around 20 (50) for FEMNIST (CIFAR10). In Figure 6.9, we report  $I_\theta$  of the best of DPA-SLF or DPA-DLF for  $|D_p| \in \{1, 10, 10^2, 10^3, 10^4\} \cdot |D|_{\text{avg}}$ ; we use impractically high  $|D_p|$ 's of up to  $10^4 \cdot |D|_{\text{avg}}$  only for experimental analyses.

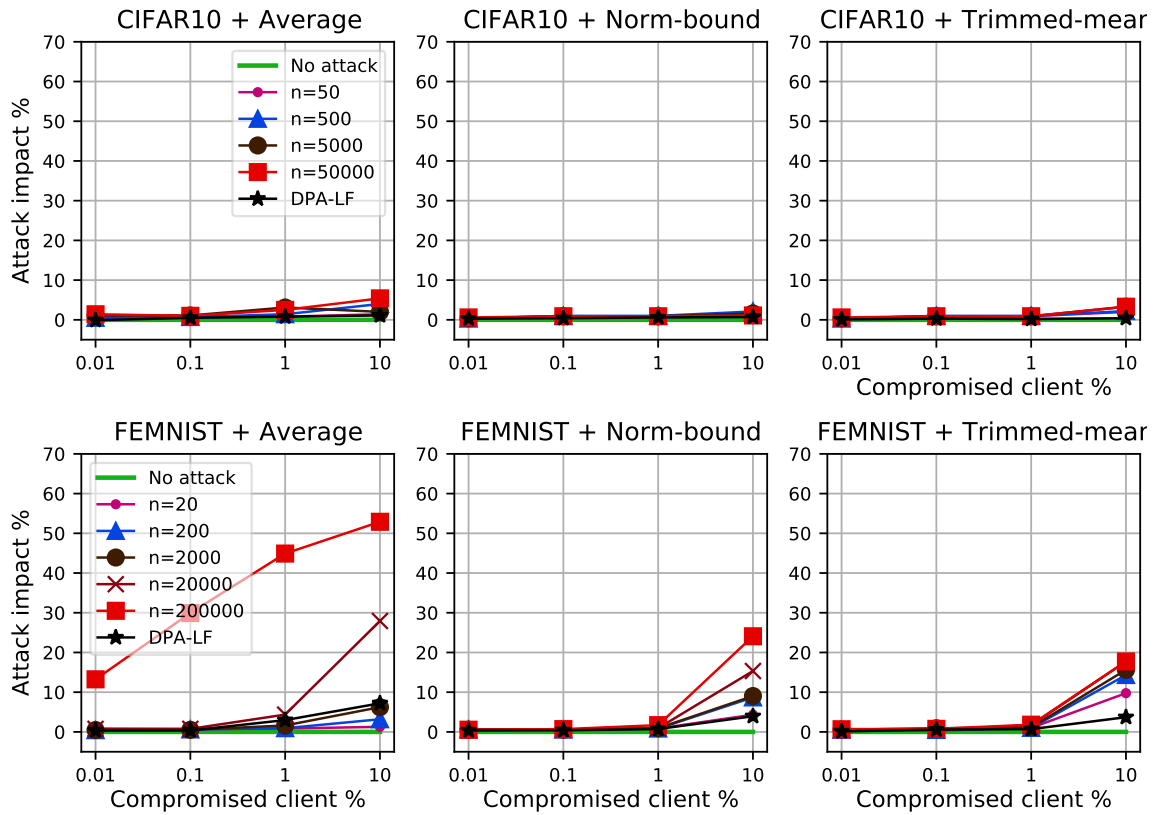


Figure 6.9: Effect of varying sizes of local poisoned dataset  $D_p$  on impacts  $I_\theta$  of the best of DPAs. When  $|D_p|$  and  $M$  are in practical ranges,  $I_\theta$ 's are negligible for robust AGRs and are dataset dependent for non-robust Average AGR.

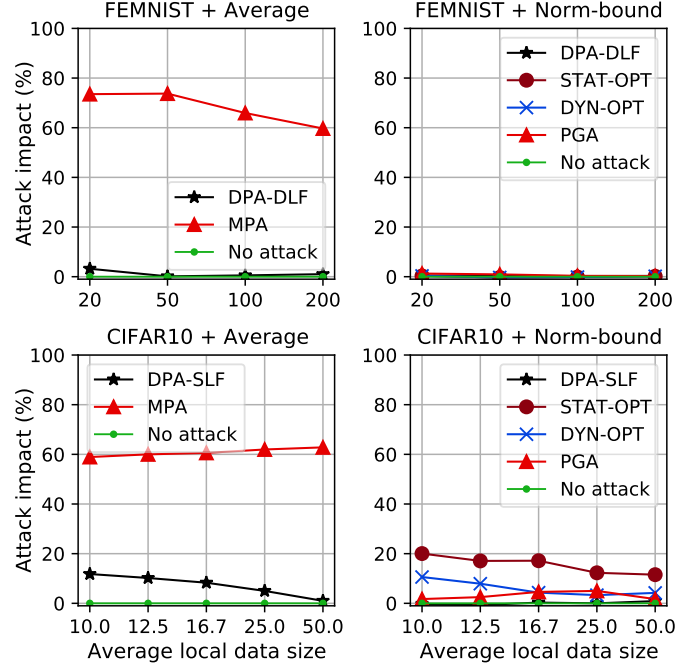


Figure 6.10: With 1% compromised clients, *increasing  $|D|_{\text{avg}}$  has no clear pattern of effects of on attack impacts, but it increases the global model accuracy*

Figure 6.9 shows that  $I_\theta$ 's of DPAs slightly increase with  $|D_p|$ . For FEMNIST and CIFAR10 with any AGR, including Average,  $I_\theta$ 's are negligible even for unrealistically high  $|D_p|$  of  $1000 \times |D|_{\text{avg}}$  for  $M \leq 1\%$ . We omit Mkrum here, as  $|D_p|$  of the effective DPAs on Mkrum is always in practical ranges and close to  $|D|_{\text{avg}}$  (Section 6.2.1.1).

To summarize, *for all robust AGRs, DPAs have negligible impacts on FL when  $|D_p|$  and  $M$  are in practical ranges, while for non-robust AGRs, the reductions in  $I_\theta$  are non-trivial and dataset dependent.* This also means that using a reasonable upper bound on the dataset sizes of FL clients can make FL highly robust to DPAs.

**(Takeaway 6.3.4.1)** Enforcing a limit on the size of the local dataset of each client can act as a highly effective (yet simple) defense against untargeted DPAs in production FL.

### 6.3.4.2 Effect of the Average Dataset Size of Benign FL Clients ( $|D|_{\text{avg}}$ )

Figure 6.10 shows  $I_\theta$  when we vary  $|D|_{\text{avg}}$ . To emulate varying  $|D|_{\text{avg}}$ , we vary the total number of FL clients,  $N$ , for given dataset, e.g., for CIFAR10,  $|D|_{\text{avg}}$  is 50 (10) for  $N=1,000$  ( $N=5,000$ ). As discussed in Section 6.1.3, we use  $|D_p|=100 \times |D|_{\text{avg}}$  for DPAs.

We observe *no clear effect of varying  $|D|_{\text{avg}}$  on  $I_\theta$ 's*. For instance, at  $M=1\%$ ,  $I_\theta$ 's of our PGA and DPA-SLF on CIFAR10 + Normb reduce with increase in  $|D|_{\text{avg}}$ , while  $I_\theta$  of any attacks on FEMNIST with robust AGRs do not change with varying  $|D|_{\text{avg}}$ . We explain each of these observations below:

At  $M=1\%$ ,  $I_\theta$ 's of STAT-OPT on CIFAR10 + Normb reduce with increase in  $|D|_{\text{avg}}$ . This is because, increasing  $|D|_{\text{avg}}$  improves the quality of updates of benign clients, but does not improve the attacks. Hence, when the benign impact of benign updates overpowers the poisoning impact of poisoned updates,  $I_\theta$ 's reduce.

On the other hand,  $I_\theta$ 's of any attacks on FEMNIST with robust AGRs do not change with varying  $|D|_{\text{avg}}$ . This is because, FEMNIST is an easy task, and therefore, the presence of compromised clients does not affect the global models.

Interestingly,  $I_\theta$  of MPAs on CIFAR10 with Average AGR *increases* with  $|D|_{\text{avg}}$ . This is because, due to the difficulty of CIFAR10 task, MPAs on CIFAR10 with Average AGR are very effective and when the server selects even a single compromised client, it completely corrupts the global model.

More importantly, we observe that *even with moderately high  $|D|_{\text{avg}}$ , cross-device FL completely mitigates state-of-the-art DPAs and MPAs despite  $M$  being impractically high*, with an exception of MPAs on Average AGR. For instance, for CIFAR10 with  $|D|_{\text{avg}}=50$  and FEMNIST with  $|D|_{\text{avg}}=200$ , all robust AGRs almost completely mitigate all of DPAs and MPAs, while Average AGR mitigates all DPAs. However, as MPAs are very effective against Average, their  $I_\theta$  remains high. As clients in FL continuously generate data locally [34, 104], it is common to have large  $|D|_{\text{avg}}$  in

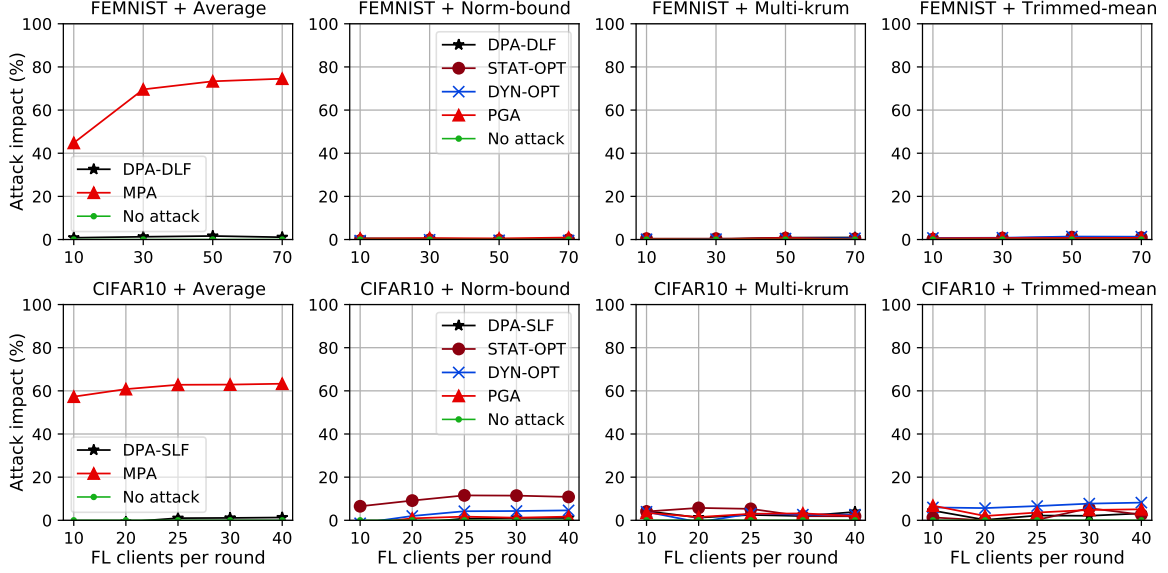


Figure 6.11: As discussed in Section 6.3.4.3, the number of clients,  $n$ , chosen in each FL round has no noticeable effect on the attack impacts, with the exception of model poisoning on Average AGR. We use  $M = 1\%$  of compromised clients.

practice. Interestingly, our evaluation also implies that simply lower bounding the dataset sizes of FL clients improves FL robustness.

