Decoding FL Defenses: Systemization, Pitfalls, and Remedies

***Abstract*— While the community has designed various defenses to counter the threat of poisoning attacks in Federated Learning (FL), there are no guidelines for evaluating these defenses. These defenses are prone to subtle pitfalls in their experimental setups that lead to a false sense of security, rendering them unsuitable for practical deployment. In this paper, we systematically understand, identify, and provide a better approach to address these challenges. First, we design a comprehensive systemization of FL defenses along three dimensions: i) how client updates are processed, ii) what the server knows, and iii) at what stage the defense is applied. Next, we thoroughly survey 50 top-tier defense papers and identify the commonly used components in their evaluation setups. Based on this survey, we uncover six distinct pitfalls and study their prevalence. For example, we discover that around 30% of these works solely use the intrinsically robust MNIST dataset, and 40% employ simplistic attacks, which may inadvertently portray their defense as robust. Using three representative defenses as case studies, we perform a critical reevaluation to study the impact of the identified pitfalls and show how they lead to incorrect conclusions about robustness. We provide actionable recommendations to help researchers overcome each pitfall.**

I. Introduction

Federated learning (FL) [59] is an emerging approach in machine learning (ML) where multiple data owners, called *clients*, collaboratively train a shared model, known as the *global model*, while keeping their individual training data private. The central *server* (service provider) iteratively aggregates *model updates* from each client, which are generated based on their local data. The server merges these updates using an *aggregation rule* (AGR) and uses them to update the global model. Following each training iteration (also known as *round*), the refined global model is distributed to the clients participating in the next round. Prominent distributed platforms such as Google's Gboard [2] for next-word prediction, Apple's Siri [1] for automatic speech recognition [72], and WeBank [93] for credit risk predictions, have adopted this FL mechanism. Its intrinsic characteristic of promoting collaboration while preserving privacy has rendered it indispensable in critical applications, notably in medical diagnosis [29], [46], [75], activity recognition [107], [26], [68], [86], and next-character prediction [87].

FL is gaining popularity due to its privacy-preserving and collaborative nature, yet it faces vulnerabilities to *poisoning attacks* [28], [84], [83], [91], where malicious or *compromised clients* intentionally corrupt FL training and *poison* the global model. This can result in a poisoned model that performs poorly on all inputs in *untargeted poisoning* attacks or on specific inputs in *targeted poisoning* attacks. To address these threats, the community has developed various defense mechanisms. Robust AGRs such as Multi-krum [14] and Trimmed-mean [103], detect and discard malicious updates. Certified defenses like CRFL [97] and Ensemble FL [20] provide robustness certifications. Tools like FLDetector [106] proactively identify and remove malicious clients during training. Meanwhile, FedRecover [18] focuses on post-poisoning recovery after an attack, aiming to restore the global model's performance. Ditto [52] integrates fairness and robustness by regularizing the local training objective, and Cronus [23] enhances security and privacy through knowledge distillation.

Systemizing FL Defenses: Besides these few defenses, the variety of available options (Table IV) poses a challenge for practitioners, i.e., determining the right defense for a specific use case or integrating multiple defenses for enhanced robustness becomes complex without a clear understanding of where a defense and its dependencies fit in the FL pipeline. To address these research gaps, we conduct a comprehensive systemization (§III) of FL defenses, organizing them along three crucial dimensions: processing of client updates, server's knowledge, and defense phase. While existing works [83], [40], [10], [13], [38], [80] have designed taxonomies primarily focused on adversarial ML, including those that guide the selection of the appropriate attacks and settings for FL, a dedicated systemization for FL defenses has been lacking.

To the best of our knowledge, we are the first to propose such a systemization. Our framework simplifies the selection and integration of defenses, clarifying when and where each defense is applied (§III-A). For example, Figure 2 shows that FLDetector operates in the pre-aggregation phase("when") at the server("where"). Moreover, the systemization highlights underrepresented defense types, encouraging further exploration and innovation. For example, our analysis identified FedRecover as the only post-training, update-modification technique using estimation, prompting exploration of not only other estimation-based but also post-training defenses. From our systemization in Table I, we can see that most defenses are on the server side, operate during pre-aggregation, and employ metric-based processing of client updates. Importantly, our systemization is designed to be expandable to incorporate more attributes in the future.

Pitfalls in experimental setups of FL defenses: During our systemization, we find that defenses are evaluated across

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23xxx
www.ndss-symposium.org

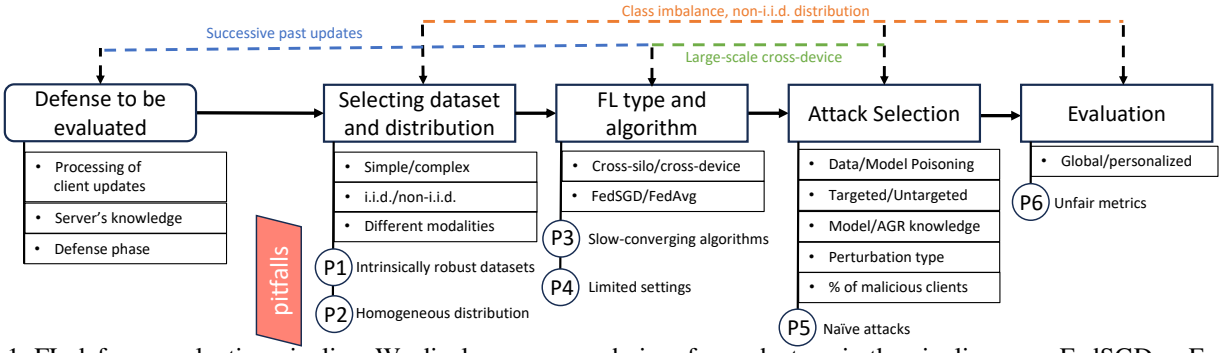


Figure 1: FL defense evaluation pipeline. We display common choices for each stage in the pipeline, e.g., FedSGD or FedAvg as the FL algorithm, and highlight the associated pitfall, i.e., using a slow-converging algorithm. We also indicate interdependencies between stages, e.g., large-scale and cross-device in FL type limit the number of malicious clients in the attack.

various experimental setups with different choices of datasets, data distributions, attacks, etc. We thoroughly survey 50 top-tier FL defense works (§IV) and report frequently used choices of six experimental setup components: *data and distribution, FL type and algorithm, attack, and evaluation type*. We find questionable trends in these choices, e.g., most works use the intrinsically robust MNIST dataset and resort to simple attacks such as label flipping. This motivates the need to uncover *pitfalls* in the experimental setups of existing FL defense works and provide guidelines for future evaluations.

Existing literature provides guidelines for robustness evaluations. For instance, [6] offers insights and recommendations, focusing on pitfalls in centralized ML-based security system evaluations. Similarly, [22] identifies pitfalls in evaluating adversarial robustness and suggests mitigation guidelines. In [83], authors question trends in existing attack threat models, demonstrating FL robustness under practical limitations. However, a comprehensive exploration of FL defenses and their experimental setups is lacking.

To fill this gap in the literature, we identify six distinct pitfalls in the evaluation setups of existing works based on our survey in §IV. We take inspiration from [43] in identifying pitfalls in FL defenses and building on top of their work. Not only do we explore additional pitfalls and analyze their impact in much more diverse settings, but we also first perform an extensive systemization to justify the choices we make in pitfall analysis. For example, Figure 2 and Table I help us to select and justify three distinct defenses for the impact analysis of pitfalls in §V.

Figure 1 shows the pitfalls at the point of their occurrence in the FL training pipeline. For instance, choosing an intrinsically robust dataset is a pitfall that occurs in the dataset selection stage. We also show the common choices for each stage of the pipeline, such as global or personalized evaluation in the last stage. Figure 1 also highlights the interdependencies between stages. For example, non-i.i.d. distribution in the distribution stage influences the choice of evaluation metrics (§V-F), while large-scale limits the threat model in the attack stage (§V-D). We thoroughly explain each pitfall and its prevalence in §IV, and provide actionable recommendations to overcome them. Finally, in §V, we perform a thorough impact analysis of the identified pitfalls using three representative defenses and show how we can avoid them by following our recommendations. These pitfalls *can also be applied to attacks*, e.g., only evaluating an attack in cross-silo settings and not considering its efficacy in the cross-device setting is a pitfall. Similarly, other pitfalls can arise from attacks, as attacks and defenses are inherently interconnected—essentially two sides of the same coin working together. ***However, for the purpose of this paper, we will analyze the pitfalls from the lens of defenses.***

In summary, we make the following contributions:

1. Systemization of FL Defenses: We perform a comprehensive systemization of FL defenses along three dimensions: processing of client updates, server's knowledge, and the defense phase (§III).

2. Identifying Major Pitfalls: We comprehensively review 50 top-tier FL defense works and identify six prevalent pitfalls. In response to each pitfall, we provide actionable recommendations to guide future research efforts (§IV).

3. Dissecting the Impact of Pitfalls: Guided by our systemization, we choose three representative FL defenses and use them to perform a thorough impact analysis of our identified pitfalls. We show how the pitfalls lead to a false sense of security, and by following our recommendations, the research community can overcome them (§V).

This work is not meant as finger-pointing, particularly to the defenses under evaluation. We have chosen them as representative contributions to the field, humbly employing them as testbeds to offer constructive guidelines for future research.

II. Background

A. Federated Learning (FL)

In FL [41], [59], [44], a service provider, called *server*, trains a *global model*, θ^g, on the private data from multiple collaborating clients, all without directly collecting their individual data. During the t^{th} FL round, the server selects n out of total N clients and shares the most recent global model (θ_g^t) with them. Then, a client k uses their local data D_k to compute an update ∇_k^t and shares it with the server. These updates serve as a client's contribution towards refining a global model. Depending on how a client computes their update, FL algorithms

can be broadly divided [59] into *FedSGD* and *FedAvg*. In *FedSGD*, a client computes the update by sampling a subset b from their local data and calculating a gradient of loss $\ell(b;\theta_g^t)$ of the global model on the subset, i.e., $\nabla_k^t = \partial\ell(b;\theta_g^t)/\partial\theta_g^t$. In *FedAvg*, a client k *fine-tunes* θ_g^t on their local data using stochastic gradient descent (SGD) for a fixed number of local epochs E, resulting in an updated local model θ_k^t. The client then computes their update as the difference $\nabla_k^t = \theta_k^t - \theta_g^t$ and shares ∇_k^t with the server. Next, the server computes an aggregate of client updates using an AGR, f_{agg}(such as mean), i.e., using $\nabla_{\mathsf{agg}}^t = f_{\mathsf{agr}}(\nabla_{\{k\in[n]\}}^t)$. Finally, the server updates the global model of the $(t+1)^{th}$ round using SGD as $\theta_g^{t+1} \leftarrow \theta_g^t + \eta\nabla_{\mathsf{agg}}^t$ with server's learning rate η. Due to these differences, FedAvg achieves faster convergence and attains higher accuracy than FedSGD [59].

After discussing the update process of FL models and the collaboration between servers and clients, we present the FL applications (*where it is deployed*) and setups (*how it is used*). There are two main types of deployments: **cross-device** and **cross-silo**, as explained in [41]. In *cross-device FL*, N is very large (ranging from a few thousand to billions) [78], and only a tiny fraction of them are chosen in each FL training round ($n \ll N$). These clients are typically resource-constrained devices such as mobile phones, smartwatches, and other IoT devices [2], [107]. Contrastingly, in *cross-silo FL*, N is moderate (up to 100) [49], and all clients are selected in each round ($n = N$). These clients are typically large corporations, including banks [93] and hospitals [66].

B. Poisoning Attacks in FL

There are various poisoning attacks in literature [14], [11], [12], [9], [60], [28], [57], [99], [63], [84]. An *untargeted* poisoning attack aims to lower the test accuracy for all test inputs indiscriminately [28], [11], [60], [57], [99]. A *targeted* poisoning attack [12], [9] lowers the accuracy on specific test inputs. For instance, in *backdoor attacks* [9] (a sub-category of targeted attacks), the goal is to misclassify only those test inputs that have an embedded *backdoor trigger*. Since these attacks only affect a subset of inputs, they are much weaker than untargeted attacks.

We can also divide the attacks based on the adversary's capabilities. In model poisoning attacks [28], [11], [60], [99], [12], [9], [84], the adversary is strong enough to access and perturb the model gradients on malicious devices before they are sent to the server in every training round. A data-poisoning adversary [63] is much weaker than the model poisoning adversary [11], [28], [83], [84] as it can only poison the datasets on malicious devices.

