# Towards Robust and Privacy-Preserving Federated Learning: Reinforcement-Guided Unlearning and Multi-Attack Defense Mechanisms

*A thesis submitted in fulfilment of the requirements*
*for the degree of*

Doctor of Philosophy
*in*
Software Engineering

*by*

**Kun Gao**

*to*

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

September 2025

# CERTIFICATE OF ORIGINAL AUTHORSHIP

I, **Kun Gao**, declare that this thesis is submitted in fulfilment of the requirements for the award of **Doctor of Philosophy in Software Engineering**, in the **School of Computer Science Faculty of Engineering and Information Technology** at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship
doi.org/10.82133/C42F-K220.

Signed: Production Note: Signature removed prior to publication.

Date: 13/1/2026

# ABSTRACT

Federated Learning (FL) has become an attractive new trend for distributed model training where the data is decentralized and is not required to be uploaded to a central server itself, alleviating the direct privacy issues. However, this decentralized paradigm also gives rise to new security risks such as gradient inversion, poisoning, and inference threats. Meanwhile, with rigorous regulatory requirements and the changing nature of privacy, federated unlearning is a crucial building block, bringing about new issues related to security and efficiency. This thesis explores secure and private defenses of FL against various threat models by combining reinforcement learning-informed defense strategies with statistical machine unlearning.

The contributions of this paper are as follows:

It proposes a statistical-unlearning-based defense through gradient inversion attacks while also achieving the balance between high model utility and communication efficiency.

It reveals potential vulnerabilities in federated unlearning by revealing invisible attacks (i.e., camouflaged poisoning attacks that can still be effective after unlearning operations) and theoretically analyzes the effects of the attacks for the long term.

It introduces the first reinforcement learning-based federated unlearning mechanism that can dynamically balance the client contribution, the privacy cost, and the computational efficiency, leading to enhanced robustness to both inference and poisoning.

It also introduces a data importance-aware reinforcement learning defense that adaptively victims protection strategies at a sample level and achieves multi-attack robustness against backdoor, model stealing, and membership inference attacks.

Both theoretically and empirically, we verify that the proposed approaches achieve better trade-offs between robustness, accuracy, and efficiency over different datasets and adversarial settings than state-of-the-art counterparts. Taken together, this thesis contributes to the understanding of attack defense dynamics on FL and proposes reinforcement-guided unlearning as a principled basis for adaptive, secure, and privacy-compliant decentralized learning.

# Dedication

*To myself . . .*

# ACKNOWLEDGMENTS

I acknowledge Professor Tianqing Zhu, Associate Professor Angela Huo, Associate Professor Bo Liu Professor Wanlei Zhou, Associate Professor Dayong Ye, for their valuable suggestions..

# List of Publications

**Related to the Thesis :**

1. **Kun Gao**, Tianqing Zhu, Dayong Ye, Wanlei Zhou. Defending against gradient inversion attacks in federated learning via statistical machine unlearning. Knowledge-Based Systems Volume 299, 5 September 2024, 111983.

2. **Kun Gao**, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou. Hidden Threats in Federated Unlearning: Camouflaged Poisoning Attacks and Their Unlearning Consequences. IEEE Transactions on Dependable and Secure Computing. Under review.

3. **Kun Gao**, Tianqing Zhu, Dayong Ye, Longxiang Gao, Wanlei Zhou. Federated unlearning with Reinforcement Learning: Adaptive Privacy Preservation for Clients. Journal of Information Security and Applications. Under review

4. **Kun Gao**, Tianqing Zhu, Dayong Ye, Heng Xu, Wanlei Zhou. Learning to Defend What Matters: Data Importance-Aware Reinforcement Learning for Multi-Attack Robustness. Under review.

**Others :**

4. Dayong Ye, Tianqing Zhu, **Kun Gao**, Wanlei Zhou. Defending against label-only attacks via meta-reinforcement learning. IEEE Transactions on Information Forensics and Security 19, 3295-3308.

5. Dayong Ye, Tianqing Zhu, **Kun Gao**, Congcong Zhu, Wanlei Zhou. Cooperating or Kicking Out: Defending against Poisoning Attacks in Federated Learning via the Evolution of Cooperation. IEEE Transactions on Dependable and Secure Computing.

6. Dayong Ye, Tianqing Zhu, Congcong Zhu, Derui Wang, **Kun Gao**, Zewei Shi, Sheng Shen, Wanlei Zhou, Minhui Xue. Reinforcement Unlearning. Accepted by NDSS 2025.

# Table of Contents

# List of Figures

**Figure** **Page**

# List of Tables

CHAPTER 1

# INTRODUCTION

This chapter introduces the motivation for studying security, privacy, and unlearning in federated learning. It outlines the research challenges, objectives, and contributions, and provides an overview of how the thesis is structured.