**(Takeaway 6.3.4.2)** When local dataset sizes of benign clients are in practical regimes (Table 6.1), cross-device FL with robust AGRs is highly robust to untargeted poisoning.

### 6.3.4.3 Number of Clients Selected Per Round.

Figure 6.11 shows the effect of varying the number of clients ( $n$ ) selected by the server in each round (for  $M=1\%$ ). Similar to [64], we do not observe any noticeable effect of  $n$  on the impact of attacks, since the expected percentage of compromised clients ( $M$ ) does not change with  $n$ . But, we observe the opposite behavior for MPAs on Average AGR. This is because, as soon as the server selects even a single compromised client, MPA prevents any further learning of the global model. An

increase in  $n$  increases the chances of selecting compromised clients, hence amplifying the attack.

**(Takeaway 6.3.4.3)** The number of clients selected in each round of production cross-device FL has no noticeable effect on the impacts of untargeted poisoning attacks, with the exception of MPAs on Average AGR.

### 6.3.4.4 Effect of Unknown Global Model Architecture on DPAs

DPA-DLF attack (Section 6.2.1.1) uses the knowledge of global model’s architecture to train a surrogate model. However, in practice, the nobox offline data poisoning adversary (Section 6.1.4.1) may not know the architecture. Hence, we evaluate impact of DPA-DLF under the unknown architecture setting.

We emulate the unknown architecture setting for FEMNIST dataset. We assume that the adversary uses a substitute convolutional neural network given in Table 6.3 as they do not know the true architecture, which is LeNet in our experiments. Figure 6.12 compares the impacts of DPA-DLF when the adversary uses the true and the substitute architectures. Note that, *impacts of DPA-DLF reduce when the adversary uses the substitute architecture.*

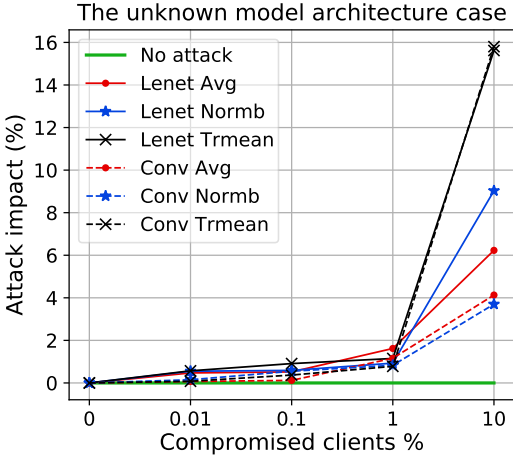


Figure 6.12: Impacts of the DPA-DLF (Section 6.2.1.1) attack, which uses the knowledge of model architecture, reduce if the architecture is unknown.

**(Takeaway 6.3.4.4)** The DPAs that rely on a surrogate model (e.g., our DLF) are less effective if the architectures of the surrogate and global models do not match.

Table 6.3: The architecture of the surrogate model that we use to emulate the unknown architecture setting (Section 6.3.4.4).

Layer name	Layer size
Convolution + Relu	$5 \times 5 \times 32$
Max pool	$2 \times 2$
Convolution + Relu	$5 \times 5 \times 64$
Max pool	$2 \times 2$
Fully connected + Relu	1024
Softmax	62

### 6.3.5 Evaluating Robustness of Cross-silo FL

In cross-silo FL, each of  $N$  clients, i.e., silos (e.g., corporations like banks, hospitals, insurance providers, government organizations, etc.), collects data from many *users* (e.g., bank customers or hospital patients) and collaboratively train the FL model; we denote the total number of users by  $N'$ .

Recall from Section 6.1.4.2 that the model poisoning adversary completely breaks into the devices of compromised clients and, to be effective, persists in their systems for long duration because model poisoning attacks are online attacks (Section 6.1.4.2). For cross-silo FL, this means that the adversary should break into large corporations, e.g., a bank, who are bound by contract and have professionally maintained software stacks. Plausible cross-silo poisoning scenarios involve strong incentives (e.g., financial) and require multiple parties to be willing to risk the breach of contract by colluding or for one party to hack thereby risking criminal liability. This makes breaking into these silos practically unlikely, hence we argue that *model poisoning threats in cross-silo FL are impractical*.

Note that this is unlike the large scale data-breaches [7, 1, 2] which are short-lived and are only capable of stealing information, but not changing the infrastructure.

Hence, we only study the data poisoning threat for cross-silo FL. For worse-case analyses, we assume that the silos train their models on all the data contributed by their users. If the silos inspect the users' data and remove the mislabeled data, one

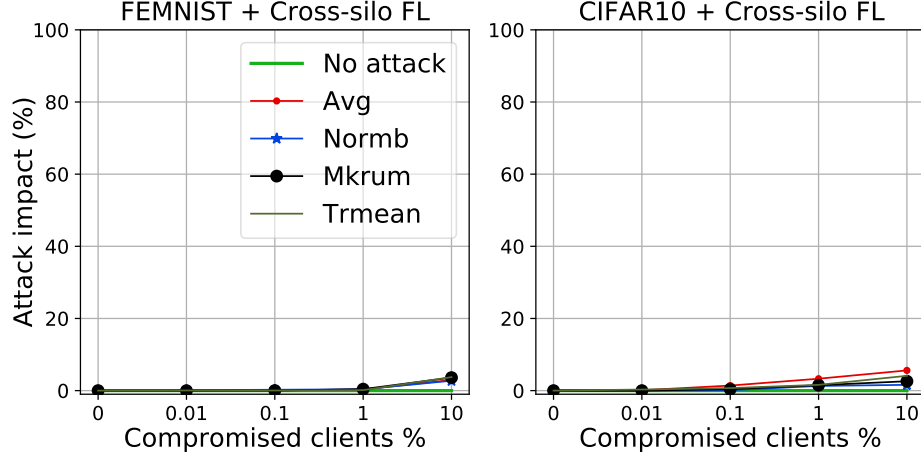


Figure 6.13: All data poisoning attacks have negligible impacts on cross-silo FL, when compromised clients are concentrated in a few silos or distributed uniformly across silos (Section 6.3.5).

should consider clean-label data poisoning attacks [129, 69]; we leave this study to future work. Note that, data inspection is not possible in cross-device FL as data of clients (who are also the users) is completely local, hence clean-label poisoning is not relevant in cross-device FL.

We assume that each silo collects data from equal number (i.e.,  $N'/N$ ) of users. For DPAs, we assume  $M\%$  of the  $N'$  users are compromised and each of them shares poisoned data  $D_p$  (computed as described in Section 6.2.1.1) with their parent silo; as discussed in Section 6.1.3, we assume  $|D_p| = 100 \times |D|_{\text{avg}}$  for *each user*. We distribute the compromised users either *uniformly* across the silos or *concentrate* them in a few silos. For instance, consider 50 silos and 50 compromised users and that, each silo can have a maximum of 50 users. Then in the uniform case, a single compromised user shares her  $D_p$  with each silo, while in the concentrated case, all the 50 compromised users share their  $D_p$  with a single silo.

Figure 6.13 shows the impacts of best of DPAs for the concentrated case. We see that *cross-silo FL is highly robust to state-of-the-art DPAs*. Because, in the concentrated case, very large numbers of *benign silos mitigate the poisoning impact of*



*the very few ( $M\%$ ) compromised silos.* We observe the same results for the uniform distribution case, because very large numbers of *benign users in each silo mitigate the poisoning impacts of the very few ( $M\%$ ) compromised users.*

**(Takeaway 6.3.5)** In production cross-silo FL, model poisoning attacks are not practical, and state-of-the-art data poisoning attacks have no impact even with Average AGR.

## 6.4 Takeaways

Numerous recent works have proposed sophisticated aggregation rules for FL with strong theoretical robustness guarantees [30, 108, 168, 13, 165, 123, 54, 61]. However, our work shows that, when it comes to production FL deployments, even simple, low-cost defenses can effectively protect FL against poisoning. We also believe that our systematization of practical poisoning threat models can steer the community towards practically significant research problems in FL robustness. Complete details of above contributions are in [134]. Furthermore, using systematization academics and practitioners can build various threat models of poisoning attacks in different FL settings of their interest. Furthermore, they can analyze how useful these threat models are in practical FL deployments, which we hope will lead to more practically relevant contributions to FL robustness literature.

## CHAPTER 7

# ROBUSTNESS EVALUATION OF PERSONALIZED FEDERATED LEARNING

In this thesis so far, we focused on evaluating robustness of the conventional federated learning (FL) algorithms, e.g., FedAvg and FedSGD, with and without robust aggregation rules (AGRs). However, in the past couple of years it has become crystal clear that the extremely heterogeneity of FL settings has adverse impacts on the utility of resulting FL models [75, 74, 82, 156] makes it important to *personalize* the global FL model to the distribution of each of the participating users to improve the utility of FL. Consequently, literature has introduced a very large number of *personalized FL (PFL)* algorithms [99, 149, 169, 102, 56, 52, 91, 124, 109, 140, 170], that aim to tune the global model to a client’s data, e.g., using techniques as simple as fine-tuning [169] to complex algorithms, e.g., that use Monreau envelopes [149]. Given the importance of personalization in FL and proliferation of PFL algorithms, in this chapter, we take a closer look at how personalization affects robustness of FL.

First we briefly motivate the need for personalization and why should we care about the robustness of PFL algorithms against poisoning. Then we introduce state-of-the-art PFL algorithms that we evaluate in this chapter. Next, using state-of-the-art untargeted and backdoor attacks, we evaluate robustness of the PFL algorithms and present key observations and conclusions.

### 7.1 Need for personalization in FL

Real-world FL settings are most of the times extremely heterogeneous [82], because the participating clients have local datasets with (mildly to significantly) different

distributions. To understand this, we present an excellent example from the work of Yu et al. [169]. Consider two common FL tasks: next-word prediction using Reddit data and image classification using CIFAR10 data. There are 80,000 and 100 total clients for Reddit and CIFAR10, respectively; we omit further experimental setup details as it is not required to demonstrate the heterogeneity in FL.