1) Attacks used in our Study: In our evaluation, we focus on untargeted model poisoning attacks *as they are stronger* [83].

Stat-Opt [28]: provides a generic model poisoning method and tailors it to specific AGRs such as TrMean [103], Median [103], and Krum [14]. The adversary first calculates the mean of the benign clients' updates, ∇^b, and finds the *static* malicious direction $w = -sign(\nabla^b)$. It directs the benign

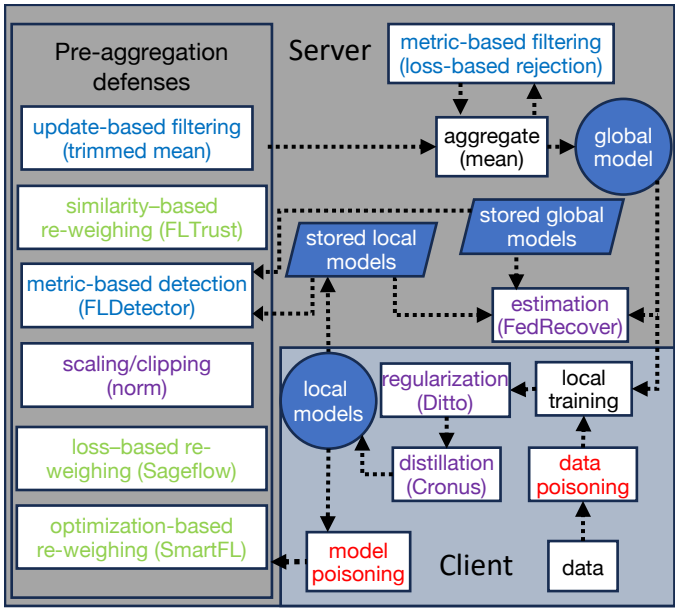


Figure 2: Systemizing FL Defenses: Categorization of defenses based on their *defense phases*. Processing operations for client updates are color-coded: filtering (blue), re-weighing (green), and modification (purple). Defenses belonging to the same category of processing may have different phases, e.g., FedRecover performs update modification at the server, while Ditto performs update modification at the client. Dependencies are also highlighted, such as FedRecover utilizing stored local and global models for estimation.

average along the calculated direction and scales it with γ to obtain the final poisoned update, $-\gamma w$.

Dyn-Opt [84]: proposes a general poisoning method and tailors it to specific AGRs, similar to Stat-Opt. The key distinction lies in the *dynamic* and *data-dependent* nature of the perturbation. The attack starts by computing the mean of benign updates, ∇^b, and a data-dependent direction, w. The final poisoned update is calculated as $\nabla^{'} = \nabla^b + \gamma w$, where the attack finds the largest γ that can bypass the AGR. They compare their attack with Stat-Opt and show that the dataset-tailored w and optimization-based scaling factor γ make their attack a much stronger one.

2) Threat Model in Our Study: Below, we detail the threat models for poisoning attacks used in our study.

Goal: Our adversary is *untargeted*, crafts malicious updates and sends them to the server, where, upon aggregation with other updates, it indiscriminately lowers the accuracy of the global model for all test inputs.

Knowledge: In line with most defense approaches, we assume that the adversary knows the AGR used by the server. Unless explicitly specified, the adversary has complete knowledge of the gradients of both malicious and benign clients. In some cases, we employ a partial knowledge adversary, where the adversary knows the server's AGR but not the gradients of benign clients.

Capabilities: Our untargeted model poisoning adversary controls m out of N clients. The adversary is strong enough to

Table I: A systematic overview of FL defenses, helping practitioners in 1) selecting defenses aligned with their use cases, 2) combining multiple defenses for heightened performance, and 3) designing new defense approaches by gaining insights into the FL defenses.

Dimension	Types	Attributes	Description	Defenses
Processing of Client Updates	Filtering	Update-based	Filters updates by comparing update values with each other.	TrMean [103], Krum [14], GeometricMean [73]
		Metric-based	Filters updates by comparing metrics associated with each update.	MST-AD & Density-AD [77], LF-Fighter [39], ERR & LFR [28], FLDetector [106]
	Update re-weighing	Similarity-based	Filters updates by comparing their similarity with a reference.	CONTRA [8], FLTrust [19]
		Loss-based	Applies weights based on loss for an update.	Sageflow [71], Anomaly Detection [51]
		Optimization-based	Applies weights based on an optimization problem.	SmartFL [100]
	Update modification	Scaling/Clipping	Scales or clips an update if it exceeds a certain threshold.	Norm Bounding [88], Signguard [102]
		Distillation	Distills an update into a low-dimensional vector.	Cronus [23], Auror [85]
		Regularization	Introduces regularization to control robustness and privacy.	Ditto [52]
		Estimation	Estimates a benign update from historical information.	FedRecover [18]
Server's Knowledge	Knowledge of data	No-knowledge	Server has no knowledge of data or its distribution at the client side.	TrMean [103], FedRecover [18], FLDetector [106], Cronus [23]
		Partial knowledge	The server uses a small auxiliary dataset.	Sageflow [71], SmartFL [100], FLTrust [19]
	Knowledge of updates	No-knowledge	Server does not have knowledge of local model updates.	Auror [85], Cronus [23]
		Full knowledge	Server has complete knowledge of local model updates.	FLCert [21], Krum [14], Anomaly Detection [51]
Defense Phase	Aggregation	Pre-aggregation	Processing of client updates is done before updates are aggregated.	MST-AD & Density-AD [77], Bulyan [60], TrMean [103], Krum [14], Sageflow [71]
		Post-aggregation	Processing of client updates is done after aggregation.	FLCert [21], ERR & LFR [28]
	Non-aggregation	Local training	The defense component is part of the local training.	FLIP [105], Ditto [52], Cronus [23]
		Post-training	The defense is applied after training.	FedRecover [18]

manipulate model updates of the malicious clients it controls and has access to the global model parameters shared every round. We set the proportion of malicious clients at 20% (unless stated otherwise), a common benchmark in prior studies [84], [28], [19], [18], which also examined how varying this percentage impacts the severity of attacks. To ensure consistency and comparability with these works, we adhere to the same 20% setting in our implementation.

III. Systemization of Defenses Against FL Poisoning

Here, we introduce a systematization for FL defenses and use it to rationalize the selection of three representative defenses from the literature. Later, in §V, we use these chosen defenses to conduct a comprehensive impact analysis of the pitfalls outlined in §IV.

A. Classification of Defenses

Here, we present the three key dimensions along which we classify FL defenses in literature, as shown in Table I. In Figure 2, we group defenses according to the third dimension, i.e., *defense phase*, and highlight their dependencies.

1) Processing of client updates: Client model updates undergo several processing steps before they are aggregated at the server. The commonly used processing operations are:

Filtering updates to entirely or partially eliminate local updates. *Filtering* defenses fall into two main categories: those based on the values of local model updates, termed *update-based filtering* [103], [14], [73], and those relying on some metrics associated with local models, known as *metric-based filtering* [77], [39], [28], [106].

a) *Update-based filtering [103], [14], [73]* is based on the values of local model updates. It further divides into *dimension-wise* and *vector-wise* filtering. Dimension-wise filtering defenses, such as TrMean [103] and Median [103], filter out malicious values along each update dimension, while vector-wise filtering defenses, such as, Krum [14], remove entire malicious updates.

b) *Metric-based filtering [77], [39], [28], [106]* relies on some metrics associated with local models, e.g., FLDetector [106] uses a *suspicious score* as the metric to identify and remove malicious clients from the training process. We describe FLDetector in detail in §C. In *loss-based rejection* [28], the loss associated with and without incorporating an update for aggregation is calculated, and updates with higher loss are removed. Similarly, *error-based rejection* [28] removes updates by assessing the error instead of the loss.

Update re-weighing involves assigning a weight to each local update, reflecting its perceived level of maliciousness. Various re-weighting approaches exist, including:

a) *Similarity-based re-weighting [8], [19]* is shown by FLTrust [19], where the server assigns a trust score to each client based on the similarity of its updates to the server's update, computed on a small dataset.

b) *Loss-based re-weighting [71], [51]* illustrated by Sageflow [71], involves the server assigning a weight to each update based on local model loss on a small dataset at the server.

c) *Optimization-based re-weighting* is demonstrated by SmartFL [100], which assigns weights through an optimization problem with the same number of parameters as that of clients.

Update Modification changes the update itself to safeguard the global model from the impact of malicious updates. The key techniques within this approach are:

a) *Scaling Updates* limits a local update by clipping it if it exceeds a certain threshold, for example, defenses like Norm-bounding [88], [102].

b) *Distilling update knowledge [23], [85]* is another facet of update modification that involves avoiding the transmission of the entire local model update to the server due to the *curse of dimensionality* [23], which increases the risk of higher impact from poisoning attacks. Instead, clients send distilled information to the server. In Cronus [23], clients send soft labels to the server, and the aggregate is used to update local models. This defense mitigates the risk of poisoning attacks and directly prevents whitebox inference attacks, as the server cannot access the local model parameters.

c) *Regularization* defenses perform regularization to achieve personalized [36], [35] client models. An example is Ditto [52], which modifies the local training objective by introducing a regularization term to control the tradeoff between privacy and robustness of the local model.

d) *Estimation* is exemplified by FedRecover [18], which leverages past information and estimation to recover the unpoisoned global model. We discuss the mechanism of FedRecover in detail in §C.

2) Server's Knowledge: The defense is generally applied at the server, where it collects all local updates and strives to obtain the best possible global model that is least affected by malicious updates. The server's knowledge varies across different defenses and can be described in terms of data and local model updates:

Knowledge of Data: In the *no-knowledge* setting [103], [18], [106], [23], as the name suggests, the server lacks information about the data used for training and testing. It only possesses knowledge of the collected local model updates, examples of which include TrMean [103], Krum [14], and Median [103]. Conversely, in the *partial-knowledge* setting [71], [100], [19], the server possesses a small dataset, which it deploys in various ways, such as calculating entropy associated with updates [71] or assigning a trust score to each client [19] to enhance aggregation robustness against attacks.

Knowledge of Local Model Updates: In the *full-knowledge* setting [21], [14], [51], the server has complete access to the model parameters of all clients, representing the widely used scenario. In the *partial-knowledge*, or *distilled-knowledge* setting [85], [23], the server only has access to some distilled form of the model parameters, such as the output layers [23].

3) Defense Phase: We categorize defenses based on the training *phase*, specifying when and where in the training pipeline the defense is applied.

Aggregation-based defenses are applied during the aggregation phase. These defenses can be further categorized into *pre-aggregation* [77], [60], [103], [14], [71] defenses that perform processing of client updates such as dimension-wise filtering before aggregating updates, or *post-aggregation* [21], [28], [20] defenses, e.g., Ensemble FL [20] that creates all possible aggregations of k models from N clients, then selects the most frequent predicted label as the correct one.

Non-aggregation-based defenses that are not performed at aggregation can be further categorized based on their *phase*:

a) *During Local Training:* [105], [52], [23] The defense takes place at the client's side during local training. For instance, Ditto [52] uses regularization in client training to control the deviation of benign local models from poisoned global models.

b) *Post-Training:* FedRecover [18], employing a recovery-based mechanism, falls into this category as it requires historical information from one training session to estimate the un-poisoned global model during the *recovery* phase.

B. Our Selected Case Study Defenses

We choose three defenses for brevity to analyze the impact of pitfalls in §V; TrMean, FLDetector, and FedRecover. Due to space constraints, we defer their detailed descriptions to Appendix C.

We justify that the chosen defenses are distinct along the dimensions in Table I. TrMean [103] is a ***pre-aggregation, update-based filtering*** defense that removes malicious components of client updates dimension-wise during training. FLDetector [106] is a ***pre-aggregation, metric-based filtering*** defense designed to detect and remove malicious clients during training by analyzing their updates. The metric FLDetector uses is the *suspicious score*, which measures the consistency between the actual update and an estimated one. It is important to note that although TrMean and FLDetector seem similar in their filtering mechanisms, they are different and have distinct approaches. TrMean filters *components* of updates before aggregating them, while FLDetector removes entire *clients* from the training process if they deviate too much from an estimated reference update, which is calculated using historical information. FedRecover [18] is a ***post-training, update modification*** defense that uses *estimation* to *recover* from a previously poisoned global model. It is an advanced defense that uses TrMean in its aggregation phase and relies on a detection mechanism to remove malicious clients before it starts the recovery process. In §V-A2 and §V-E3, we show the performance of FedRecover when we perform Stat-Opt on TrMean and FLDetector, respectively.

All three defenses do not require any auxiliary dataset at the server, and we prefer this setting because the server might not have access to the dataset in practical, real-world scenarios. Similarly, we prefer the *full-knowledge* of client updates scenario, as most of the works [21], [14], [51], [52], [71], [19] use this setting, and it leads to a stronger adversarial setting. Although the server in all three of our chosen defenses has full knowledge of the client model updates, the defenses differ in the amount of historical information needed. Figure 2 clearly shows these different dependencies. TrMean does not require model updates from the past rounds and performs filtering using client model updates of the current round. FLDetector requires updates from the past *few* rounds and uses them to calculate the malicious score for updates in the current round. Since FedRecover is a post-training defense, it requires updates from all the rounds in the original training. Therefore, the volume of updates required is highest for FedRecover and lowest for TrMean.

IV. Pitfalls in FL Defense Evaluations

After presenting the detailed systemization of defenses, it is imperative to unveil critical pitfalls in FL robustness evaluations. By scrutinizing 50 defenses (Table IV in Appendix), we link each pitfall to specific components in the FL workflow (Figure 1). We examine each pitfall's prevalence (Figure 3) across the 50 works, discuss their implications, and conclude with practical recommendations.

Pitfall-1: Intrinsically robust datasets. The chosen datasets are intrinsically robust and lead to incorrect conclusions about a defense's performance.

Description: A designed defense may seem to perform well against specific attacks upon evaluation [103]. However, the evaluation dataset might be inherently robust because it is simple and lacks complexity. Therefore, in such situations, we cannot tell if an attack is mitigated due to the inherent robustness of the dataset or the effectiveness of the defense.

Prevalence and implications: While it is intuitive that a defense mechanism's performance inevitably varies across datasets, using overly simple datasets like MNIST fails to yield

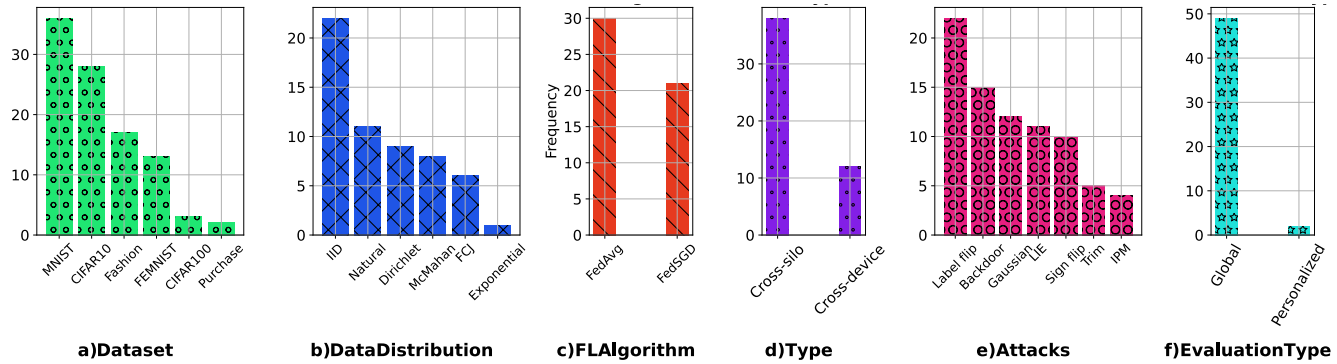


Figure 3: Frequency of choices of the six key components (Dataset, distribution of clients' data, FL algorithm, FL type, attacks, and evaluation) of robustness evaluation setup. § V discusses the impacts of choices on the robustness of FL poisoning defenses.

meaningful insights into the true robustness of a defense mechanism, (§V-A). MNIST is a class-balanced dataset used in FL using synthetic techniques such as Dirichlet Distribution [61]. Conversely, real-world FL tasks are complex and characterized by highly class-imbalanced datasets (§V-B1).

Despite efforts to create open-source datasets mirroring real-world scenarios [17], [25], ***our survey reveals that MNIST remains predominant, constituting 30% of the works*** **[42], [103], [20], [94], [48], [55]** (Figure 3a). CIFAR10 and FashionMNIST, though commonly used, lack true FL representation due to their class-balanced nature (§V-B1). FEMNIST [17], a real-world dataset specifically curated for FL, is used by only 20%. Another interesting observation from Figure 3a is the *exclusive reliance on image-classification datasets*, despite the popularity of language and vision-language models in contemporary research.

Recommendations: Future Evaluations should consider using FL tasks of varying complexities, such as FEMNIST and CIFAR10, for classification and exploring other modalities, such as language, for NLP tasks. In our evaluations in §V we use image classification datasets, FEMNIST and CIFAR10, and a language dataset, StackOverflow, in §V-D1 for large-scale FL evaluation.

Pitfall-2: Homogeneous client data distributions. Using i.i.d. (homogeneous) distributions with low heterogeneity may create a deceptive sense of system robustness that does not reflect real-world complexities.

Description: Evaluating a defense on a particular dataset may yield perceived robustness due to inherent homogeneity (§V-B1) in the dataset distribution rather than the efficacy of the defense technique itself.

Prevalence and implications: We find that ***around 50% of the works, use i.i.d. distributed data, despite evidence that it is easier to defend against such distributions*** [28], [84], [11], [104] (as seen in Figure 3b). The second most common approach involves a natural data distribution where each sample is associated with a client such as StackOverflow [3]. Other artificial distributions include *FCJ* [28], *Dirichlet (Dir)* [9], [78] and *Mcmahan* [59]. In §V-B1, we prove that Dirichlet more closely aligns with real-world distributions, informing our subsequent analysis in §V-B on how defense performance varies with different distributions and levels of heterogeneity.

Recommendations: Future works should prioritize the use of real-world datasets to provide a more realistic evaluation. When working with class-balanced datasets like MNIST, FashionMNIST, and CIFAR10, it is crucial to distribute them among clients heterogeneously. This approach aims to mimic, to some extent, the diversity found in real-world distributions.

Pitfall-3: Slow-converging algorithms. Evaluations often overlook the use of state-of-the-art, fast-converging algorithms, thereby compromising robustness.

Description: The robustness evaluation of an FL defense may heavily rely on the choice of FL algorithms, such as FedAvg or FedSGD, used in its implementation. Using a superior algorithm can enhance system robustness by addressing potential weaknesses in the FL algorithm rather than focusing solely on defense improvement.

Prevalence and implications: Despite FedAvg's recognized advantages in performance, faster convergence, and lower communication overhead compared to FedSGD [59], ***approximately 40% of prior works employ the slow-converging FedSGD algorithm for evaluations*** (Figure 3c). This suboptimal choice contributes to a larger window for attacks and results in a more significant accuracy drop (§V-C).

Recommendations: Future evaluations should prioritize state-of-the-art, fast-converging FL algorithms to remove any weaknesses (such as slow convergence) associated with the FL algorithm.

Pitfall-4: Limited FL settings. Considering only limited scenarios while ignoring practical limitations and factors related to scalability, such as computation, communication, cost, and storage.

Description: The performance of a defense can vary when constrained by real-world limitations, e.g., cost constraints may lead to selecting a very low percentage of malicious clients. Also, the computation, communication, cost, and storage overhead associated with scaling up the system might not be feasible in practical scenarios.

Prevalence and implications: Only 24% of the works in our survey use the cross-device setting (Figure 3d). As demonstrated by [83] on FEMNIST, CIFAR10, and Purchase datasets with the cross-device setting, using a low percentage

of malicious clients due to cost constraints reduces attack performance. We show that on an even larger scale using the naturally distributed Stackoverflow dataset in the cross-device setting, the attack shows *shows no effect on the non-robust mean AGR* (§V-D1). Additionally, defenses that rely on consistent historical information [106], [18] are incompatible with the cross-device setting (§V-D2) because a client is not selected in every round. We thoroughly discuss the scalability issues of FedRecover and FLDetector in §V-D2.

Recommendations: Future evaluations should include deployment conditions of a much larger scale (cross-device) and ensure that the defense provides significant utility compared to the computation and communication cost it incurs. While we do not label the cross-silo setting as impractical, we emphasize designing defenses that are compatible with the cross-device setting as well.

Pitfall-5: Naive attacks. Evaluating defenses solely against simple and naive attacks rather than incorporating strong state-of-the-art attacks makes a defense seem robust.

Description: The true robustness of a defense emerges when tested against strong and adaptive attacks, i.e., attacks tailored for a defense algorithm. Relying on attacks known to be weak and naive for evaluating a new defense will not give us a true picture of the defense's robustness.

Prevalence and implications: Figure 3e shows the frequency of various attacks used in our survey. Despite the existence of several strong poisoning attacks in the literature [84], [28], [99], [11], [91], our analysis reveals that ***about 40% of the works opt for simplistic approaches*** such as random Gaussian [14], label flipping [52], sign flipping [48], bit flipping [98]), even though prior works have shown their poor performance even under strong adversarial settings [83], [28], [48], [74]. We discuss the impact of this pitfall extensively in §V-E by using state-of-the-art attacks, and in §V-E2, we design our own adaptive attack against FLDetector. Our impact analysis reveals that this is one of the most crucial components of FL evaluation.

Recommendations: To correctly evaluate the robustness of a defense, one should use 1) strong adaptive attacks that are tailored to the defense algorithm and maximally reduce its performances and 2) strong state-of-the-art attacks from existing literature.

Pitfall-6: Unfair metrics. Not capturing clients' performances separately and only reporting global accuracy metrics does not give a good measure of per-client robustness.

Description: Data heterogeneity across clients in practical FL systems results in varying performance across clients. This phenomenon is also known as *representation disparity* [37]. Global accuracy does not give us any idea of the individual performances of clients. In addition to this, real-world datasets are class imbalanced [17] as opposed to synthetic datasets such as MNIST and CIFAR10. The commonly reported overall accuracy metric lacks information on per-class performance.

Prevalence and implications: Despite heterogeneity being a well-known issue, only 4% of the surveyed works incorporate personalized evaluations (Figure 3f). Our per-client analysis of TrMean and FedRecover demonstrates that per-client performances vary a lot, highlighting the need to account for these variations in real-world FL systems (§V-F). We also highlight the importance of reporting per-class accuracies, as we show that the overall accuracy is a misleading metric (§V-F3).

Recommendations: Future evaluations should include personalized evaluations along with global evaluations to capture the variation in clients' performances. In addition to this per-client evaluation, future works should also report per-class performance, especially when using class-imbalanced datasets.

V. Impact Analysis of Pitfalls

In this section, we analyze the impact of each identified pitfall. We test representative defenses across diverse setups by following the recommendations outlined in §IV and scrutinizing their implications. This systematic exploration is intended to help researchers make informed decisions about the robustness of FL defenses.

A. Intrinsically Robust Datasets

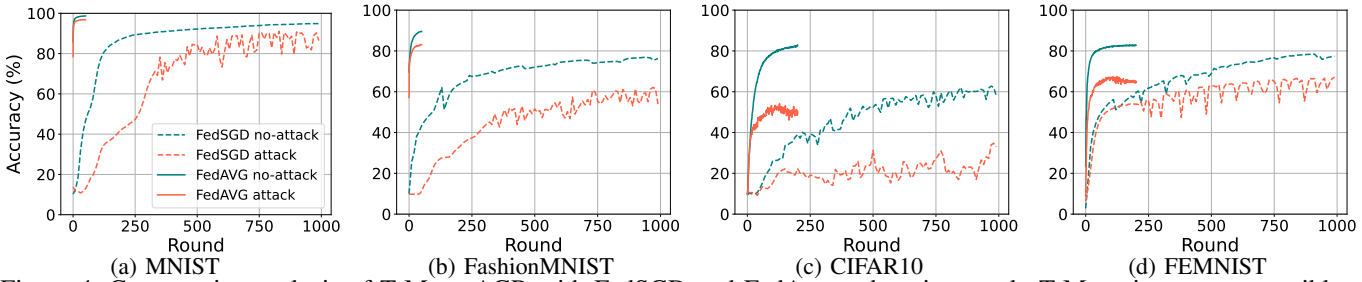
First, we evaluate how the selection of datasets impacts the robustness of our three representative FL defenses: TrMean, FedRecover, and FLDetector.

1) TrMean is only robust with MNIST: To assess TrMean's sensitivity to different datasets, we employ FedAvg and FedSGD on MNIST, FashionMNIST, CIFAR10, and FEMNIST, both without attacks and under Stat-Opt [28]. Results in Figure 4a highlight MNIST's intrinsic robustness. Despite a high (20%) amount of malicious clients, MNIST-trained FL model accuracy drops less than 1% in FedAvg, while other datasets experience more substantial declines, peaking at 50% for CIFAR10.