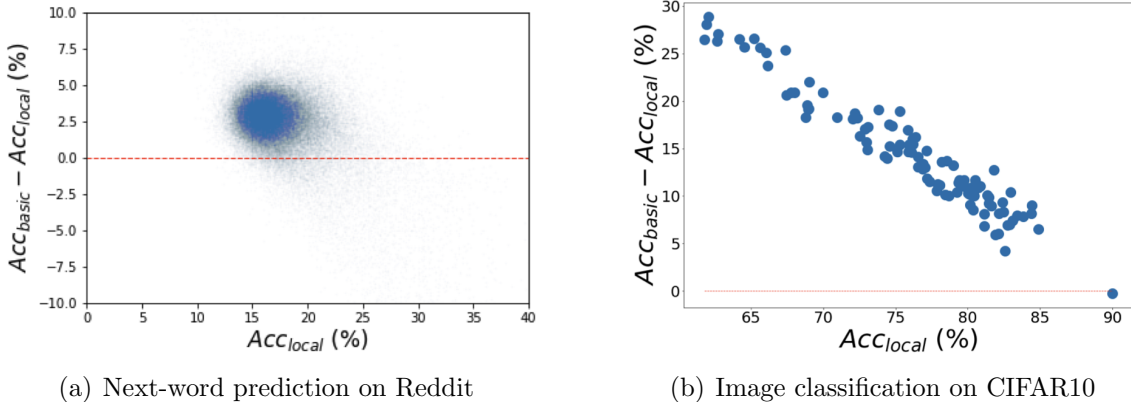


Figure 7.1: Conventional FL algorithms, e.g., FedAvg, compute a single global model for all clients. As we can see in the two plots here, such single model cannot provide good performance for all the clients. In fact, for many clients (as in the case of Reddit) the accuracy of the model trained only on the local data is more than the accuracy of the global model on their local test data. This implies participating in FL does not benefit these clients.

To produce Figure 7.1, for each task, we first train a single global model using FL and all clients’ data. Then for each client we train a local model on their local training data. Then for each client, the plots show the difference in the accuracy of the global FL model ( $Acc_{basic}$ ) and the accuracy of local model ( $Acc_{local}$ ) on the client’s local test data, i.e., it plots  $(Acc_{basic} - Acc_{local})$ . To interpret the plots, note that whenever  $(Acc_{basic} - Acc_{local}) \leq 0$ , the accuracy of local model is more than or equal to that of the global model for the client, hence the client does not benefit by participating in FL.

We observe from the plots that as the conventional FL algorithms compute a single global model for all the clients, many of the clients (as in case of next-word

prediction using Reddit) do not benefit from FL. In fact the more heterogeneous the FL environment, the more difficult it is for the single global model to cater to the heterogeneous needs of all entire population. This shortcoming of conventional FL algorithms motivates the need for personalized FL, where the goal is to compute one model per client such that every client always benefits from participating in FL.

### 7.1.1 State-of-the-art PFL algorithms that we evaluate

As mentioned above, personalized FL has received significant attention from the research community that has introduced numerous PFL algorithms. However, a recent work by Wu et al. [160] comprehensively evaluated representative PFL algorithms that are can be implemented in real-world cross-silo or cross-device FL settings. They conclude that not all PFL algorithms consider practical constraints and in fact only a few can be used in practice. We discuss more on this next.

First note that, depending on the task of interest, [160] considers multiple metrics to measure the performance of PFL; but we consider three metrics that are relevant to our work: average per-client accuracy, % of clients hurt after using PFL, and a fairness metric (variance of accuracy across clients).

**Key observations from Motley benchmark study:** For cross-device FL, [160] observes that the simplest PFL algorithm, i.e., FedAvg + fine-tuning (FedAvg-FT), works the best most datasets and all the three metrics (Table 4 in [160]). While in the case of cross-silo FL, they observe that when considering the average per-client accuracy and % of clients hurt metrics, FedAvg-FT almost always performs the best, while Ditto performs close to FedAvg-FT. For fairness metric, Ditto almost always outperforms other PFL algorithms.

Hence, based on the observations of [160], we evaluate two PFL algorithms in this work: FedAvg-FT and Ditto. We describe them in detail below.

### 7.1.1.1 FedAvg + Fine-tuning (FedAvg-FT)

FedAvg-FT first trains a global model on the data of all clients participating in FL training. Then this global model is distributed to all the participating FL clients. Each of the FL clients then performs local fine-tuning of the global model using their local dataset to personalize the model to perform well on their local data distribution. The clients can fine-tune any subset of the model parameters; common techniques include fine-tuning the entire model or just the last few layers of the model.

### 7.1.1.2 Ditto

Multi-task learning (MTL) approaches aim to compute personalized models for a set of tasks by learning the task relations (either explicitly or implicitly), and Ditto is also a MTL approach, where different tasks correspond to different FL clients.<sup>1</sup>

The objective of traditional FL which fits a single global model,  $w$ , for all clients is to solve:

$$\min_w G(F_1(w), \dots, F_K(w)), \quad (\text{Global Obj})$$

where  $F_k(w)$  is the local objective for device  $k$ , and  $G(\cdot)$  is a function that aggregates the local objectives  $\{F_k(w)\}_{k \in [K]}$  from each device. For example, in FedAvg [104],  $G(\cdot)$  is typically set to be a weighted average of local losses, i.e.,  $\sum_{k=1}^K p_k F_k(w)$ , where  $p_k$  is a pre-defined non-negative weight such that  $\sum_k p_k = 1$ . However, in general, each device may generate data  $x_k$  via a distinct distribution  $\mathcal{D}_k$ , i.e.,  $F_k(w) := \mathbb{E}_{x_k \sim \mathcal{D}_k} [f_k(w; x_k)]$ . To better account for this heterogeneity, it is common to consider techniques that learn personalized, device-specific models,  $\{v_k\}_{k \in [K]}$  across the network.

---

<sup>1</sup>We use some of the language and notations from the original work [99] as it is to explain Ditto

To this end, Ditto considers two ‘tasks’: the global objective (Global Obj), and the local objective  $F_k(v_k)$ , which aims to learn a model using only the data of device  $k$ . To relate these tasks, we incorporate a regularization term that encourages the personalized models to be close to the optimal global model. The resulting bi-level optimization problem for each device  $k \in [K]$  is given by:

$$\begin{aligned} \min_{v_k} \quad & h_k(v_k; w^*) := F_k(v_k) + \frac{\lambda}{2} \|v_k - w^*\|^2 \\ \text{s.t.} \quad & w^* \in \underset{w}{\operatorname{argmin}} G(F_1(w), \dots, F_K(w)). \end{aligned} \tag{Ditto}$$

---

**Algorithm 8** Ditto for Personalized FL

---

```

1: Input:  $K, T, s, \lambda, \eta, w^0, \{v_k^0\}_{k \in [K]}$ 
2: for  $t = 0, \dots, T - 1$  do
3:   Server randomly selects a subset of devices  $S_t$ , and sends  $w^t$  to them
4:   for device  $k \in S_t$  in parallel do
5:     Solve the local sub-problem of  $G(\cdot)$  inexactly starting from  $w^t$  to obtain  $w_k^t$ :
6:      $w_k^t \leftarrow \text{UPDATE\_GLOBAL}(w^t, \nabla F_k(w^t))$ 
7:     /* Solve  $h_k(v_k; w^t)$  */
8:     Update  $v_k$  for  $s$  local iterations:  $v_k = v_k - \eta(\nabla F_k(v_k) + \lambda(v_k - w^t))$ 
9:     Send  $\Delta_k^t := w_k^t - w^t$  back
10:  end for
11:  Server aggregates  $\{\Delta_k^t\}$ :
12: end for
13: Return:  $\{v_k\}_{k \in [K]}$  (personalized),  $w^T$  (global)

```

---

Ditto solves the Ditto objective jointly by solving for the global model  $w^*$  and personalized models  $\{v_k\}_{k \in [K]}$  in an alternating fashion, as summarized in Algorithm 8. Ditto optimization proceeds in two phases: (i) updates to the global model,  $w^*$ , are computed across the network, and then (ii) the personalized models  $v_k$  are fit on each local device. The process of optimizing  $w^*$  is exactly the same as optimizing for any objective  $G(\cdot)$  in federated settings: If we use iterative solvers, then at each communication round, each selected device can solve the local subproblem of  $G(\cdot)$

approximately (Line 5). For personalization, device  $k$  solves the global-regularized local objective  $\min_{v_k} h_k(v_k; w^t)$  inexactly at each round (Line 6).

## 7.2 A robustness evaluation personalized FL

### 7.2.1 Why should we care about robustness of PFL?

We believe that, due to numerous attractive properties of personalization in FL, all the future applications of FL will adopt personalization in some fashion. Second, although majority of PFL works focus on providing FL with improved performance for all clients, multiple recent PFL works [169, 99] have gone beyond just FL performance and claimed that the personalization in FL can provide intrinsic and improved robustness against poisoning attacks on FL. However, as we will show these works do not perform a comprehensive analyses of their proposed PFL algorithm, e.g., Ditto [99] uses simplest of untargeted and backdoor attacks to evaluate their robustness. And, as we discussed in Chapter 4, such robustness claims based on sub-optimal attacks may lead to vulnerable FL applications.

Hence, we perform a thorough robustness analysis of two state-of-the-art and practical PFL algorithms (FedAvg-FT and Ditto) for two reasons: 1) to understand and quantify the threat of FL poisoning to PFL, as personalization is increasingly becoming important to FL, and 2) to verify the robustness claims of some of the recent PFL works.

### 7.2.2 Methodology

To evaluate the robustness of the two PFL algorithms, we consider both backdoor and untargeted poisoning attacks. In this work, we do not design new poisoning attacks against the PFL algorithms and instead evaluate them against existing state-of-the-art attacks. The reason for this is as follows: Both the PFL algorithms we consider perform local fine-tuning over the global model (FedAvg-FT does it once at

the end of FL training, while Ditto does it iteratively at each FL round). Hence, both these PFL algorithms (and many others [124, 52, 140] that use local fine-tuning) have to rely on the robust aggregation rules (AGRs) they use at the server for the overall robustness of their PFL algorithm. For example, as the baseline Ditto algorithm uses simple averaging at the server, to make it robust against poisoning, Ditto proposes to use various robust AGRs, e.g., Median, Multi-krum and Norm-bounding, during server’s aggregation step. Therefore, to appropriately assess the robustness of (robust) PFL algorithms, we use the state-of-the-art poisoning attacks that are tailored to specific robust AGRs that these PFL algorithms use. The specific backdoor and untargeted attacks we use are detailed below:

### 7.2.2.1 The backdoor attack we use

Here, we first justify the choice of our backdoor attack and then provide its details. Literature has proposed a number of backdoor attacks [18, 164, 155, 172] against FL. However, as Sun et al. [148] show the early attacks [18, 164] that rely on a technique called *boosting* (here the adversary scales their model update poisoned with backdoor by a large constant) are ineffective against simplest of robust AGRs, i.e., Norm-bounding. To address this issue, Wang et al. [155] propose a novel backdoor attack, called *edge-case attack*, that targets the clients who data distribution lies on the tail of the overall FL data distribution; we use this attack for evaluation in this work.

Once the adversary introduces a backdoor into the FL model, its impact generally peaks after a few rounds but then starts to diminish; this is especially true for cross-device FL where backdoors are more relevant in practice. This is because in cross-device FL, it is highly unlikely that the server will select an adversary-controlled device multiple times during the lifespan of FL training, unless the adversary controls a large number of devices which as we argued in Chapter 6 makes the attack very costly and prohibitive. To address this issue Zhang et al. [172] propose a technique to



make a backdoor attack *more durable*. We omit evaluating Neurotoxin, because we evaluate how effective the PFL algorithms are at removing the backdoors from the most corrupt global model, i.e., the model for which backdoor accuracy is the highest during FL training.

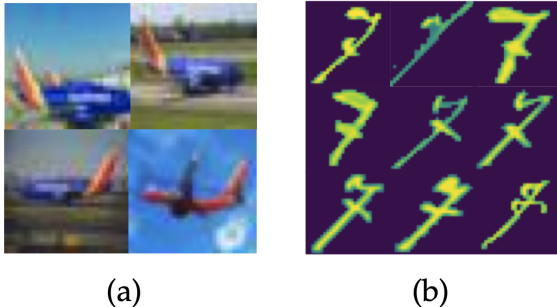


Figure 7.2: Samples from the tail of the distribution of CIFAR10 (a) and FEMNIST (b) datasets. (a) Images of Southwest airline planes which are not in the original CIFAR10 data although it has multiple plane images (b) Images of digit '7' from ARDIS dataset [93] written in a different style than '7' in the original FEMNIST data.

**Details of edge-case backdoor attack:** There are three steps of this attack.

- **Collect tail data:** First the adversary acquires some edge-case data, i.e., the data from the tail of the distribution of overall FL data. For instance, in case of FEMNIST digits data, [155] uses handwritten digits written in Swedish style. Such different styled data is not in train or test data of the original FEMNIST hence they are good candidates for tail data. For CIFAR10 which has images of arbitrary planes with label “plane”, one can use plane images of a specific airline, e.g., Southwest, as the tail data, because such Southwest airline images are not in either train or test split of original CIFAR10. Examples of such tail data are shown in Figure 7.2.
- **Generate poisoning data from tail data:** Next adversary creates poisoning data,  $D_p$ , using these tail data; specifically, they assign the incorrect target label,  $y^t$ , to these images to generate pairs of  $(\mathbf{x}, y^t)$ . For instance, for FEMNIST, we

assign label  $y^t = 1$  to all the images of digit 7 from ARDIS data to generate  $D_p$ . Then we populate  $D_p$  on the malicious clients’ devices that the adversary controls.

- **Compute malicious, backdoored model updates using  $D_p$ :** There are multiple way to obtain malicious update by training on  $D_p$ . As we focus on model poisoning attacks in this work, we consider the projected gradient decent (PGD) attack from [155]. Under this attack, malicious clients apply PGD on the losses for  $D' = D \cup D_p$ , where  $D$  is the benign FEMNIST data owned by the malicious client. That is they periodically project the model on the ball centered around the global model of the previous iteration. A heuristic choice for the radius of this ball is the maximum norm difference allowed by the server’s robust AGR, e.g., norm threshold in case of the Norm-bounding defense.

### 7.2.2.2 Untargeted attack we use

We use our own state-of-the-art untargeted poisoning attacks (Chapter 4) tailored to specific robust AGRs that we use with the PFL algorithms (Norm-bounding, Multikrum and Trimmed-mean) to evaluate their robustness.

### 7.2.3 Experimental setup

Below is the experimental setup we use:

#### 7.2.3.1 Datasets and model architectures

We use FEMNIST dataset for evaluations and Section 4.3 provides the details of the dataset; we use the LeNet [96] architecture. However, in this section, we use the dataset splitting strategy proposed in [160] which are different for cross-device and cross-silo FL. We explain these the splitting strategies below.

**Cross-device:** We divide the 3,400 clients in FEMNIST data in three groups: training, validation and test clients. We use training clients (with all of their data) to train

the FL model, e.g., using FedAvg algorithm. For validation and test client, we split their data in two parts: personalization and test. We use the personalization data for personalization, e.g., using fine-tuning, and validate the performance of personalized model on the test data. For hyperparameters tuning, we use validation clients and finally report the accuracy using test clients.

**Cross-silo:** We use 300 out of 3,400 FEMNIST clients for cross-silo experiments. We split each client’s data into three splits: train, validation and test. We use train data of all clients to train the FL model. When personalizing the model as well, we use the train data. We use validation data to perform tuning of hyperparameters of FL and personalization algorithms. Finally we report accuracy on clients’ test data.

### 7.2.3.2 Attack settings

**Backdoor attack:** For FEMNIST, we use ARDIS [93] as the tail data and use the same data split as in [155] for the attack. Specifically, for cross-device setting, we use 660 images of digit 7 for training and 100 images for testing. Out of 660, we use  $p\%$  of images for generate poisoning data  $D_p$  for backdoor and add this to the data of a single client who is the malicious client. We distribute the rest among 5 benign FEMNIST clients. We add the 100 test images to one of test clients, and use 50 of them for personalization and test on the rest of 50 images. Note that there is only a single client with poisoning data; we use *fixed-frequency* attack setting where the single attacking client participates every  $t_f$  rounds, starting from round 1,000.

While for cross-silo setting, we use  $p\%$  of the entire 760 images of digit 7 for generating  $D_p$  and add it to the data of a single client, who is the only malicious client in population. We add the rest of the  $(100 - p)\%$  data with correct labels to 1 or 2 of the benign clients. Note that in cross-silo setting both the benign and malicious clients participate in each round.

**Untargeted attack:** We use  $p\%$  of the 3,400 (in case of cross-device) or 300 (in case of cross-silo) clients as the malicious clients, we vary  $p \in \{1, 5, 10, 15, 20\}$ . Note that these clients are all among the training client population of cross-device and cross-silo FL.

### 7.2.3.3 FL training hyperparameters

For each setting, we first obtain the best possible accuracy in the benign setting and then use the same parameters for evaluating the robustness of corresponding PFL algorithm. For cross-device FL, we randomly select 30 out of 3,400 FEMNIST clients and train for 1,500 rounds where server optimizer is SGD with learning rate 1. For local training, each client uses 5 local epochs, batch size of 32, and SGD optimizer with momentum of 0.9, weight decay of  $1e^{-4}$  and learning rate of  $0.05 \times 0.999^t$  where  $t$  is the round number. For cross-silo FL, we use all 300 clients in each round and train for 300 rounds where server optimizer is SGD with learning rate 1. For local training, each client uses 2 local epochs, batch size of 32, and SGD optimizer with momentum of 0.9, weight decay of  $1e^{-4}$  and learning rate of  $0.05 \times 0.995^t$  where  $t$  is the round number. For Ditto, we search the parameter  $\gamma \in \{0.1, 0.2, 0.5, 1, 2\}$  as suggested in [99].

For Norm-bounding, we use norm threshold of 2 for cross-device and 3 for cross-silo FL. For both Trimmed-mean and Multi-krum defenses, we assume that the server knows exact number of attackers participating in each FL round.

### 7.2.3.4 Measurement metrics

We use standard measurement metrics in our evaluations. For the edge-case backdoor attacks, we compute *backdoor accuracy*  $A_{bd}$  that measures the percentage of edge-case test data from the source class are classified into the target class, i.e., in our case, the percentage of ARDIS 7’s classified as 1. We also measure main task accuracy, i.e., the accuracy on the benign test data, i.e., in our case, data from all

FEMNIST classes and data from all classes, except class 7, of ARDIS. For untargeted attacks, we measure the main task accuracy, i.e., percentage of correctly classified FEMNIST test data.

#### 7.2.4 Experimental results

In this section, we provide the experimental results. First we demonstrate the PFL in fact improves the average per-client accuracy, then we demonstrate how PFL performs against backdoor and untargeted poisoning attacks.

##### 7.2.4.1 Personalized FL improves average per-client accuracy

First, in Table 7.1, we show that the personalized FL algorithms improve the average per-client accuracy for our specific cross-device and cross-silo settings of FEMNIST.

**For cross-device FL**, accuracy of local training and simple FedAvg are 59.68% and 85.69%, respectively, while local fine-tuning (FedAvg-FT) achieves 90.01% accuracy. Note also that FedAvg-FT improves the overall fairness across test clients: the standard deviation of client accuracies is 8.27% for FedAvg while it is 5.68% for FedAvg-FT. However, we also note that FedAvg-FT *can hurt some of the FL clients*: For 6.14% (43/700) test clients the FedAvg global model performs better than individual fine-tuned local models. This is because we use the same hyperparameters for all the FL clients, which may not be suitable for some of the clients. Note that FedAvg-FT performance for these clients can be improved by personalizing the fine-tuning hyperparameters, e.g., making local  $lr = 0$  will achieve the accuracy same as that of the global model.

**For cross-silo FL** as well, we make similar observations. More specifically, FedAvg-FT (Ditto) improves over local-training and FedAvg by 29.05% (28.03%) and 3.24% (2.22%), respectively. Furthermore, both FedAvg-FT and Ditto significantly reduce the standard deviation over client accuracies, i.e., improve fairness.

Table 7.1: Personalization helps improve the overall performance of FL.

FL type	Algorithm	Metric	EMNIST
Cross-device	Local training	Per-client acc	$59.68 \pm 17.21$
	FedAvg + Fine-tuning (FedAvg-FT)	Per-client acc before FT	$85.69 \pm 8.27$
		Per-client acc after FT	$90.01 \pm 5.68$
		% clients hurt	6.14
Cross-silo	Local training	Per client acc	$60.23 \pm 18.01$
	FedAvg + Fine-tuning (FedAvg-FT)	Per-client acc before FT	$86.04 \pm 10.99$
		Per-client acc after FT	$89.28 \pm 9.54$
		% clients hurt	6
	Ditto	Per client acc	$88.26 \pm 10.7$

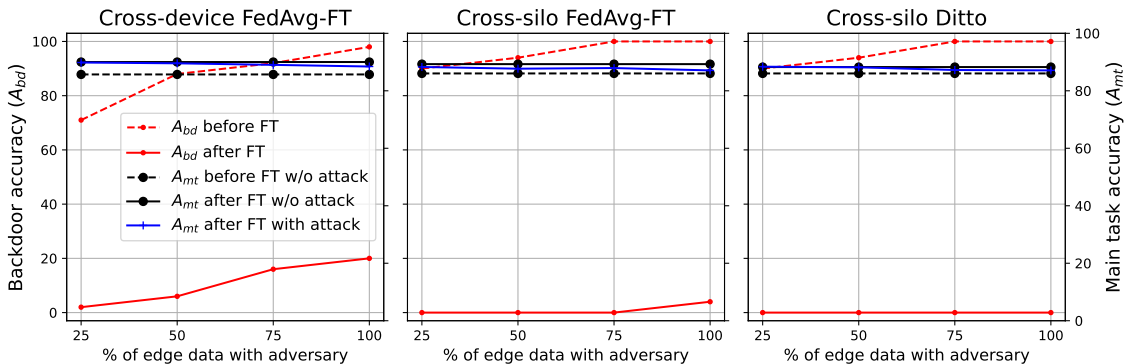


Figure 7.3: Impact of varying the % of total edge data (660 samples in our case) that the adversary holds; benign clients hold rest of the data. Edge-case backdoor attacks are very effective against FedAvg, but personalized FL algorithms are very effective in mitigating their threat to FL. For cross-device FL here, adversary participates every  $10^{th}$  round starting from round 1,000. We use FedAvg + Norm-bounding here.