The variation in performance can be attributed to task complexity. Baseline accuracies in Figure 4 reflect this complexity, with MNIST having the highest no-attack accuracy and CIFAR10 the lowest. From this set of experiments, we can conclude that *TrMean is highly robust using MNIST-based evaluations, but not with other datasets as the evaluations of TrMean using other three datasets, FashionMNIST, FEMNIST, and CIFAR10 show.*

2) FedRecover works better with simple datasets: We evaluate FedRecover on FashionMNIST, FEMNIST, and CIFAR10, in addition to MNIST, since MNIST is heavily evaluated in [18]. We test FedRecover under *recovery-from-benign* and *recovery-from-attack* scenarios. In the former, no attack occurs during original training, while in the latter, Stat-Opt attack is applied during original training but not during recovery (we discuss attack during recovery in §V-E3). Consistent results with [18] on MNIST and FashionMNIST validate our implementation.

Figure 5 shows that even without attacks, FedRecover does not fully recover for complex datasets like FEMNIST and CIFAR10. In recovery from an attack for MNIST (FashionMNIST), FedRecover achieves 91% (72.7%) recovery accuracy from a post-attack accuracy of 80% (64%), where the original training accuracy is 92% (75.2%). For FEMNIST



(a) MNIST (b) FashionMNIST (c) CIFAR10 (d) FEMNIST

Figure 4: Comparative analysis of TrMean AGR with FedSGD and FedAvg under trim attack. TrMean is more susceptible to poisoning with FedSGD due to slow convergence, which gives adversaries more time for poisoning (threat model in §II-B2)

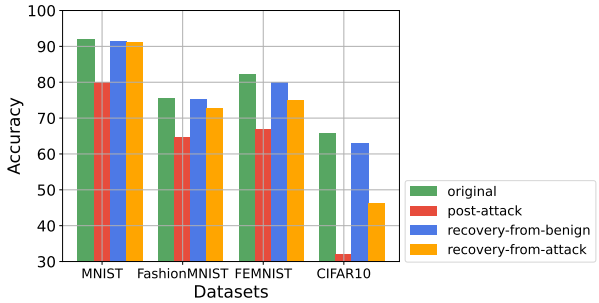


Figure 5: Performance of FedRecover on four datasets, both with and without trim attack in the original training.

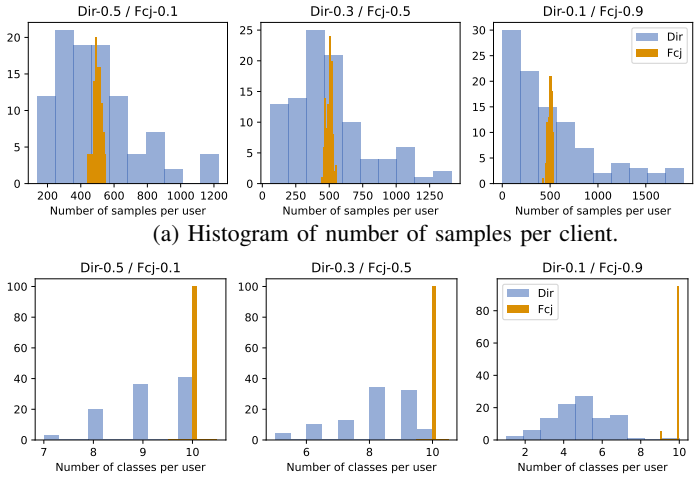
Table II: Impact of data-level perturbations on FLDetector.

dataset	pertubation	FedSGD		FedAvg	
		FPR	FNR	FPR	FNR
MNIST	Noisy-features	0	0	0	0
	Noisy-label	0	0	0	0
Fashion	Noisy-features	0	0	0	0
	Noisy-label	0	0	0	0
FEMNIST	Noisy-features	0	1	0	1
	Noisy-label	0	1	0	1

and CIFAR10, differences between the baseline accuracy and recovery accuracies are 11% and 19%, respectively, indicating an incomplete recovery in the attack setting. This performance variation stems from dataset complexity, with estimation errors higher for complex tasks such as FEMNIST. The impact of *periodic correction* and *warmup* phase on estimation error and recovery performance is discussed in §V-C, §V-D, and §V-E.

3) FLDetector's performance varies with task complexity: We assess FLDetector's performance across varying task complexities using MNIST, FashionMNIST, and FEMNIST. To enhance task complexity, we introduce minor perturbations in features and labels, as we already know the results of these datasets in the unperturbed setting [106]. Table II shows that MNIST and FashionMNIST datasets remain robust, with FLDetector achieving perfect detection, i.e., zero FNR (False Negative Rate) and FPR (False Positive Rate). However, for FEMNIST, FLDetector shows an FPR of 0 and FNR of 1 across all conditions, indicating that it fails to detect any attacks, classifying all malicious clients as benign and allowing them to remain in the training process.

Performance variations across datasets stem from task complexity, extensively discussed in §V-A1 and §V-A2. MNIST and FashionMNIST, less affected by the Stat-Opt attack (Figure 4), have closely clustered benign updates, making it difficult for malicious updates to be stealthy. This proximity aids



(a) Histogram of number of samples per client.

(b) Histogram of number of classes per client.

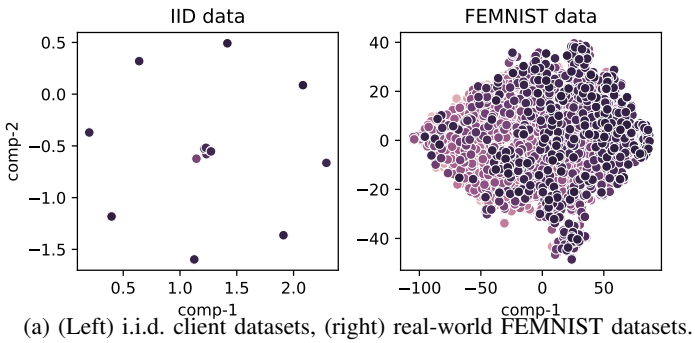
Figure 6: Comparison of Sample and Class Distribution in FL Datasets: Histograms illustrating (a) the number of samples and (b) the number of classes per client, generated using FCJ and Dir distributions. From left to right, the non-i.i.d. degree of generated datasets increases, reflecting the impact of higher FCJ (or Dir) parameters in generating more (or less) non-i.i.d. datasets. We note that all FCJ client datasets are almost the same size, while Dir client datasets are widely varying. Similarly, with FCJ, all clients have all the classes, while with Dir, the number of classes varies widely.

FLDetector in distinguishing between malicious and benign updates. Conversely, FEMNIST, a naturally distributed and more heterogeneous dataset than MNIST and FashionMNIST (§V-B1), results in client updates being further apart. This increased distance enables a malicious update to blend in seamlessly, evading detection by FLDetector.

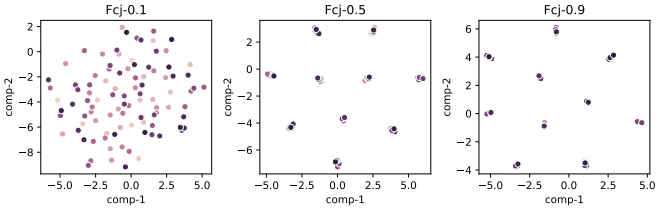
B. Homogeneous Data Distribution

In this section, we first show that the Dir (Dirichlet) distribution is more *real-world* than the FCJ distribution through statistical analyses and visualization. Subsequently, we show the effect of these distributions and their varying levels of heterogeneity on the robustness of TrMean and FedRecover. The original work on FLDetector [106] already analyzed different levels of heterogeneity, so we skip this here.

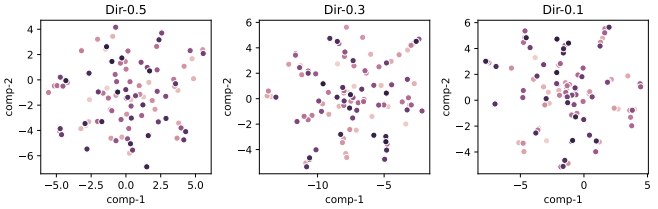
1) Statistical analyses of FCJ and Dir distributions: We consider a classification task with a total of C classes; we



(a) (Left) i.i.d. client datasets, (right) real-world FEMNIST datasets.



(b) FCJ distributed client datasets.



(c) Dir distributed client datasets.

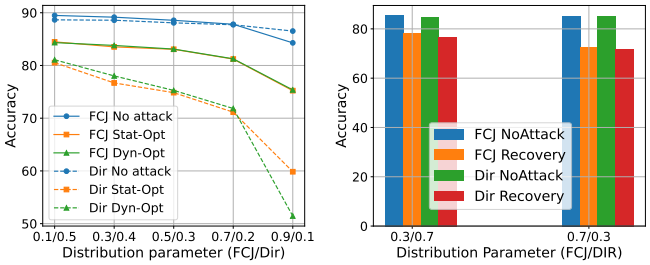
Figure 7: T-SNE projections of class frequency vectors of client datasets generated using FCJ (b) and Dir (c) distributions. From left to right, non-i.i.d. degree increases.

generate client datasets using Dir and FCJ for 100 clients with varying degrees of non-i.i.d. We provide analyses for CIFAR10, but it applies to other datasets. We then plot the following three statistics of the datasets:

(Stat-1): We plot the number of samples per user, which is motivated by the visualizations of real-world FL datasets in Leaf (Figure 14). Figure 6a shows the results for three degrees of non-i.i.d. For Dir we use $\alpha \in \{0.1, 0.3, 0.5\}$ and for FCJ we use bias $b \in \{0.9, 0.5, 0.1\}$. We note that Dir produces client datasets with heterogeneous sizes; the histograms are similar to real-world datasets in the Leaf repository (Figure 14). However, FCJ makes client datasets with almost equal size; note that the FCJ histograms are always concentrated around 500 (the total number of samples in the CIFAR10 dataset / total number of clients).

(Stat-2): In Figure 6b, we show the number of classes each client has when we use Dir or FCJ. We observe that for FCJ, all clients have all the classes except when the bias is 0.9. In the real-world datasets, all clients generally do not have all the classes [59]. Similar to these real-world datasets, the clients have a widely varying number of classes in Dir distribution.

(Stat-3): For each client, we compute a C-dimensional vector where the i^{th} dimension represents the number of samples from class i; here, we use CIFAR10 with 100 clients; hence



(a) TrMean (b) FedRecover

Figure 8: The effect of varying heterogeneity levels for FCJ and Dir distributions on the FashionMNIST dataset.

we get 100 10-dimensional vectors. Then, we plotted T-SNE projections of these 100 vectors; we plotted them for both Dir and FCJ distributed client datasets. Figures 7b and 7c show the projections for Dir and FCJ, respectively. For reference, we also show in Figure 7a how the T-SNE projections look like for (1) 100 clients with perfectly i.i.d. datasets and (2) the real-world FEMNIST dataset.

For FCJ distribution, we note that, for bias values greater than 0.2 (Figure 7b center and right), the client datasets form local clusters, i.e., within these clusters, the clients have highly i.i.d. datasets. This is expected because FCJ forms C groups of clients where i^{th} group gets the bias fraction of data from i^{th} class and bias/$(C-1)$ fraction of data from other classes. This data is then randomly assigned to clients within the i^{th} group, which makes these clients' datasets i.i.d. On the other hand, globally, these clusters form potentially non-i.i.d. structures. However, for a bias of 0.1, we observe almost i.i.d. datasets as expected; Figure 7a-left shows a perfectly i.i.d. dataset. To summarize, *although both are globally non-i.i.d., we have shown that the FCJ distribution is locally-i.i.d., while Dir is locally non-i.i.d.*. Next, we study their impact.

2) TrMean is more robust with lower heterogeneity: We demonstrate the impact of FCJ and Dir on TrMean's robustness for FashionMNIST[1] in Figure 8a. In the no-attack setting, FCJ (DIR) shows little difference, with accuracy going from 89% (88%) at 0.1 (0.5) bias to 84% (86%) at 0.9 (0.1) bias. However, the distinction emerges in the attack setting where FCJ is more robust. For the Stat-Opt attack, FCJ (DIR) accuracy decreases from 84% (80%) at 0.1 (0.5) bias to 75% (59%) at 0.9 (0.1) bias. A similar trend is observed for the Dyn-Opt attack. The performance change is rooted in the nature of the distributions (§V-B1). FCJ is locally i.i.d. but globally non-i.i.d., while Dir is non-i.i.d. both locally and globally. Due to Dir's greater heterogeneity, benign updates are more dispersed, making it easier for a malicious update to hide. Consequently, TrMean struggles to detect malicious updates, leading to lower global model performance.