#### 7.2.4.2 Impact of personalization on edge-case backdoor attacks

We now discuss the impact of personalization on the success of edge-case backdoor attacks in FL. Recall that we have total of 660 data for training from class 7 of ARDIS. Figure 7.3 shows the results when we vary % of 660 training data that the adversary uses to train their malicious update; the benign clients use the rest of the data.

Even with Norm-bounding defense, we note that edge-case backdoor attacks are highly effective against FedAvg. For instance, With 50% edge data, i.e., 330 edge

samples, the attack achieves 88% accuracy against cross-device FedAvg and 91% accuracy against cross-silo FedAvg; this is without any effect on the main task accuracy.

However, **personalization almost completely mitigates the backdoor attack**: In all cases we note that the backdoor accuracy reduces to close to 0%, except in cross-device FedAvg-FT where it reduces to between 15-20%. Furthermore, we do not observe any significant impact on the main task accuracy. We observe the maximum reduction in FedAvg-FT accuracy for the case of cross-device FL: when adversary has 100% of edge-data, the accuracy of FedAvg-FT reduces from 89.28% to 87.1%. The reason behind this is that both FedAvg-FT and Ditto use some form of fine-tuning, and fine-tuning is well-known to remove backdoors from machine learning models.

Next, Figure 7.4 shows the results when we vary the frequency of adversary’s participation in cross-device FL; note that in cross-silo FL, adversary participate in every round by default. Here, the adversary uses 50% of edge data for malicious update training and the rest is with benign clients. We make similar observations as before that although edge backdoor attacks are very effective against traditional FL, personalization effectively mitigates them.

Next, Figure 7.4 shows the results when we vary the frequency of adversary’s participation in cross-device FL; note that in cross-silo FL, adversary participate in every round by default. Here, the adversary uses 50% of edge data for malicious update training and the rest is with benign clients. We make similar observations as before that although edge backdoor attacks are very effective against traditional FL, personalization effectively mitigates them.

### 7.2.4.3 Impact of personalization on untargeted attacks

In this section, we now turn to evaluate the impact of personalization on untargeted poisoning in FL. We use three defenses, i.e., robust AGRs, for this evaluation: Norm-bounding [148], Trimmed-mean [168] and Multi-krum [30]. We combine these three AGRs with each of the three settings we consider, i.e., cross-device FedAvg-

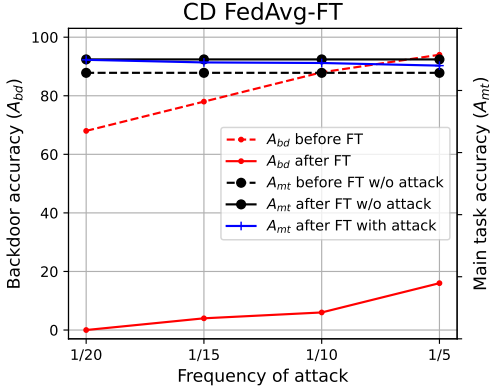


Figure 7.4: Impact of attack frequency.

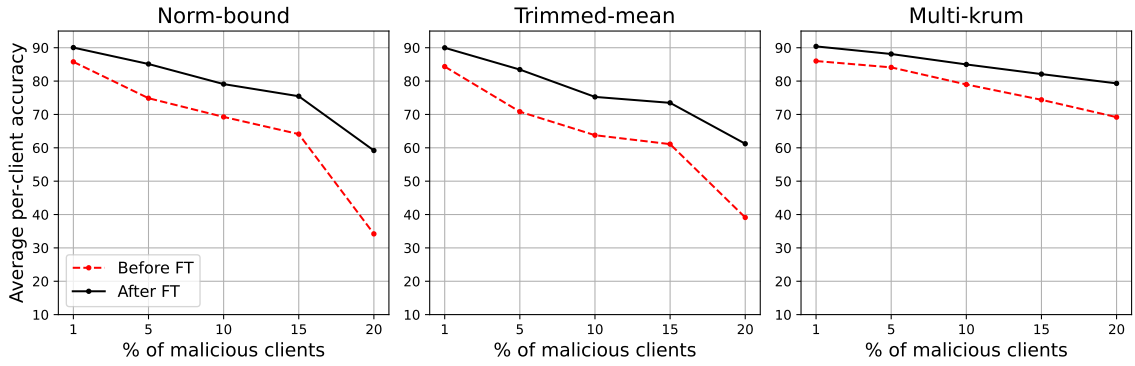


Figure 7.5: Cross-device FL + local fine-tuning

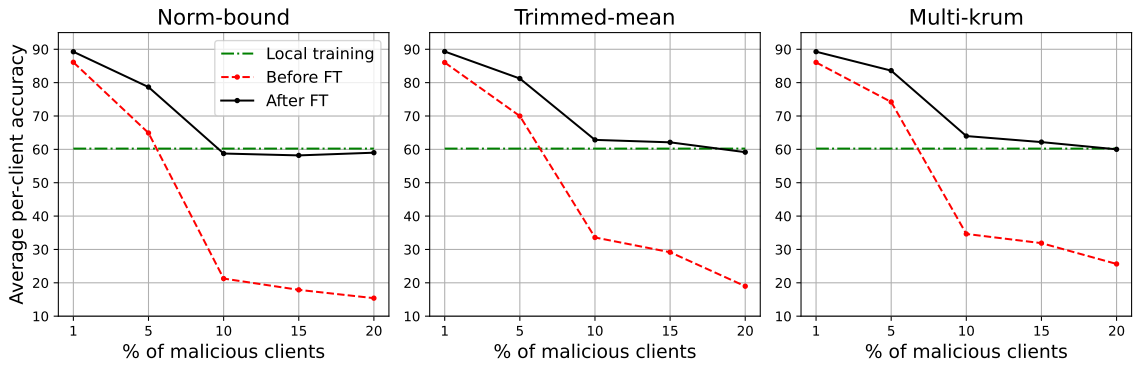


Figure 7.6: Cross-silo FL + local fine-tuning

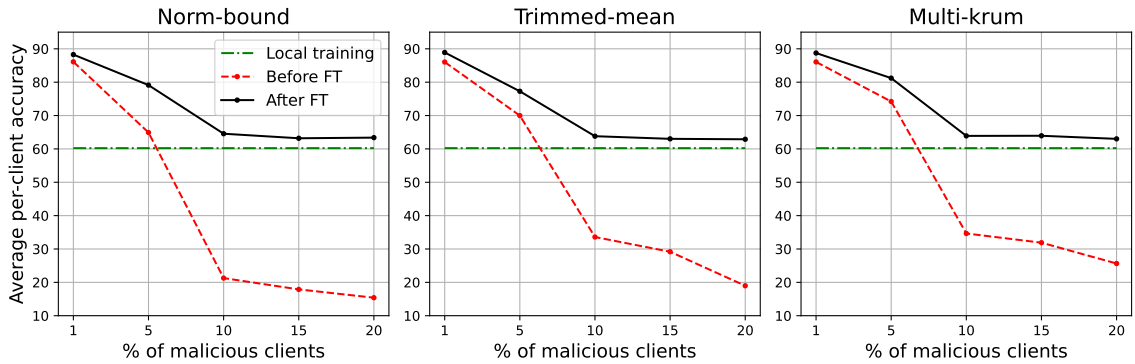


Figure 7.7: Cross-silo FL + Ditto

Figure 7.8: Impact of our state-of-the-art untargeted poisoning attacks (Chapter 4) on PFL algorithms. We make the following observations. First, under production FL settings (e.g., with practical percentages of malicious clients), PFL algorithms remain highly robust. However, personalization algorithms fail to salvage FL against stronger adversary, e.g., with higher percentages of malicious clients.



FT, cross-silo FedAvg-FT and cross-silo Ditto. Figure 7.8 shows results for all the 9 combinations described above. We make two key observations as follows.

**(1) Under real-world, production FL environments, personalized FL remains highly robust to untargeted poisoning.** Recall from Chapter 6 that under real-world settings, an adversary can afford to control a very small percentage, typically  $< 1\%$ , of FL population as malicious clients. We use our state-of-the-art AGR tailored untargeted poisoning attacks from Chapter 4 against Trimmed-mean and Multi-krum and from Section 6.2.1.2 against Norm-bounding. We note from Figure 7.8 that for all 9 settings, these attacks reduce the average per-client accuracy of the personalized FL algorithms by less than 3%.

**(2) Under stronger (theoretically relevant) threat models, personalized FL is vulnerable to poisoning but is much more robust than tradition FL.** As discussed in Chapter 4, majority of prior FL poisoning defenses aim to defend against poisoning by very strong adversaries, e.g., who control up to 25% of FL population as malicious clients. Many of personalized FL algorithms also make similar claims, e.g., Ditto [99] claims that it can defend even when there are 40% malicious clients in FL population! To test these claims, we evaluate the aforementioned 9 combinations under stronger threat models, i.e., with higher % of malicious clients and strong poisoning attacks.

First, we note in Figure 7.8 that all three robust AGRs are highly vulnerable to poisoning by the strong adversary in traditional FL setting. We observe that, with 10% malicious clients, the minimum reduction in accuracy due to our attacks is 10% and it is for cross-device FL with FedAvg + Multi-krum. For cross-silo FL, we observe significantly more reductions of at least 50%. As discussed in Section 4.4.2, this is because all malicious clients poison the global model in every round of cross-silo FL.

Next, we note that, in all the 9 cases for all the percentages of malicious clients, personalization significantly improves the per-client accuracy and reduces the impact

of our attacks. For instance, with 20% malicious clients, in case of cross-device FL, local fine-tuning improves per-client accuracy for Norm-bounding from 34.2% to 59.2%, for Trimmed-mean from 39.1% to 69.2% and for Multi-krum from 69.2% to 79.3%. We make similar observations for other percentages of malicious clients.

Finally, even in the case of cross-silo FL settings, where our attacks have a very large impact on traditional FL, personalization improves the average per-client accuracy. For example, with 20% malicious clients, FedAvg-FT improves the accuracy Norm-bounding from 15.4% to 59.0%, for Trimmed-mean from 19.0% to 59.2% and for Multi-krum from 25.7% to 60.0%. While, with 20% malicious clients, Ditto performs slightly better and improves the accuracy Norm-bounding from 15.4% to 63.4%, for Trimmed-mean from 19.0% to 62.9% and for Multi-krum from 25.7% to 63.0%.

Note that the green horizontal line in Figures 7.6 and 7.7 represents the average per-client accuracy of local-only training, which is 60.23%. Hence, as expected, our observations above suggest that, *under very strong poisoning attacks, personalized FL algorithms can perform at least as good as local-only training* where clients do not collaborate.

### 7.3 Takeaways

The major implications of our study in this chapter are as follows: The carefully tuned personalized FL (PFL) always improves the average *per-client accuracy* over traditional FL. In terms of robustness, PFL is very effective at mitigating backdoor attacks that impact a small portion of the global ML model and remains highly robust to untargeted poisoning under practical, production FL environments. Under strong untargeted poisoning threat models, PFL fails to completely salvage traditional FL, but can significantly improve the average per-client accuracy over traditional FL algorithms.

## BIBLIOGRAPHY

- [1] Billion passwords stolen: Change all of yours, now! <https://www.nbcnews.com/tech/security/billion-passwords-stolen-change-all-yours-now-n174321>, 2014. [Online; accessed 18-July-2021].
- [2] Hackers expose 8.4 billion passwords post them online in possibly largest dump of passwords ever. <https://www.thegatewaypundit.com/2021/06/hackers-expose-8-4-billion-passwords-post-online-possibly-largest-dump-passwords-ever/>, 2014. [Online; accessed 18-July-2021].
- [3] Federated learning: Collaborative machine learning without centralized training data. <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>, 2017. [Online; accessed 06-April-2021].
- [4] Acquire Valued Shoppers Challenge at Kaggle. <https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data>, 2019. [Online; accessed 19-June-2020].
- [5] Utilization of fate in risk management of credit in small and micro enterprises. <https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/>, 2019. [Online; accessed 27-Mar-2021].
- [6] Google workshop on federated learning and analytics. <https://docs.google.com/document/d/1dWzVeFLrPinonQMauxIo0oI-Vbvqup5cZzgdPXvu97Y/edit#heading=h.7dsxad3c3nf7>, 2020. [Online; accessed 06-April-2021].

- [7] 26 million stolen passwords found online — see if you’re affected. <https://www.tomsguide.com/news/mystery-malware-info-stealer>, 2021. [Online; accessed 18-July-2021].
- [8] Google play protect. <https://developers.google.com/android/play-protect>, 2021. [Online; accessed 22-July-2021].
- [9] Safetynet attestation api. <https://developer.android.com/training/safetynet/attestation>, 2021. [Online; accessed 18-July-2021].
- [10] Abadi, Martín, Chu, Andy, Goodfellow, Ian, McMahan, H Brendan, Mironov, Ilya, Talwar, Kunal, and Zhang, Li. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM.
- [11] Agarwal, Naman, Kairouz, Peter, and Liu, Ziyu. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems 34* (2021), 5052–5064.
- [12] Aji, Alham, and Heafield, Kenneth. Sparse communication for distributed gradient descent. In *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing* (2017), Association for Computational Linguistics (ACL), pp. 440–445.
- [13] Alistarh, Dan, Allen-Zhu, Zeyuan, and Li, Jerry. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems* (2018), pp. 4613–4623.
- [14] Alistarh, Dan, Grubic, Demjan, Li, Jerry, Tomioka, Ryota, and Vojnovic, Milan. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems* (2017), pp. 1709–1720.

- [15] Alistarh, Dan, Hoeffler, Torsten, Johansson, Mikael, Konstantinov, Nikola, Khirirat, Sarit, and Renggli, Cédric. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems* (2018), pp. 5973–5983.
- [16] Amid, Ehsan, Ganesh, Arun, Mathews, Rajiv, Ramaswamy, Swaroop, Song, Shuang, Steinke, Thomas, Suriyakumar, Vinith M, Thakkar, Om, and Thakurta, Abhradeep. Public data-assisted mirror descent for private model training. In *International Conference on Machine Learning* (2022), PMLR, pp. 517–535.
- [17] Augenstein, Sean, McMahan, H Brendan, Ramage, Daniel, Ramaswamy, Swaroop, Kairouz, Peter, Chen, Mingqing, Mathews, Rajiv, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679* (2019).
- [18] Bagdasaryan, Eugene, Veit, Andreas, Hua, Yiqing, Estrin, Deborah, and Shmatikov, Vitaly. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics* (2020), PMLR, pp. 2938–2948.
- [19] Balle, Borja, Cherubin, Giovanni, and Hayes, Jamie. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)* (2022), IEEE, pp. 1138–1156.
- [20] Balle, Borja, Kairouz, Peter, McMahan, Brendan, Thakkar, Om, and Guha Thakurta, Abhradeep. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems 33* (2020), 4623–4634.
- [21] Barreno, Marco, Nelson, Blaine, and Joseph, Anthony D. The security of machine learning. *Machine Learning* (2010).

- [22] Baruch, Moran, Gilad, Baruch, and Goldberg, Yoav. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems* (2019).
- [23] Bassily, Raef, Thakkar, Om, and Guha Thakurta, Abhradeep. Model-agnostic private learning. *Advances in Neural Information Processing Systems 31* (2018).
- [24] Bengio, Yoshua, Léonard, Nicholas, and Courville, Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).
- [25] Bernstein, Jeremy, Zhao, Jiawei, Azizzadenesheli, Kamyar, and Anandkumar, Anima. signsgd with majority vote is communication efficient and fault tolerant. In *International Conference on Learning Representations* (2018).
- [26] Berthelot, David, Carlini, Nicholas, Goodfellow, Ian, Papernot, Nicolas, Oliver, Avital, and Raffel, Colin A. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems* (2019), pp. 5050–5060.
- [27] Bhagoji, Arjun Nitin, Chakraborty, Supriyo, Mittal, Prateek, and Calo, Seraphin. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning* (2019), PMLR, pp. 634–643.
- [28] Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *Proceedings of 29th International Conference on Machine Learning* (2012).
- [29] Biggio, Battista, and Roli, Fabio. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* (2018).

- [30] Blanchard, Peva, Guerraoui, Rachid, Stainer, Julien, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems* (2017), pp. 119–129.
- [31] Boenisch, Franziska, Dziedzic, Adam, Schuster, Roei, Shamsabadi, Ali Shahin, Shumailov, Ilia, and Papernot, Nicolas. Is federated learning a practical pet yet? *arXiv preprint arXiv:2301.04017* (2023).
- [32] Bonawitz, Keith, Eichner, Hubert, Grieskamp, Wolfgang, Huba, Dzmitry, Ingerman, Alex, Ivanov, Vladimir, Kiddon, Chloé, Konečný, Jakub, Mazzocchi, Stefano, McMahan, Brendan, Overveldt, Timon Van, Petrou, David, Ramage, Daniel, and Roselander, Jason. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019* (2019), mlsys.org.
- [33] Bonawitz, Keith, Ivanov, Vladimir, Kreuter, Ben, Marcedone, Antonio, McMahan, H Brendan, Patel, Sarvar, Ramage, Daniel, Segal, Aaron, and Seth, Karn. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM.
- [34] Brendan, McMahan H, Ramage, Daniel, Talwar, Kunal, and Zhang, Li. Learning differentially private recurrent language models. *International Conference on Learning and Representation* (2018).
- [35] Briggs, Christopher, Fan, Zhong, and Andras, Peter. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), IEEE, pp. 1–9.

- [36] Caldas, Sebastian, Wu, Peter, Li, Tian, Konečný, Jakub, McMahan, H Brendan, Smith, Virginia, and Talwalkar, Ameet. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).
- [37] Cao, Xiaoyu, Jia, Jinyuan, and Gong, Neil Zhenqiang. Provably secure federated learning against malicious clients. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 6885–6893.
- [38] Cao, Xiaoyu, Jia, Jinyuan, Zhang, Zaixi, and Gong, Neil Zhenqiang. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 IEEE Symposium on Security and Privacy (SP)* (2022), IEEE Computer Society, pp. 326–343.
- [39] Carlini, Nicholas, Hayes, Jamie, Nasr, Milad, Jagielski, Matthew, Sehwag, Vikash, Tramèr, Florian, Balle, Borja, Ippolito, Daphne, and Wallace, Eric. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188* (2023).
- [40] Carlini, Nicholas, Liu, Chang, Erlingsson, Úlfar, Kos, Jernej, and Song, Dawn. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)* (Santa Clara, CA, Aug. 2019), USENIX Association, pp. 267–284.
- [41] Carlini, Nicholas, Tramer, Florian, Wallace, Eric, Jagielski, Matthew, Herbert-Voss, Ariel, Lee, Katherine, Roberts, Adam, Brown, Tom, Song, Dawn, Erlingsson, Úlfar, et al. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805* (2020).
- [42] Chang, Hongyan, Shejwalkar, Virat, Shokri, Reza, and Houmansadr, Amir. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279* (2019).



- [43] Charikar, Moses, Steinhardt, Jacob, and Valiant, Gregory. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (2017), ACM, pp. 47–60.
- [44] Chaudhuri, Kamalika, Monteleoni, Claire, and Sarwate, Anand D. Differentially private empirical risk minimization. *Journal of Machine Learning Research* (2011).
- [45] Chen, Lingjiao, Wang, Hongyi, Charles, Zachary, and Papailiopoulos, Dimitris. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning* (2018), PMLR, pp. 903–912.
- [46] Chen, Ting, Kornblith, Simon, Norouzi, Mohammad, and Hinton, Geoffrey. A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (2020), PMLR, pp. 1597–1607.
- [47] Chen, Wei-Ning, Kairouz, Peter, and Ozgur, Ayfer. Breaking the communication-privacy-accuracy trilemma. *Advances in Neural Information Processing Systems 33* (2020), 3312–3324.
- [48] Chen, Xinyun, Liu, Chang, Li, Bo, Lu, Kimberly, and Song, Dawn. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
- [49] Cho, Yae Jee, Wang, Jianyu, Chiruvolu, Tarun, and Joshi, Gauri. Personalized federated learning for heterogeneous clients with clustered knowledge transfer. *arXiv preprint arXiv:2109.08119* (2021).
- [50] Choquette-Choo, Christopher A, Tramer, Florian, Carlini, Nicholas, and Papernot, Nicolas. Label-only membership inference attacks. In *International conference on machine learning* (2021), PMLR, pp. 1964–1974.

- [51] Cohen, Gregory, Afshar, Saeed, Tapson, Jonathan, and Van Schaik, Andre. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), IEEE, pp. 2921–2926.
- [52] Collins, Liam, Hassani, Hamed, Mokhtari, Aryan, and Shakkottai, Sanjay. Exploiting shared representations for personalized federated learning. In *International Conference on Machine Learning* (2021), PMLR, pp. 2089–2099.
- [53] CS231n: Convolutional Neural Networks for Visual Recognition. <https://cs231n.github.io/optimization-2/#grad>, 2021.
- [54] Data, Deepesh, and Diggavi, Suhas. Byzantine-resilient sgd in high dimensions on heterogeneous data. *arXiv preprint arXiv:2005.07866* (2020).
- [55] Data, Deepesh, Song, Linqi, and Diggavi, Suhas. Data encoding for byzantine-resilient distributed optimization. *IEEE Transactions on Information Theory* (2020).
- [56] Deng, Yuyang, Kamani, Mohammad Mahdi, and Mahdavi, Mehrdad. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461* (2020).
- [57] Diakonikolas, Ilias, Kamath, Gautam, Kane, Daniel, Li, Jerry, Steinhardt, Jacob, and Stewart, Alistair. Sever: A robust meta-algorithm for stochastic optimization. In *International Conference on Machine Learning* (2019), pp. 1596–1606.
- [58] Diakonikolas, Ilias, Kamath, Gautam, Kane, Daniel M, Li, Jerry, Moitra, Ankur, and Stewart, Alistair. Robust estimators in high dimensions without the computational intractability. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (2016), IEEE, pp. 655–664.