3) FedRecover performs better under lower heterogeneity: We show the impact of FCJ and Dir distributions on FedRecover for the FashionMNIST dataset in Figure 8b. We observe that the performance of FedRecover lowers when we

[1]We observe similar trends for other datasets, but for brevity, we only include FashionMNIST here.

increase the level of heterogeneity and that FCJ performs slightly better than Dir. Specifically, the difference between no-attack accuracies for FCJ at bias 0.3 and 0.7 is 7% and 12.6%, respectively, whereas, for Dir, that difference is slightly higher; 8.2% and 13.3% corresponding to non-i.i.d. parameters of 0.7 and 0.3 respectively[2].

The reason for FedRecover's performance reduction under high heterogeneity is that a higher heterogeneity means that client updates are far apart and drift away from the global model. Since *the update estimation in FedRecover uses knowledge of the current and past global models, the HVP calculation step (§C) incurs significant estimation errors if the local model drifts away from the global model.*

C. Slow-converging Algorithms

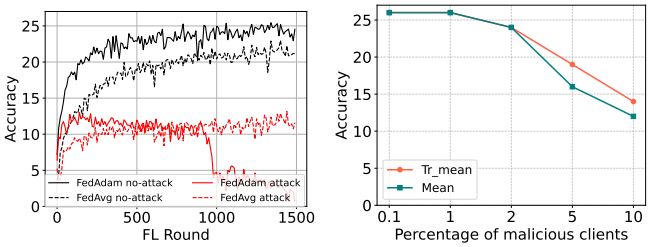
Here, we will study the impact of the choice of the two widely used FL algorithms, FedSGD and FedAvg, on the robustness of TrMean and FedRecover. We combine our analysis of algorithms with the choice of attacks for FLDetector in §V-E2; therefore, we do not discuss it in this section.

1) Fast algorithms make TrMean more robust: In Figure 4, we show the accuracy of FedSGD and FedAvg with TrMean in benign and Stat-Opt attack scenarios. For FedSGD, we align our implementation with recent defenses [18], [28]. In contrast, FedAvg is optimized for greater accuracy with only 5% to 20% of FedSGD's communication (the number of rounds).

Under both benign and adversarial conditions, FedAvg greatly surpasses FedSGD in performance, convergence, and communication for all datasets, as shown in Figure 4. We can conclude here that in adversarial situations, *FedAvg proves more resilient to untargeted poisoning due to its rapid convergence*, giving adversaries minimal time for poisoning.

2) Fast algorithms lower estimation errors in FedRecover: We examine FedRecover's performance under different algorithms, FedSGD and FedAvg. The original study [18] applies the Stat-Opt attack to MNIST with FedSGD over 2000 rounds, reducing accuracy from 96% to 81%. By employing FedAvg with carefully chosen hyperparameters (Appendix D), as shown in Figure 4a, we achieve 98% accuracy in 50 rounds, with the Stat-Opt attack lowering it to only 96%. This allows for perfect recovery, as depicted in Figure 10b, where both MNIST and FashionMNIST (no-attack accuracy: 87%) show perfect recovery under the fast-converging FedAvg. This perfect recovery holds even with variations in the periodic correction periodicity T_c. FedRecover excels in this scenario because *FedAvg's rapid convergence (§II-A) provides a high starting accuracy during FedRecover's* warmup phase. Throughout periodic correction and abnormality fixing phases, *client updates computed over multiple local epochs (in contrast to FedSGD's single local epoch) lead to lower estimation errors, ensuring perfect recovery.*

[2]For FCJ, a higher value of the i.i.d. parameter means higher heterogeneity, but it is the opposite case for Dir.



(a) Trim Attack (b) Varying % malicious

Figure 9: Stackoverflow in practical settings.

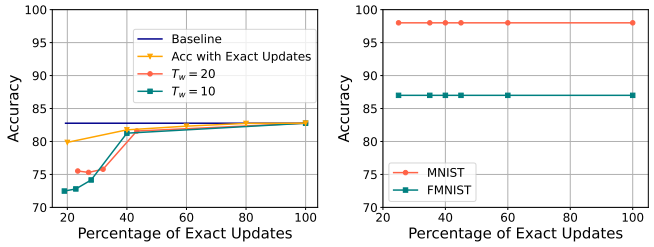
D. Limited FL Settings

To show the effect of *scale* on FL defenses, we first assess the impact of a scale-constrained threat model on TrMean. We then show why FedRecover and FLDetector are incompatible with cross-device settings and discuss the practicality of storage, computation, and communication for FedRecover (or, any mechanism requiring stored historical information).

1) Mean AGR is robust on a large scale: In this section, we showcase the robustness of the mean AGR, typically not robust in cross-silo settings [103], within a large-scale, cross-device environment. *Given the growing popularity of language models [90], [15], although underrepresented in our defense survey (Figure 3a), we leverage the StackOverflow [7] dataset for large-scale evaluation* (setup details in Appendix A). The no-attack baselines in Figure 9a are obtained following the settings in [78]. We can see that with the standard amount of 20% malicious clients, the Stat-Opt attack significantly impacts StackOverflow in cross-device.

The StackOverflow FL setting has about 300,000 clients and 20% amounts to 60,000 malicious clients. Accessing and modifying such a vast number of devices is considered impractical, factoring in operational and financial costs [83]. Figure 9b shows a decline in attack performance as the percentage of malicious clients decreases. Notably, ***with less than 5% malicious clients, the Mean AGR remains unaffected*** by the Stat-Opt attack. *It is important to highlight the difference between our findings and [83]* where the attack impact with Mean AGR is $> 0\%$ even for 0.01% of malicious clients for FEMNIST, CIFAR10, and Purchase, while in our Stack Overflow case, the attack impact is 0% for $< 2\%$ malicious clients. With our results, ***we strongly emphasize using datasets especially designed for FL and evaluating defenses under constraints imposed by scaling up the system in addition to small-scale experimentation.*** We do not dismiss the significance of small-scale experiments. Cross-silo FL is indeed widely used. Instead, we emphasize the critical need for evaluating FL defenses within scaled-up settings.

2) Resource overheads for FedRecover: Here, we first explain the reasons behind our conviction that FedRecover and FLDetector are incompatible in the cross-device setting. Consequently, we do not evaluate these two defenses under the cross-device setting. Nevertheless, in this section, we comment on some of the *practical* aspects of FedRecover, such as computation, communication, and storage costs, to assess the feasibility of scaling up such systems.



(a) FEMNIST (b) MNIST and Fashion

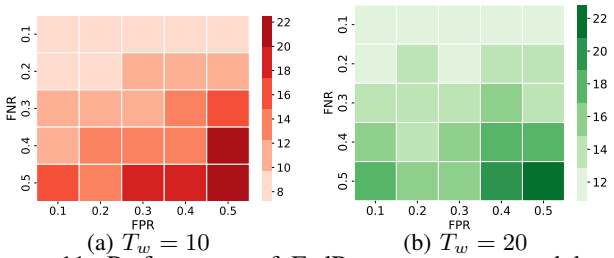
Figure 10: Communication-accuracy tradeoff for FedRecover. Despite the presence of four lines, their overlap is discernible as we can achieve baseline accuracy with exact updates.

Compatibility of Fedrecover and FLDetector with the cross-device FL: FedRecover's reliance on historical information from clients' past updates is hindered in cross-device, where clients participate in a few rounds, limiting available historical updates. Similarly, FLDetector faces challenges in the cross-device setting due to a lack of consistent historical information. Therefore, *we find FedRecover and FLDetector incompatible with the cross-device setting*.

Communication-accuracy tradeoff for FedRecover: In Figure 10a, we show the tradeoff between the recovery accuracy and communication of FedRecover. The recovery accuracy increases as it relies more on exact updates (locally computed updates instead of estimated ones) from the clients, where the *minimum number of exact updates* is $T_w + \frac{T_{total}-T_w-T_f}{T_c} + T_f$. Interestingly, using the same amount of exact updates (effectively the same amount of communication as the server would communicate with clients for the same number of rounds) gives us more accuracy by training from scratch, i.e., not using FedRecover but restarting training with benign clients only. For instance, with only 20% exact updates and $T_w = 20$, FedRecover achieves $\approx 76\%$ accuracy while training from scratch with an equivalent 40 rounds achieves $\approx 80\%$.

In Figure 10b, we can see that with a small percentage of exact updates, FedRecover completely recovers for simpler datasets like MNIST and FashionMNIST with FedAvg and is unaffected by the variation in periodicity T_c. With the same amount of exact updates, we can achieve the same results without FedRecover as well. This is because a fast algorithm gives less time for the adversary to attack and a higher starting point for FedRecover to recover, leading to lower estimation errors (§V-C2).

Based on these observations, we conclude that with fast converging algorithms and proper settings, we might not need to use FedRecover, especially when it comes with an additional computational and storage cost. In our experiments, for the slow baseline that uses FedSGD over 1000 epochs for MNIST, we require $\approx 200GB$ of storage for saving the client model updates every round. By extension, this applies to any defense that requires knowledge of past updates. This cost would significantly increase with the number of clients and by using larger models for more complex tasks.



(a) $T_w = 10$ (b) $T_w = 20$

Figure 11: Performance of FedRecover, represented by the difference between no-attack and post-recovery accuracies, under non-zero FPRs and FNRs caused by adaptive attacks.

E. Naive Attacks

This component is critical in our research, given the prevalence of simple attacks highlighted in §IV. We demonstrate TrMean's vulnerability to powerful model poisoning attacks such as Stat-Opt and Dyn-Opt. We also test our adaptive attack on FLDetector, revealing high rates of *imperfect client detection* and showcasing its impact on FedRecover, which relies on FLDetector to identify malicious clients before recovery.

1) Choice of attacks greatly impact TrMean's performance: We conduct Stat-Opt [28] and Dyn-Opt [84] attacks on FashionMNIST, varying the heterogeneity for both FCJ and Dir distributions. The results are depicted in Figure 8a. Generally, Dyn-Opt is stronger than Stat-Opt, particularly noticeable at higher heterogeneity levels. For instance, at Dir bias level 0.1, the accuracy drops to 51% for Dyn-Opt compared to 59% for Stat-Opt. Dyn-Opt's strength lies in finding optimal perturbations tailored for the dataset at every FL round, making it more potent compared to the static nature of perturbation in Stat-Opt, as discussed in §II-B1.

2) Overcoming FLDetector with our adaptive attack: We introduce a novel attack to assess FLDetector's resilience against *adaptive attacks*. Malicious clients craft updates using our attack and evade detection by FLDetector, resulting in non-zero FNRs and FPRs.

Our attack adds a carefully crafted perturbation vector to client model updates, so they are close enough to the estimated model updates, thereby bypassing FLDetector's detection. We defer the **attack formulation** steps to Appendix B due to space constraints. Following those steps, an adversary is able to craft a malicious update using Equations 2, 3, and 4. The impact of our adaptive attack on the performance of FLDetector is shown in Table III and across all cases (different combinations of dataset, FL algorithm, and percentage of compromised clients), except one, we find that the FNR is non-zero, which means that malicious clients have not been detected, and they continue to be part of the training process. Since the attack is designed to craft malicious updates that are statistically close to the benign ones, it leads to a non-zero FPR as well, which means that many benign clients are falsely detected as malicious and are removed from the training process. Since *we achieve a 0 FPR and a 1 FNR for FEMNIST, this means that the attack never gets detected, no benign or malicious client is removed, and all malicious clients seem benign to FLDetector on the server.* We discuss the impact of non-zero FNR and FPR on further

Table III: Impact of our adaptive attack on FLDetector. Here $\%m$ represents the percentage of malicious clients.

% m	Baseline	MNIST		Fashion		CIFAR10		FEMNIST	
		FPR	*FNR*	*FPR*	*FNR*	*FPR*	*FNR*	*FPR*	*FNR*
5	FedSGD	0.4	1	0.52	0.4	0.21	0	0	1
	FedAvg	0.02	1	0	1	0.13	1	0	1
10	FedSGD	0.39	1	0.56	0.6	0.37	0	0	1
	FedAvg	0.02	1	0.02	1	0.23	0.67	0	1
15	FedSGD	0.48	0.8	0.54	0.3	0.35	0.35	0	1
	FedAvg	0.02	1	0.05	1	0.18	0.67	0	1
20	FedSGD	0.45	1	0.62	0.3	0.35	0.33	0	1
	FedAvg	0.01	1	0.06	1	0.17	0.67	0	1

training and recovery in §V-E3.

3) Imperfect detection leads to lower recovery performance: Our adaptive attack results in imperfect client detection (§V-E2), allowing escaped clients into the recovery process. The escaped clients are denoted by the FNR, while benign clients, incorrectly classified as malicious and subsequently removed from training, are represented by the FPR.

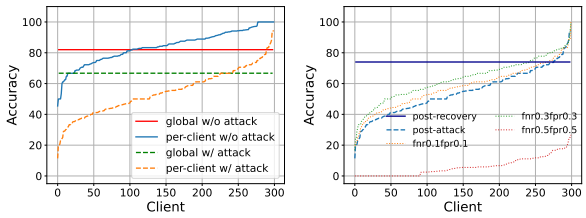
Figure 11 shows that non-zero FNRs and FPRs challenge FedRecover's ability to reach the no-attack accuracy of 82%. For instance, at $FNR = FPR = 0.3$ for $T_w = 10$, the recovery accuracy is only 67%. This is because malicious updates in the recovery process cause higher estimation errors and deviate the recovery model from the benign one. A marginal improvement is observed with more warmup rounds, achieving a post-recovery accuracy of 70% for $T_w = 20$, as increased warmup rounds provide a higher starting point for the model, resulting in lower overall estimation errors.

F. Unfair Metrics

A key feature of FL is heterogeneity (§V-B1), which makes testing and reporting individual client's performance essential. Unfortunately, almost all of the surveyed works only report the global model accuracy (§IV). To show that this is unfair, we show how personalized performance differs from that of the global model using TrMean and FedRecover under benign and adversarial settings.

1) Different clients have different levels of robustness under TrMean: We train on FEMNIST for 200 rounds, achieving a global model accuracy of 82%. Figure 12a displays the accuracy trends, where *global w/o attack* represents the global model accuracy, and *per-client w/o attack* shows individual client test accuracies. *Per-client w/o attack* clusters around the global accuracy, indicating the model learns from the combined data. Due to heterogeneity, we see a lot of variation in the performance of individual clients. In the attack scenario (*global w/ attack* at 66% and *per-client w/ attack*), a similar trend persists, but most clients now fall below the global attack accuracy. *Based on our observations due to heterogeneity across clients, we strongly advocate using per-client metrics in future evaluations.*

2) FedRecover's performance greatly varies on a per-client basis: Similar to the personalized evaluations for TrMean in §V-F1, we apply personalized evaluations to FedRecover for the FEMNIST dataset. After performing FedRecover on the global model with 66% accuracy (§V-F1), we achieve a post-recovery accuracy of 74% (*post-recovery* in Figure 12b). This is illustrated on a per-client basis with the lines



(a) TrMean (b) FedRecover

Figure 12: Personalized evaluations, i.e., per-client accuracy for FEMNIST with TrMean and FedRecover. Note that the accuracy does not increase monotonically with the client number, rather we plot it in an ascending order here since order does not matter when we want to show variation in per-client accuracy.

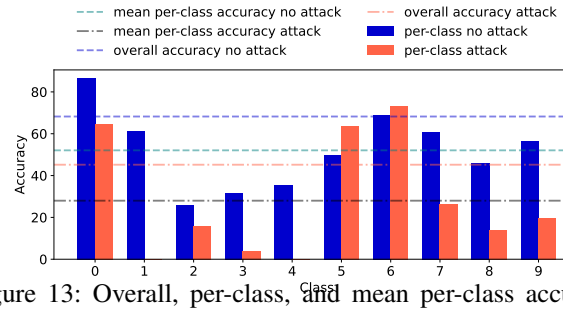


Figure 13: Overall, per-class, and mean per-class accuracies for imbalanced CIFAR10 before and after Stat-Opt attack. Class 0 has the most samples.

$fnr0.1fpr0.1$, $fnr0.3fpr0.3$, $fnr0.5fpr0.5$ in Figure 12b. The post-recovery accuracies for various FNRs and FPRs indicate that most client models fall below the global post-recovery accuracy. At an FNR and FPR of 0.5, all clients lie below both the post-recovery accuracy and the post-attack accuracy, signifying a failure in achieving recovery. This aligns with our findings in §V-E3 that *the presence of malicious clients during recovery leads to higher estimation errors, as mirrored in our per-client evaluations. This consistency underscores the necessity for personalized evaluations, particularly in non-i.i.d. datasets.*

3) Impact of class-imbalance on per-class accuracies: We examine the impact of class imbalance in CIFAR10 on the performance of TrMean under the no-attack and Stat-Opt attack scenarios. Following the setup in Appendix D3, we are able to achieve an overall accuracy of 70.82% without attack and 42.66% with attack using balanced CIFAR10. We define the *overall accuracy* as the total number of correct samples out of the total number of samples in the test dataset and the *mean per-class accuracy* as the mean of all the per-class accuracies in the test dataset. Next, we create an imbalance in the CIFAR10 dataset by removing 90% of the samples of all classes except class zero. This produces 5000 samples of class zero and 500 samples each for other classes in the training set. Figure 13 shows that in the imbalanced setting, the overall accuracy in no-attack is 68%, which is close to its balanced accuracy of 70.82%. However, this accuracy is biased towards class 0, which has the highest per-class accuracy of 86.2%(75.8% in the balanced scenario) since it has ten times

more samples than the rest. The mean per-class accuracy is 52% as it ignores the class imbalance. This indicates that *the mean per-class accuracy is a better metric than overall accuracy in this case, as it shows that the model loses utility with reduced samples.*

The imbalanced Stat-Opt scenario yields an overall accuracy of 48%, surpassing the balanced counterpart at 42.66%. Figure 13 shows that the overall accuracy is significantly influenced by class 0, dropping from 86.2% to 64.3% under attack, compared to a drop from 75.8% to 17.3% in the balanced scenario. The mean per-class accuracy for the imbalanced attack is 27%. This highlights that *dominant classes are less affected by attacks*, influencing the overall accuracy to reflect their performance. *While overall accuracy is crucial, we stress the importance of reporting per-class and mean per-class accuracies, especially in real-world datasets with class imbalances, as neglecting this aspect can lead to misjudgments about a system's robustness.*

VI. Related Work

Previous systemizations: In prior research, various taxonomies in adversarial ML have been developed, extending beyond FL [32], [40], [10], [13], [38], [80] and their main focus is on the attacks. [80] presents a taxonomy specifically for FL defenses, categorizing them based on their occurrence at the client, server, or communication channel. Our contribution extends beyond this by providing a more comprehensive and multidimensional systemization of the existing defenses in the literature. Additionally, we leverage our systemization to identify representative defenses for in-depth pitfall analyses.

Shejwalkar et al. [83] have performed the systemization of FL attacks, highlighting misconceptions about the robustness of FL systems that may arise from overlooking practical considerations about the threat model in deployment scenarios. Their conclusion emphasizes the high robustness of FL with simple and cost-effective defenses in practical threat models. Our work complements this perspective by focusing on defenses. We systematically categorize and re-evaluate representative defenses, uncovering common pitfalls, and providing actionable recommendations to address each.

Pitfalls and guidelines: Arp et al. [6] have extensively studied the pitfalls associated with ML in computer security, identifying and providing recommendations for several challenges in this domain. It is crucial to emphasize the distinctions between our work and theirs. While we may identify similar pitfalls, such as Inappropriate Baseline (§V-C), Inappropriate Performance Measures (§V-F), and Lab-only Evaluation (§V-D), the underlying reasons for these pitfalls and their impact are specific to FL. For example, a pitfall, "inappropriate performance measures," addresses inappropriate performance measures like overall accuracy due to its limited capture of information about false positives and false negatives. In our context (§V-F), we discuss inappropriate performance measures, particularly in relation to fairness, and advocate for the use of personalized metrics, especially when dealing with non-i.i.d. data.

In the space of adversarial ML, Carlini et al. [22] identified numerous pitfalls associated with evaluating defenses against adversarial learning. While there are some commonalities with our work, such as considerations regarding the choice of attacks and testing against adaptive attacks, the primary distinction lies in the fact that our evaluation is specifically conducted in FL. Certain evaluation components are exclusive to FL, such as dealing with client heterogeneity and personalized evaluations.

Improvements over [43]: We take inspiration from [43], which initiated the identification of FL pitfalls and used FedRecover as a case study to showcase the impact of some of these pitfalls. Before building upon their work, we set out to perform a systemization of FL defenses to understand the FL defenses space. This helps us to select representative defenses for pitfall analysis. We have expanded [43]'s analysis by exploring additional pitfalls and studying their impact. Additionally, we broadened the evaluation spectrum by incorporating additional representative defenses like FLDetector and TrMean. Our study extends to a large-scale evaluation of FL with the language modality using StackOverflow [7].

VII. Limitations and Future Work

Performing a thorough exploration and evaluation of all the defenses in Table IV, along with additional ones, goes beyond the scope of a single paper. The defenses chosen for evaluation serve as representatives of the broader defense literature; however, it's acknowledged that alternative evaluations may differ based on different sets of defenses. Also, a newly designed defense based on a novel technique might not fit exactly along the attributes in our systemization. Therefore, our systemization remains adaptable (as detailed in §I), allowing for the incorporation of additional attributes in the future. As highlighted in §I, our aim is to expand the scope of our evaluation across diverse dimensions, encompassing data distributions, data modalities, and the nature of the ML task. While we have addressed image classification and NLP, other modalities, such as multimodal time-series tasks [107], and the emerging paradigm of *vision-language models* [76], remain unexplored. The evaluation of these modalities, along with more complex vision-language models, under traditional threat models could prove intriguing. Such exploration might lead to the development of new attacks and defenses, potentially uncovering novel pitfalls in the process.

VIII. Conclusion

Our study contributes a crucial systemization of FL defenses, offering valuable insights for researchers and practitioners in the selection, combination, and design of defenses. Additionally, we address an often-overlooked aspect in the FL poisoning defense literature— the *experimental setups employed* to assess defense efficacy. After reviewing a variety of defense works, we identify prevalent questionable experimental trends. Through case studies featuring well-known defenses such as TrMean, FLDetector, and FedRecover, we illustrate how the choice of experimental setups can significantly impact robustness claims.

REFERENCES

[1] "How Apple personalizes Siri without hoovering up your data — technologyreview.com," https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/.

[2] "Federated learning: Collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[3] "The stack overflow data," https://www.kaggle.com/datasets/stackoverflow/stackoverflow, 2019.

[4] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, "Byzantine-resilient non-convex stochastic gradient descent," *arXiv preprint arXiv:2012.14368*, 2020.

[5] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "Baffle: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[6] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3971–3988.

[7] T. T. F. Authors., "Tensorflow federated stack overflow dataset," 2019.

[8] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 455–475.

[9] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[10] M. Barreno, B. Nelson, and A. D. Joseph, "The security of machine learning," *Machine Learning*, 2010.

[11] M. Baruch, B. Gilad, and Y. Goldberg, "A Little Is Enough: Circumventing Defenses For Distributed Learning," in *NeurIPS*, 2019.

[12] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019.

[13] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, 2018.

[14] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[15] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[16] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Attestable robustness for secure federated learning," *arXiv preprint arXiv:2107.03311*, 2021.

[17] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[18] X. Cao, J. Jia, Z. Zhang, and N. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2023, pp. 326–343. [Online]. Available: https://arxiv.org/pdf/2210.10936.pdf

[19] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping," in *NDSS*, 2021.

[20] X. Cao, J. Jia, and N. Z. Gong, "Provably Secure Federated Learning against Malicious Clients," in *AAAI*, 2021.

[21] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "Flcert: Provably secure federated learning against poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3691–3705, 2022.

[22] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[23] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer," *arXiv:1912.11279*, 2019.

[24] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017.

[25] J. O. Du Terrail, S.-S. Ayed, E. Cyffers, F. Grimberg, C. He, R. Loeb, P. Mangold, T. Marchand, O. Marfoq, E. Mushtaq *et al.*, "Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings," in *NeurIPS, Datasets and Benchmarks Track*, 2022.

[26] S. Ek, F. Portet, P. Lalanda, and G. Vega, "Evaluation of federated learning aggregation algorithms: application to human activity recognition," in *Adjunct proceedings of the 2020 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2020 ACM international symposium on wearable computers*, 2020, pp. 638–643.