- [59] Diakonikolas, Ilias, Kamath, Gautam, Kane, Daniel M., Li, Jerry, Moitra, Ankur, and Stewart, Alistair. Being robust (in high dimensions) can be practical. *In Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017).
- [60] Dwork, Cynthia, Roth, Aaron, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* (2014).
- [61] El-Mhamdi, El-Mahdi, Guerraoui, Rachid, Guirguis, Arsany, and Rouault, Sebastien. Sgd: Decentralized byzantine resilience. *arXiv preprint arXiv:1905.03853* (2019).
- [62] Erlingsson, Úlfar, Feldman, Vitaly, Mironov, Ilya, Raghunathan, Ananth, Song, Shuang, Talwar, Kunal, and Thakurta, Abhradeep. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618* (2020).
- [63] Facebook has shut down 5.4 billion fake accounts this year. <https://www.cnn.com/2019/11/13/tech/facebook-fake-accounts/index.html>, 2019.
- [64] Fang, Minghong, Cao, Xiaoyu, Jia, Jinyuan, and Gong, Neil Zhenqiang. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)* (Boston, MA, aug 2020), USENIX Association.
- [65] Fu, Shuhao, Xie, Chulin, Li, Bo, and Chen, Qifeng. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464* (2019).
- [66] Fung, Clement, Yoon, Chris JM, and Beschastnikh, Ivan. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)* (2020), pp. 301–316.

- [67] Ghosh, Avishek, Chung, Jichan, Yin, Dong, and Ramchandran, Kannan. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems 33* (2020), 19586–19597.
- [68] Ghosh, Avishek, Hong, Justin, Yin, Dong, and Ramchandran, Kannan. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629* (2019).
- [69] Goldblum, Micah, Tsipras, Dimitris, Xie, Chulin, et al. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *arXiv:2012.10544* (2020).
- [70] Gupta, Nirupam, and Vaidya, Nitin H. Randomized reactive redundancy for byzantine fault-tolerance in parallelized learning. *arXiv preprint arXiv:1912.09528* (2019).
- [71] Hayes, Jamie, Melis, Luca, Danezis, George, and De Cristofaro, Emiliano. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 1 (2019), 133–152.
- [72] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.
- [73] He, Lie, Karimireddy, Sai Praneeth, and Jaggi, Martin. Byzantine-robust learning on heterogeneous datasets via resampling. *arXiv preprint arXiv:2006.09365* (2020).
- [74] Hsieh, Kevin, Phanishayee, Amar, Mutlu, Onur, and Gibbons, Phillip. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning* (2020), PMLR, pp. 4387–4398.

- [75] Hsu, Tzu-Ming Harry, Qi, Hang, and Brown, Matthew. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).
- [76] Huang, Ling, Joseph, Anthony D, Nelson, Blaine, Rubinstein, Benjamin IP, and Tygar, J Doug. Adversarial machine learning. In *AISec* (2011).
- [77] Iyengar, Roger, Near, Joseph P, Song, Dawn, Thakkar, Om, Thakurta, Abhradeep, and Wang, Lun. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 299–316.
- [78] Jagielski, Matthew, Oprea, Aline, Biggio, Battista, Liu, Chang, Nita-Rotaru, Cristina, and Li, Bo. Manipulating machine learning: Poisoning attacks and countermeasures against regression learning. *39th IEEE Symposium on Security and Privacy* (2018).
- [79] Jain, Prateek, Kothari, Pravesh, and Thakurta, Abhradeep. Differentially private online learning. In *Conference on Learning Theory* (2012), JMLR Workshop and Conference Proceedings, pp. 24–1.
- [80] Kairouz, Peter, Liu, Ziyu, and Steinke, Thomas. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning* (2021), PMLR, pp. 5201–5212.
- [81] Kairouz, Peter, McMahan, Brendan, Song, Shuang, Thakkar, Om, Thakurta, Abhradeep, and Xu, Zheng. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning* (2021), PMLR, pp. 5213–5225.

- [82] Kairouz, Peter, McMahan, H Brendan, Avent, Brendan, Bellet, Aurélien, Bennis, Mehdi, Bhagoji, Arjun Nitin, Bonawitz, Keith, Charles, Zachary, Cormode, Graham, Cummings, Rachel, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).
- [83] Karimireddy, Sai Praneeth, Jaggi, Martin, Kale, Satyen, Mohri, Mehryar, Reddi, Sashank J, Stich, Sebastian U, and Suresh, Ananda Theertha. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606* (2020).
- [84] Karimireddy, Sai Praneeth, Kale, Satyen, Mohri, Mehryar, Reddi, Sashank, Stich, Sebastian, and Suresh, Ananda Theertha. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning* (2020), PMLR, pp. 5132–5143.
- [85] Ker, Justin, Wang, Lipo, Rao, Jai, and Lim, Tchoyoson. Deep learning applications in medical image analysis. *Ieee Access* 6 (2017), 9375–9389.
- [86] Konečný, Jakub, McMahan, H Brendan, Yu, Felix X, Richtárik, Peter, Suresh, Ananda Theertha, and Bacon, Dave. Federated learning: Strategies for improving communication efficiency. *NIPS Private Multi-Party Machine Learning Workshop* (2016).
- [87] Konstantinidis, Konstantinos, and Ramamoorthy, Aditya. Byzshield: An efficient and robust system for distributed training. *Proceedings of Machine Learning and Systems* 3 (2021).
- [88] Krizhevsky, Alex. Learning multiple layers of features from tiny images. Tech. rep., University of Toronto, 2009.

- [89] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012).
- [90] Kukkala, Vipin Kumar, Tunnell, Jordan, Pasricha, Sudeep, and Bradley, Thomas. Advanced driver-assistance systems: A path toward autonomous vehicles. *IEEE Consumer Electronics Magazine* 7, 5 (2018), 18–25.
- [91] Kulkarni, Viraj, Kulkarni, Milind, and Pant, Aniruddha. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)* (2020), IEEE, pp. 794–797.
- [92] Kurakin, Alexey, Chien, Steve, Song, Shuang, Geambasu, Roxana, Terzis, Andreas, and Thakurta, Abhradeep. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328* (2022).
- [93] Kusetogullari, Huseyin, Yavariabdi, Amir, Cheddad, Abbas, Grahn, Håkan, and Hall, Johan. Ardis: a swedish historical handwritten digit dataset. *Neural Computing and Applications* 32, 21 (2020), 16505–16518.
- [94] Lai, Fan, Dai, Yinwei, Singapuram, Sanjay, Liu, Jiachen, Zhu, Xiangfeng, Madhyastha, Harsha, and Chowdhury, Mosharaf. FedScale: Benchmarking model and system performance of federated learning at scale. In *International Conference on Machine Learning* (2022), PMLR, pp. 11814–11827.
- [95] Lai, Kevin, Rao, Anup, and Vempala, Santosh. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (2016).

- [96] LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [97] Li, Jerry Zheng. *Principled approaches to robust machine learning and beyond*. PhD thesis, Massachusetts Institute of Technology, 2018.
- [98] Li, Liping, Xu, Wei, Chen, Tianyi, Giannakis, Georgios B, and Ling, Qing. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 1544–1551.
- [99] Li, Tian, Hu, Shengyuan, Beirami, Ahmad, and Smith, Virginia. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning* (2021), PMLR, pp. 6357–6368.
- [100] Li, Tian, Sanjabi, Maziar, Beirami, Ahmad, and Smith, Virginia. Fair resource allocation in federated learning. In *International Conference on Learning Representations* (2019).
- [101] Lin, Tao, Kong, Lingjing, Stich, Sebastian U, and Jaggi, Martin. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems 33* (2020), 2351–2363.
- [102] Mansour, Yishay, Mohri, Mehryar, Ro, Jae, and Suresh, Ananda Theertha. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619* (2020).
- [103] McMahan, H Brendan, Andrew, Galen, Erlingsson, Ulfar, Chien, Steve, Mironov, Ilya, Papernot, Nicolas, and Kairouz, Peter. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210* (2018).



- [104] McMahan, H Brendan, Moore, Eider, Ramage, Daniel, Hampson, Seth, and Arcas, Blaise Aguera y. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th AISTATS* (2017).
- [105] Mehta, Harsh, Krichene, Walid, Thakurta, Abhradeep, Kurakin, Alexey, and Cutkosky, Ashok. Differentially private image classification from features. *arXiv preprint arXiv:2211.13403* (2022).
- [106] Mehta, Harsh, Thakurta, Abhradeep, Kurakin, Alexey, and Cutkosky, Ashok. Large scale transfer learning for differentially private image classification. *arXiv preprint arXiv:2205.02973* (2022).
- [107] Melis, Luca, Song, Congzheng, De Cristofaro, Emiliano, and Shmatikov, Vitaly. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 691–706.
- [108] Mhamdi, El Mahdi El, Guerraoui, Rachid, and Rouault, Sébastien. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning* (2018), pp. 3518–3527.
- [109] Mills, Jed, Hu, Jia, and Min, Geyong. Multi-task federated learning for personalised deep neural networks in edge computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 3 (2021), 630–641.
- [110] Minka, Thomas. Estimating a dirichlet distribution, 2000.
- [111] Mozaffari, Hamid, Shejwalkar, Virat, and Houmansadr, Amir. Every vote counts: Ranking-based training of federated learning to resist poisoning attacks. *To appear in USENIX Security Symposium* (2023).

- [112] Muñoz-González, Luis, Biggio, Battista, Demontis, Ambra, Paudice, Andrea, Wongrassamee, Vasin, Lupu, Emil C, and Roli, Fabio. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017), ACM, pp. 27–38.
- [113] Muñoz-González, Luis, Co, Kenneth T, and Lupu, Emil C. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125* (2019).
- [114] Muñoz-González, Luis, Pfitzner, Bjarne, Russo, Matteo, Carnerero-Cano, Javier, and Lupu, Emil C. Poisoning attacks with generative adversarial nets. *arXiv preprint arXiv:1906.07773* (2019).
- [115] Murphy, Kevin. Machine learning: A probabilistic perspective. *The MIT Press* (2012).
- [116] Nasr, Milad, Shokri, Reza, and Houmansadr, Amir. Machine learning with membership privacy using adversarial tuning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018).
- [117] Nasr, Milad, Shokri, Reza, and Houmansadr, Amir. 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).
- [118] Newell, Andrew, Potharaju, Rahul, Xiang, Luo, and Nita-Rotaru, Cristina. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligence and Security Workshop* (2014), pp. 83–93.