[27] E. M. El Mhamdi, R. Guerraoui, and S. L. A. Rouault, "Distributed momentum for byzantine-resilient stochastic gradient descent," in *9th International Conference on Learning Representations (ICLR)*, no. CONF, 2021.

[28] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *USENIX*, 2020.

[29] I. Feki, S. Ammar, Y. Kessentini, and K. Muhammad, "Federated learning for covid-19 screening from chest x-ray images," *Applied Soft Computing*, vol. 106, p. 107330, 2021.

[30] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," *arXiv:1912.11464*, 2019.

[31] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *RAID*, 2020.

[32] M. Goldblum, D. Tsipras, C. Xie *et al.*, "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses," *arXiv:2012.10544*, 2020.

[33] E. Gorbunov, S. Horváth, P. Richtárik, and G. Gidel, "Variance reduction is an antidote to byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top," *arXiv preprint arXiv:2206.00529*, 2022.

[34] H. Guo, H. Wang, T. Song, Y. Hua, Z. Lv, X. Jin, Z. Xue, R. Ma, and H. Guan, "Siren: Byzantine-robust federated learning via proactive alarming," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 47–60.

[35] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik, "Lower bounds and optimal algorithms for personalized federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2304–2315, 2020.

[36] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *arXiv preprint arXiv:2002.05516*, 2020.

[37] T. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, "Fairness without demographics in repeated loss minimization," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1929–1938.

[38] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *AISec*, 2011.

[39] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," *arXiv preprint arXiv:2207.01982*, 2022.

[40] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Security & Privacy*, 2020.

[41] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977*, 2019.

[42] S. P. Karimireddy, L. He, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," *arXiv preprint arXiv:2006.09365*, 2020.

[43] M. A. Khan, V. Shejwalkar, A. Houmansadr, and F. M. Anwar, "On the pitfalls of security evaluation of robust federated learning," in *2023 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2023, pp. 57–68.

[44] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *NIPS Workshop on Private Multi-Party ML*, 2016.

[45] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[46] H. Ku, W. Susilo, Y. Zhang, W. Liu, and M. Zhang, "Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption," *Computer Standards & Interfaces*, vol. 80, p. 103583, 2022.

[47] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[48] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI*, 2019.

[49] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 713–10 722.

[50] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[51] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[52] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *ICML*, 2021.

[53] Z. Li, L. Liu, J. Zhang, and J. Liu, "Byzantine-robust federated learning through spatial-temporal analysis of local model updates," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 372–379.

[54] F. Lin, W. Li, and Q. Ling, "Stochastic alternating direction method of multipliers for byzantine-robust distributed learning," *arXiv preprint arXiv:2106.06891*, 2021.

[55] Y. Liu, R. Zhao, J. Kang, A. Yassine, D. Niyato, and J. Peng, "Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 3, pp. 1–22, 2021.

[56] X. Ma, Q. Jiang, M. Shojafar, M. Alazab, S. Kumar, and S. Kumari, "Disbezant: secure and robust federated learning against byzantine attack in iot-enabled mts," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[57] S. Mahloujifar, M. Mahmoody, and A. Mohammed, "Universal multi-party poisoning attacks," in *ICML*, 2019.

[58] R. A. Mallah, D. Lopez, G. B. Marfo, and B. Farooq, "Untargeted poisoning attack detection in federated learning via behavior attestation," *arXiv preprint arXiv:2101.10904*, 2021.

[59] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[60] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *ICML*, 2018.

[61] T. Minka, "Estimating a Dirichlet distribution," 2000.

[62] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, "Frl: Federated rank learning," *arXiv preprint arXiv:2110.04350*, 2021.

[63] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *AISec*, 2017.

[64] L. Nagalapatti and R. Narayanam, "Game of gradients: Mitigating irrelevant clients in federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9046–9054.

[65] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint arXiv:2009.03561*, 2020.

[66] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–37, 2022.

[67] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[68] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "Clusterfl: a similarity-aware federated learning system for human activity recognition," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 54–66.

[69] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against backdoors in federated learning with robust learning rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9268–9276.

[70] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, "Justinian's gaavernor: Robust distributed learning with gradient aggregation agent," in *Proceedings of the 29th USENIX Conference on Security Symposium*, 2020, pp. 1641–1658.

[71] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Advances in neural information processing systems*, vol. 34, pp. 840–851, 2021.

[72] M. Paulik, M. Seigel, H. Mason *et al.*, "Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications," *arXiv:2102.08503*, 2021.

[73] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv:1912.13445*, 2019.

[74] S. Praneeth Karimireddy, L. He, and M. Jaggi, "Learning from History for Byzantine Robust Optimization," *arXiv e-prints*, pp. arXiv–2012, 2020.

[75] A. Qayyum, K. Ahmad, M. A. Ahsan, A. Al-Fuqaha, and J. Qadir, "Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 172–184, 2022.

[76] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[77] P. Ranjan, A. Gupta, F. Coro, and S. K. Das, "Securing federated learning against overwhelming collusive attackers," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 1448–1453.

[78] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive Federated Optimization," in *ICLR*, 2020.

[79] J. H. Ro, A. T. Suresh, and K. Wu, "FedJAX: Federated learning simulation with JAX," *arXiv preprint arXiv:2108.02117*, 2021.

[80] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, "Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges," *Information Fusion*, vol. 90, pp. 148–173, 2023.

[81] A. Roy Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "Eiffel: Ensuring integrity for federated learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2535–2549.

[82] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[83] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2022, pp. 1117–1134. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00065

[84] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning," in *NDSS*, 2021.

[85] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending againsts poisoning attacks in collaborative deep learning systems," *2016 Annual Computer Security Applications Conference*, 2016.

[86] K. Sozinov, V. Vlassov, and S. Girdzijauskas, "Human activity recognition using federated learning," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. IEEE, 2018, pp. 1103–1111.

[87] J. Sun, A. Li, L. Duan, S. Alam, X. Deng, X. Guo, H. Wang, M. Gorlatova, M. Zhang, H. Li *et al.*, "Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices," in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2022, pp. 106–119.

[88] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *NeurIPS FL Workshop*, 2019.

[89] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[90] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[91] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.

[92] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Flare: defending federated learning against model poisoning attacks via latent space representations," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 946–958.

[93] "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.

[94] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv:2011.01767*, 2020.

[95] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.
[96] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[97] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "CRFL: Certifiably Robust Federated Learning against Backdoor Attacks," in *ICML*, 2021.
[98] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv:1802.10116*, 2018.
[99] C. Xie, S. Koyejo, and I. Gupta, "Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation," *arXiv:1903.03936*, 2019.
[100] Y. Xie, W. Zhang, R. Pi, F. Wu, Q. Chen, X. Xie, and S. Kim, "Robust federated learning against both data heterogeneity and poisoning attack via aggregation optimization," *arXiv preprint*, 2022.
[101] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, "Tdfl: Truth discovery based byzantine robust federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4835–4848, 2022.
[102] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Signguard: Byzantine-robust federated learning through collaborative malicious gradient filtering," *arXiv preprint arXiv:2109.05872*, 2021.
[103] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *ICML*, 2018.
[104] S. Zawad, A. Ali, P.-Y. Chen, A. Anwar, Y. Zhou, N. Baracaldo, Y. Tian, and F. Yan, "Curse or redemption? how data heterogeneity affects the robustness of federated learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 10 807–10 814.
[105] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma *et al.*, "Flip: A provable defense framework for backdoor mitigation in federated learning," *arXiv preprint arXiv:2210.12873*, 2022.
[106] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.
[107] Y. Zhao, P. Barnaghi, and H. Haddadi, "Multimodal federated learning on iot data," in *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2022, pp. 43–54.

APPENDIX

A. Our methodology to classify 50 defenses

Table IV shows our survey of the 50 defense works and our classification methodology along the six dimensions: Datasets, Attacks, Data Distribution, FL-Algorithm, FL Type, and Evaluation Type.

B. Our adaptive attack

FLDetector computes the estimated update (§C) for client k as:

$$\hat{\nabla}_t^k = \nabla_{t-1}^k + \hat{H}^t \cdot (\theta_t - \theta_{t-1}) \tag{1}$$

In this attack, we introduce a perturbation vector, $\mathcal{P}$, that modifies the updates sent from malicious clients. A malicious client computes its update so that its final update is computed as the sum of its previous update, the HVP or $Hessian\ Vector\ Product$ $(\hat{H}^t \cdot (\theta_t - \theta_{t-1}))$, and the perturbation vector. This can be written as:

$$\hat{\nabla}_t^k = \nabla_{t-1}^k + \hat{H}^t \cdot (\theta_t - \theta_{t-1}) + \mathcal{P} \tag{2}$$

The server estimates an update by adding the HVP to the last round's exact update and compares it with the actual update in that round (§C). The malicious update, therefore, deviates from the estimated update by $\mathcal{P}$. We proceed to detail the calculation of this perturbation vector $\mathcal{P}$. To calculate $\mathcal{P}$, we first calculate a *good distance range*, $\mathcal{R}$, that is a safe perturbation distance for the perturbation vector by taking the norm between the old and new client updates. The good distance range, $\mathcal{R}_t^k$, for client k, at round t is given by:

$$\mathcal{R}_t^k = ||\hat{\nabla}_t^k - \nabla_t^k|| \tag{3}$$

Here, $\hat{\nabla}_t^k$ is the estimated update for client k at round t, and ∇_t^k is the actual update for client k at round t. The deviation, $\mathcal{D}$, for the perturbation vector is calculated by following deviation strategies in [84]. It can either be *unit vector*, *sign*, or *std*. Finally, $\mathcal{P}$ is computed by taking the average of all the *good distance ranges*, $\mathcal{R}$, and directing it in the direction of the deviation, $\mathcal{D}$:

$$\mathcal{P} = \frac{\mathcal{D}}{||\mathcal{D}||} \cdot \frac{1}{N} \sum_{k=1}^{N} \mathcal{R}_t^k \tag{4}$$

C. Descriptions of our chosen defenses

Trimmed Mean (*TrMean*) [103] is a foundational defense used in advanced defenses [18], [106], [84]. It sorts each input dimension j of the client updates, discards the m largest and smallest values (where m indicates compromised clients), and averages the rest.

FLDetector [106] is designed to *detect* and eliminate malicious clients, ensuring a byzantine-robust FL system obtains a precise global model. FLDetector operates on the principle that malicious updates, tainted by adversaries, differ statistically from benign ones.

For discerning these updates, the server estimates a global model update for client k at round t using the L-BFGS algorithm: $(\hat{\nabla}_t^k = \nabla_{t-1}^k + \hat{H}^t \cdot (\theta_t - \theta_{t-1}))$. Here, ∇_{t-1}^k. The server retains past N global model differences $(\Delta\theta_t)$ and updates differences $(\Delta\nabla_{t-1})$ to compute the *HVP (Hessian Vector Product)* with *L-BFGS*. It then gauges a client's *suspicious score* by comparing actual and predicted updates through their Euclidean distance. With scores from the last N rounds, clients are clustered via Gap statistics [89] and K-means. The group with higher average scores is deemed malicious. Upon detecting a rogue client, the server ceases training, removes the offender, and restarts training to achieve enhanced accuracy.

FedRecover [18] aims to *recover* an FL global model compromised by a poisoning attack. In the *original training phase*, for each round t, FedRecover saves ∇_t^k from client k and global models θ_g^t. This data is used as *historical information* in the *recovery phase*, which consists of the following stages. In the *warmup phase*, the server requests clients' *exact updates* for the initial T_w rounds. In the *estimation phase*, the server *estimates* client updates each round, with $\hat{\nabla}_k^t$ representing client k's estimated update at round t. This estimate is derived using the L-BFGS algorithm [67] based on the original global model, client update, and recovered global model. The model's estimated update is defined as $\hat{\nabla}_t^k = \overline{\nabla}_k^t + \tilde{H}_k^t(\hat{\theta}_g^t - \overline{\theta}_g^t)$, where the latter term is the HVP(Hessian Vector Product). Every

Table IV: Classification of 50 defense works across 6 dimensions of evaluation setup.

Work	Datasets	Attacks	Data Distribution	FL Algorithm	FL Type	Evaluation
FLDetector [106]	FA,FE,C10	Stat-Opt	Fang, Natural	FedSGD	CS	Global
FedRecover [18]	M,FA,P,H	Stat-Opt	Fang	FedSGD	CS	Global
Machine Learning with Adversaries [14]	M, spambase	RGA	IID	FedSGD	CS	Global
FLTrust [19]	M,CHM,C10,H	Krum, Stat-Opt, LF	Fang	FedAvg	CS	Global
Byzantine-Robust Distributed Learning [103]	M	RGA	IID	FedSGD	CS	Global
Provably Secure Federated Learning against [20]	M	Not applicable	Fang	FedAvg		Global
Learning to Detect Malicious Clients for [51]	M, FE, S140	SF, AN, BD	Natural, McMahan	FedAvg	CD	Global
Robust Federated Learning [100]	M,FE,C10/100,N20	LF, LIE, Fang	Dirichlet, Natural	FedAvg	CD	Global
The Hidden Vulnerability of Distributed [60]	M, C10	Specific attack	IID	FedSGD	CS/CD	Global
Sageflow [71]	M, FA, C10	SF, LF	McMahan	FedAvg	CS	Global
Mitigating Irrelevant Clients in FL [64]	M	LF	McMahan	FedAvg	CS	Global
Cronus [23]	M, C10, P, Svhn	{LF, LIE}	IID	FedAvg	CS	Global
Can You Really Backdoor Federated Learning? [88]	FE	BD	Natural	FedAvg	CS/CD	Global
Generalized Byzantine-tolerant SGD [98]	M, C10	BF, LF, LIE	IID	FedSGD	CS	Global
The Limitations of Federated Learning [31]	M, VGG, KDD, A	LF, BD	Each class to a client	Both	CS	Global
Auror [85]	M	Targeted-LF	IID	FedSGD	CS	Global
Robust Aggregation for Federated Learning [73]	FE,S140,S	Specific attacks, RGA	Natural	FedAvg	CS/CD	Global
CRFL [97]	M, FE	BD	IID	FedAvg	CS	Global
FLIP [105]	M, FA, C10	BD	Dirichlet	FedAvg	CS	Global
RoFL [16]	FE, C10	BD	Natural, Dirichlet	FedAvg	CD	Global
Securing FL against Overwhelming [77]	M, FA	LF, BD	Dirichlet	FedAvg	CS	Global
Defending against the Label-flipping Attack [39]	M, C10	LF,	IID, Dirichlet	FedAvg	CS	Global
FRL [62]	M, FE, C10	Fang, Dyn-Opt	Dirichlet, Natural	FedAvg	CD	Global
CONTRA [8]	M, C10, Loan	LF, BD	Dirichlet	FedAvg	CS	Global
EIFFeL [81]	M, FA, FE,C10	LIE, RGA, SF, Dyn-Opt	IID, Natural	FedAvg	CS/CD	Global
Local and central DP for robustness [65]	E, C10, s140, Reddit	BD	McMahan	FedAvg	CS/CD	Global
Signguard [102]	M, FA, C10, AGnews	LIE, RGA, SF, Dyn-Opt	IID	FedSGD	CS	Global
DisBezant [56]	{M, FA, C10}	RGA	{Fang}	FedAvg		Global
Learning from History for Byzantine [74]	M, C10	BF, LF, LIE	Exponential	FedSGD	CS	Global
Byzantine-robust learning on heterogeneous [42]	M	BF, LF, LIE, IPM, Mimic	{McMahan}	FedSGD	CS	Global
Byzantine-Resilient Non-Convex Stochastic [4]	C10, C100	SF, LF, LIE, Delayed-grad	IID	FedSGD	CS	Global
Byzantine-robust Federated Learning [53]	M, FA, C10,Spambase	LF, IPM, LIE, Uniform, arbitrary	McMahan	FedAvg	CS/CD	Global
Stochastic alternating direction method of [54]	M, Covertype	RGA, SF, LF	IID	FedSGD	CS	Global
Variance reduction is an antidote to byzantines [33]	LIBSVM	LF, BF, LIE, IPM	{IID}	FedSGD	CS	Global
On the byzantine robustness of clustered FL [82]	M, FA, C10	RGA, LF, Uniform noise	IID	FedSGD	CS	Global
RSA [48]	M	SF	IID	FedSGD	CS	Global
Federated variance-reduced [95]	ijcnn1, covtype	RGA, SF, Zero-grad	IID	FedSGD	CS	Global
Abnormal client behavior detection in [50]	FE	SF, RGA, Grad ascent	Natural	FedAvg	CD	Global
Distributed Momentum for Byzantine [27]	M, FA, C10/100	LIE, IPM	IID	FedSGD	CS	Global
Attack-resistant FL with residual-based [30]	M, C10, Amazon, Loan	LF, BD	Dirichlet, Natural	FedAvg	CS	Global
Towards communication-efficient [55]	M	LF	IID	FedSGD	CS	Global
Justinian's GAAvernor [70]	M, C10, Yelp, Health	RGA	IID	FedSGD	CS	Global
Untargeted poisoning attack detection [58]	M, C10, MTL Trajet	BD	IID	FedSGD		Global
TDFL [101]	M, FA, C10	LF, RGA, Krum, Stat-Opt, BD	McMahan	FedAvg	CD	Global
Siren [34]	FA, C10	SF, LF, bhagoji	Fang	FedAvg	CS	Global
FLARE [92]	FA, C10, Kather	Krum, Stat-Opt	IID	FedAvg	CS	Global
Analyzing Federated Learning Through [12]	FA, UCI Census	Specific attack	IID	FedAvg		Global
BaFFLe [5]	C10, FE	BD	Dirichlet	FedAvg		Global
Defending against backdoors in FL [69]	FA, FE	BD	Both	FedAvg		Global
Ditto [52]	FA, FE, CelebA	LF, RGA, BD	Natural, McMahan	FedAvg	CS	Personalized

T_c rounds, to ensure the estimated global model $\hat{\theta}_g^t$ aligns closely with the accurate model, the server initiates a *periodic correction* by requesting *exact updates*. If any component of a client's estimated update surpasses the *abnormality threshold* τ, that client is prompted for an exact update. In the final *fine-tuning phase*, spanning T_f rounds, clients are asked to provide their exact updates,∇_k^t, to refine the global model by eliminating potential estimation errors.

D. Experimental setup

Due to space constraints, we provide a detailed experimental setup here.

1) FEMNIST [17], [24]: FEMNIST 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 their own handwritten digits or letters. We use 300 randomly selected clients with their original data in a cross-silo fashion, as FedRecover uses the cross-silo setting in its implementation. We use the CNN used by [18] and use the Xavier weight initialization.

Hyperparameters for re-eval: For FEMNIST, we run over 200 epochs with 300 clients. In the attack setting, 60 clients

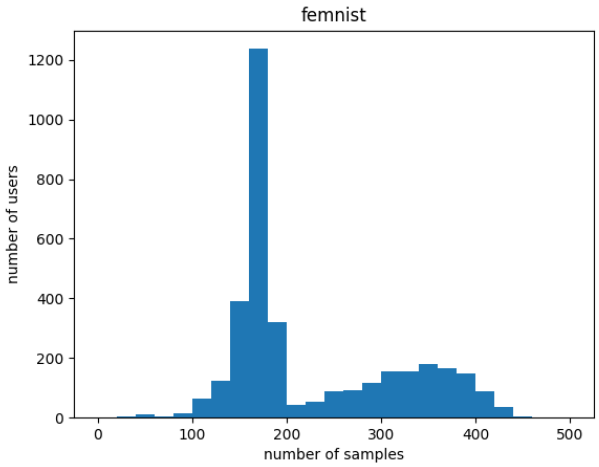


Figure 14: FEMNIST histogram from leaf.cmu.edu

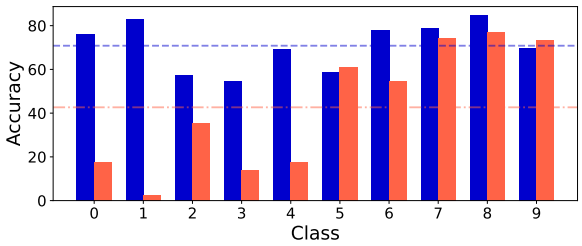


Figure 15: Overall and per-class accuracies for CIFAR10 before and after Stat-Opt attack.

are malicious. The results in Figure 5 use $T_w = 10$, and $T_c = 10$. The FL algorithm used here is FedAVG with a local learning rate of 0.05 and a global learning rate of 1. We keep the batch size to 32. The number of local epochs is kept at 1. For Figure 11a, we consider the possibility of benign clients being misclassified as malicious or malicious clients being misclassified as benign, so we vary the false negative and false positive rates between 0.1 and 0.5.

2) CIFAR10 [45]: CIFAR10 is a 10-class classification task with 60,000 total RGB images, each of size 32×32. We divide all the data among 100 clients using either Dirichlet [78] or *FCJ* [28] distributions, which are the two most popular synthetic strategies to generate the FL dataset. We use a Resnet20 model with the CIFAR dataset.

Hyperparameters for re-eval: We run over 100 epochs with 100 clients. In the attack setting, 20 clients are malicious. The FL algorithm used here is FedAVG, with a local learning rate of 0.01 and a global learning rate of 1. We keep the batch size to 16. The number of local epochs is kept at 2. The results in Figure 5 use $T_w = 10$, and $T_c = 5$ and the fang distribution. Contrary to the rest of the datasets used, we use $T_c = 5$ because CIFAR10 was a much more challenging learning task.

3) Imbalanced CIFAR10: To set up our baseline, we train an Alexnet model with 100 clients over 100 epochs. We use the standard CIFAR10 dataset that has 50000 training samples and 10000 test samples. Figure 15 shows that we achieve an overall accuracy of 70.82% without attack and 42.66% with attack. It also shows the respective per-class accuracies. *We define the mean accuracy as the mean of all the per-class accuracies*. In this case, since the test dataset is perfectly balanced, i.e., each class has the same number of test samples, the overall accuracy and the mean accuracy are the same.

4) MNIST [47]: MNIST is a 10-class digit image classification dataset, which contains 70,000 grayscale images of size 28×28. We consider 100 FL clients and divide all data using Dirichlet or FCJ distributions. We use the same CNN as the FEMNIST dataset.

Hyperparameters for re-eval: For MNIST, we run over 2000 epochs with 100 clients, a learning rate of 0.03, and a batch size of 32. In the attack setting, 20 clients are malicious. We set $T_w = 20$, and $T_c = 10$. The FL algorithm used here is FedSGD. The results reported in Figure 5 use the FCJ distribution.

5) Fashion-MNIST [96]: Fashion-MNIST is a 10-class image classification dataset with grayscale images of clothing of size 28×28. It contains 70,000 total images. We consider 100 FL clients and divide all 70,000 images using Dirichlet or FCJ distributions. For CIFAR10, MNIST, and FashionMNIST, we divide each client's data in train/test/validation splits in the ratio of $10 : 1 : 1$. We combine clients' validation data and use it for validation and hyperparameter tuning and report accuracy on test data. We use the same CNN as the FEMNIST dataset.

Hyperparameters for re-eval: We run over 2000 epochs with 100 clients, a learning rate of 3×10^{-3} [3], and a batch size of 32. In the attack setting, 20 clients are malicious. We set $T_w = 20$, and $T_c = 10$. The FL algorithm used here is FedSGD. The results reported in Figure 5 use the FCJ distribution.

6) StackOverflow [7]: StackOverflow is a language-modeling dataset that is used for tag prediction and next-word prediction. It consists of 342,477 users who are used as clients, and the training data consists of 135,818,730 examples. We use an RNN with a 96-dimensional embedding and a 10000-word vocabulary. The complete network consists of an input layer followed by an embedding layer, an LSTM layer, and two dense layers. We use the cross-device setting to obtain the baseline in [78] by using the fedjax [79] framework. The FL algorithm is $FedAdam$, and the training consists of 1500 rounds with 50 clients chosen every round with one local epoch. We keep the batch size to 16, the client optimizer as SGD with a learning rate of 10^{-3}, and Adam as the server optimizer with a learning rate of 10^{-2}.

Table V: Abbreviations and full-forms of datasets in Table IV.

M	MNIST
FA	FashionMNIST
FE	FEMNIST
C10	CIFAR10
C100	CIFAR100
P	Purchase
H	HAR
S140	Sentiment140
SVHN	Street-view House Numbers
VGG	VGGFace
KDD	KDDCup
N20	News 20
A	Amazon
S	Shakespeare

E. Full names for datasets and attacks

Table VI: Abbreviations and full-forms of attacks in Table IV.

LF	Label Flip
IPM	Inner Product Manipulation
SF	Sign Flip
BF	Bit Flip
LIE	Little is Enough
AN	Additive Noise
BD	Backdoor [9]
RGA	Random Gaussian Attack

[3]We could not achieve the same accuracy reported in [18] using their reported 3×10^{-4} learning rate, hence we use 3×10^{-3}.