- [119] Nguyen, Dinh C, Ding, Ming, Pathirana, Pubudu N, Seneviratne, Aruna, Li, Jun, and Poor, H Vincent. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials* 23, 3 (2021), 1622–1658.
- [120] Papernot, Nicolas, Thakurta, Abhradeep, Song, Shuang, Chien, Steve, and Erlingsson, Úlfar. Tempered sigmoid activations for deep learning with differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 9312–9321.
- [121] Paulik, Matthias, Seigel, Matt, Mason, Henry, Telaar, Dominic, Kluivers, Joris, van Dalen, Rogier, Lau, Chi Wai, Carlson, Luke, et al. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503* (2021).
- [122] Peng, Xingchao, Huang, Zijun, Zhu, Yizhe, and Saenko, Kate. Federated adversarial domain adaptation. In *International Conference on Learning Representations* (2020).
- [123] Pillutla, Krishna, Kakade, Sham M, and Harchaoui, Zaid. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* (2019).
- [124] Pillutla, Krishna, Malik, Kshitiz, Mohamed, Abdel-Rahman, Rabbat, Mike, Sanjabi, Maziar, and Xiao, Lin. Federated learning with partial model personalization. In *International Conference on Machine Learning* (2022), PMLR, pp. 17716–17758.
- [125] Rajput, Shashank, Wang, Hongyi, Charles, Zachary, and Papailiopoulos, Dimitris. Detox: A redundancy-based framework for faster and more robust gradient aggregation. *Advances in Neural Information Processing Systems* (2019), 10320–10330.

- [126] Ramanujan, Vivek, Wortsman, Mitchell, Kembhavi, Aniruddha, Farhadi, Ali, and Rastegari, Mohammad. What’s hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020).
- [127] Reddi, Sashank J, Charles, Zachary, Zaheer, Manzil, Garrett, Zachary, Rush, Keith, Konečný, Jakub, Kumar, Sanjiv, and McMahan, Hugh Brendan. Adaptive federated optimization. In *International Conference on Learning Representations* (2020).
- [128] Rieke, Nicola, Hancox, Jonny, Li, Wenqi, Milletari, Fausto, Roth, Holger R, Albarqouni, Shadi, Bakas, Spyridon, Galtier, Mathieu N, Landman, Bennett A, Maier-Hein, Klaus, et al. The future of digital health with federated learning. *NPJ digital medicine* 3, 1 (2020), 119.
- [129] Shafahi, Ali, Huang, W Ronny, Najibi, Mahyar, Suciu, Octavian, Studer, Christoph, Dumitras, Tudor, and Goldstein, Tom. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems* (2018), pp. 6103–6113.
- [130] Shejwalkar, Virat, Ganesh, Arun, Mathews, Rajiv, Thakkar, Om, and Thakurta, Abhradeep. Recycling scraps: Improving private learning by leveraging intermediate checkpoints. *arXiv preprint arXiv:2210.01864* (2022).
- [131] Shejwalkar, Virat, and Houmansadr, Amir. Reconciling utility and membership privacy via knowledge distillation. *arXiv preprint arXiv:1906.06589* (2019).
- [132] Shejwalkar, Virat, and Houmansadr, Amir. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *The Network and Distributed System Security Symposium (NDSS)* (2021).

- [133] Shejwalkar, Virat, and Houmansadr, Amir. Membership privacy for machine learning models through knowledge transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2021).
- [134] Shejwalkar, Virat, Houmansadr, Amir, Kairouz, Peter, and Ramage, Daniel. Back to the drawing board: A critical evaluation of poisoning attacks on federated learning. *arXiv preprint arXiv:2108.10241* (2021).
- [135] Shejwalkar, Virat, Inan, Huseyin A, Houmansadr, Amir, and Sim, Robert. Membership inference attacks against NLP classification models. In *NeurIPS 2021 Workshop Privacy in Machine Learning* (2021).
- [136] Shen, Shiqi, Tople, Shruti, and Saxena, Prateek. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (2016), pp. 508–519.
- [137] Shokri, Reza, and Shmatikov, Vitaly. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (2015), ACM.
- [138] Shokri, Reza, Stronati, Marco, Song, Congzheng, and Shmatikov, Vitaly. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on* (2017).
- [139] Simonyan, Karen, and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations* (2015).

- [140] Singhal, Karan, Sidahmed, Hakim, Garrett, Zachary, Wu, Shanshan, Rush, John, and Prakash, Sushant. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems 34* (2021), 11220–11232.
- [141] Smith, Adam, Thakurta, Abhradeep, and Upadhyay, Jalaj. Is interaction necessary for distributed private learning? In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), IEEE, pp. 58–77.
- [142] Song, Congzheng, and Raghunathan, Ananth. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security* (2020), pp. 377–390.
- [143] Song, Congzheng, Ristenpart, Thomas, and Shmatikov, Vitaly. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).
- [144] Song, Congzheng, and Shmatikov, Vitaly. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 196–206.
- [145] Song, Congzheng, and Shmatikov, Vitaly. Overlearning reveals sensitive attributes. In *International Conference on Learning Representations* (2019).
- [146] Song, Liwei, and Mittal, Prateek. Systematic evaluation of privacy risks of machine learning models. In *30th {USENIX} Security Symposium ({USENIX} Security 21)* (2021).
- [147] Sun, Gan, Cong, Yang, Dong, Jiahua, Wang, Qiang, Lyu, Lingjuan, and Liu, Ji. Data poisoning attacks on federated machine learning. *IEEE Internet of Things Journal* 9, 13 (2021), 11365–11375.

- [148] Sun, Ziteng, Kairouz, Peter, Suresh, Ananda Theertha, and McMahan, H Brendan. Can you really backdoor federated learning? *NeurIPS Workshop on Federated Learning* (2019).
- [149] T Dinh, Canh, Tran, Nguyen, and Nguyen, Josh. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems 33* (2020), 21394–21405.
- [150] Tang, Xinyu, Mahloujifar, Saeed, Song, Liwei, Shejwalkar, Virat, Nasr, Mirlad, Houmansadr, Amir, and Mittal, Prateek. Mitigating membership inference attacks by self-distillation through a novel ensemble architecture. In *31th {USENIX} Security Symposium ({USENIX} Security 22)* (2022).
- [151] Tolpegin, Vale, Truex, Stacey, Gursoy, Mehmet Emre, and Liu, Ling. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security* (2020), Springer, pp. 480–501.
- [152] Tramèr, Florian, Shokri, Reza, San Joaquin, Ayrton, Le, Hoang, Jagielski, Matthew, Hong, Sanghyun, and Carlini, Nicholas. Truth serum: Poisoning machine learning models to reveal their secrets. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022), pp. 2779–2792.
- [153] Tran, Brandon, Li, Jerry, and Madry, Aleksander. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems* (2018), pp. 8000–8010.
- [154] Wang, Bolun, Yao, Yuanshun, Shan, Shawn, Li, Huiying, Viswanath, Bimal, Zheng, Haitao, and Zhao, Ben Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 707–723.

- [155] Wang, Hongyi, Sreenivasan, Kartik, Rajput, Shashank, Vishwakarma, Harit, Agarwal, Saurabh, Sohn, Jy-yong, Lee, Kangwook, and Papailiopoulos, Dimitris. Attack of the tails: Yes, you really can backdoor federated learning. In *Advances in Neural Information Processing Systems* (2020).
- [156] Wang, Jianyu, Liu, Qinghua, Liang, Hao, Joshi, Gauri, and Poor, H Vincent. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems 33* (2020), 7611–7623.
- [157] Wen, Wei, Xu, Cong, Yan, Feng, Wu, Chunpeng, Wang, Yandan, Chen, Yiran, and Li, Hai. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems* (2017), pp. 1509–1519.
- [158] Wortsman, Mitchell, Ramanujan, Vivek, Liu, Rosanne, Kembhavi, Aniruddha, Rastegari, Mohammad, Yosinski, Jason, and Farhadi, Ali. Supermasks in superposition. In *NeurIPS* (2020).
- [159] Wu, Chen, Yang, Xian, Zhu, Sencun, and Mitra, Prasenjit. Mitigating backdoor attacks in federated learning. *arXiv preprint arXiv:2011.01767* (2020).
- [160] Wu, Shanshan, Li, Tian, Charles, Zachary, Xiao, Yu, Liu, Ziyu, Xu, Zheng, and Smith, Virginia. Motley: Benchmarking heterogeneity and personalization in federated learning, 2022.
- [161] Xiao, Han, Xiao, Huang, and Eckert, Claudia. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence* (2012), pp. 870–875.



- [162] Xiao, Huang, Biggio, Battista, Nelson, Blaine, Xiao, Han, Eckert, Claudia, and Roli, Fabio. Support vector machines under adversarial label contamination. *Neurocomputing 160* (2015), 53–62.
- [163] Xie, Chulin, Chen, Minghao, Chen, Pin-Yu, and Li, Bo. Crfl: Certifiably robust federated learning against backdoor attacks. In *International Conference on Machine Learning* (2021), PMLR.
- [164] Xie, Chulin, Huang, Keli, Chen, Pin-Yu, and Li, Bo. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations* (2019).
- [165] Xie, Cong, Koyejo, Oluwasanmi, and Gupta, Indranil. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116* (2018).
- [166] Xu, Jie, Glicksberg, Benjamin S, Su, Chang, Walker, Peter, Bian, Jiang, and Wang, Fei. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research 5* (2021), 1–19.
- [167] Yang, Chaofei, Wu, Qing, Li, Hai, and Chen, Yiran. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340* (2017).
- [168] Yin, Dong, Chen, Yudong, Ramchandran, Kannan, and Bartlett, Peter. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning* (2018).
- [169] Yu, Tao, Bagdasaryan, Eugene, and Shmatikov, Vitaly. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758* (2020).
- [170] Zhang, Jie, Guo, Song, Ma, Xiaosong, Wang, Haozhao, Xu, Wenchao, and Wu, Feijie. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems 34* (2021), 10092–10104.

- [171] Zhang, Jie, Li, Bo, Chen, Chen, Lyu, Lingjuan, Wu, Shuang, Ding, Shouhong, and Wu, Chao. Delving into the adversarial robustness of federated learning. *AAAI* (2023).
- [172] Zhang, Zhengming, Panda, Ashwinee, Song, Linyue, Yang, Yaoqing, Mahoney, Michael, Mittal, Prateek, Kannan, Ramchandran, and Gonzalez, Joseph. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning* (2022), PMLR, pp. 26429–26446.
- [173] Zhou, Hattie, Lan, Janice, Liu, Rosanne, and Yosinski, Jason. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *NeurIPS* (2019).
- [174] Zhu, Zhuangdi, Hong, Junyuan, and Zhou, Jiayu. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning* (2021), PMLR, pp. 12878–12889.