Artificial Intelligence (AI) and Deep Learning (DL) have rapidly achieved remarkable success in many applications (e.g., health, finance, transport, and mobile services). However, the growing adoption of data-driven models also gives rise to important issues of data privacy, security, and robustness.

In particularly, for centralized learning systems, centralizing raw data at the center has the effect of creating a single point of failure, and in the case of personal or sensitive data, it carries significant risks of privacy disasters. To overcome these limitations, Federated Learning (FL) has been proposed as a promising paradigm to allow for multiple distributed participants (clients) to aggregate their models to train a global model without exchanging raw data. Instead, clients communicate model updates with a centralized server only, thereby maintaining data locality and providing a primary level of privacy.

While FL holds great potential, its decentralized nature introduces new attack surfaces and technical challenges. Research has demonstrated that adversaries can exploit model updates to reconstruct private data, inject malicious behaviors, or infer sensitive information about training samples. Moreover, the growing demand for compliance with privacy regulations such as GDPR [100] and CCPA [9] has emphasized the necessity of machine unlearning, i.e., the ability to remove the influence of specific users or data from a trained model upon request. Unfortunately, both FL and unlearning remain vulnerable

to sophisticated adversarial strategies, motivating the need for robust and adaptive defense mechanisms.

Early efforts to address these challenges primarily focused on gradient inversion attacks, where adversaries reconstruct private inputs from shared updates. It has been demonstrated that a single gradient snapshot can leak sensitive training data with high quality [136] [118]. The technical obstacle in this is that gradients contain rich information regarding original data distributions. Current defenses like differential privacy mix calibrated noise, but always come with a loss of model utility.

However, beyond inference attacks, the integrity of federated models is also threatened by poisoning attacks. Classical Byzantine-robust aggregation methods [6] filter anomalous updates but are insufficient against carefully crafted manipulations. The key technical finding is that poisoning can be camouflaged during the training phase-through gradient matching and label manipulation-to appear benign under standard aggregation rules. However, once unlearning is applied, the removal of benign gradients disproportionately exposes the malicious component, thereby amplifying the attack's impact. This work connects directly to the previous study by showing that while unlearning can be a defense against inversion, it simultaneously creates new attack surfaces when adversaries embed malicious signals that exploit the very process of forgetting.

The recognition that both defenses and attacks evolve dynamically motivates the introduction of adaptive, learning-based decision mechanisms. Inspired by reinforcement learning (RL), we developed in Federated Unlearning with Reinforcement Learning: Adaptive Privacy Preservation for Clients a framework where unlearning is not a fixed operation but a policy-driven decision. Technically, the defense agent observes a state vector that encodes client contribution, privacy sensitivity, and computational cost, and learns through Q-learning to choose between partial unlearning, full unlearning, or no unlearning. This extends the previous two works in two ways: first, it systematizes unlearning into a decision-making process, rather than a rule-based mechanism; and second, it directly addresses the trade-offs exposed in prior studies by treating privacy preservation, robustness, and efficiency as joint optimization objectives.

However, the RL framework in its initial form still assumed uniform vulnerability across clients and data samples. In practice, different samples influence the model unequally: some shape decision boundaries, some are frequently queried, and others are inherently more prone to overfitting. Ignoring this heterogeneity leads to suboptimal protection policies. To bridge this technical gap, our most recent work Learning to Defend What Matters: Data Importance-Aware Reinforcement Learning for Multi-Attack

Robustness incorporates per-sample importance estimation into the RL state space. This enriches the defense agent's observation with signals reflecting sample sensitivity and contribution. By doing so, the defense agent can learn fine-grained action allocation such as applying stronger protection to high-importance samples while maintaining efficiency on low-importance ones. Moreover, this framework generalizes to multiple adversarial settings, including backdoor poisoning, model stealing, and membership inference. Technically, it extends the reinforcement-guided unlearning paradigm by embedding importance-aware profiling, thus achieving robustness across diverse attack vectors.

Taken together, these four studies form a technically coherent trajectory: starting with the development of statistical unlearning to neutralize gradient inversion, moving to the discovery of camouflaged poisoning that exploits the unlearning process itself, advancing to reinforcement learning for adaptive unlearning decisions, and culminating in an importance-aware RL framework that achieves multi-attack robustness with balanced trade-offs. Each step is not isolated but technically builds upon the limitations revealed in the previous one, gradually evolving toward a unified vision of reinforcement-guided, privacy-preserving, and robust FL.

## 1.1 Research Objectives

Building on the above challenges and the identified research gaps, this thesis is driven by the following objectives:

- Objective 1. Mitigate privacy leakage from gradient inversion attacks by leveraging statistical unlearning principles that selectively eliminate sensitive information while preserving efficiency and accuracy.

- Objective 2. Reveal and understand the hidden risks of federated unlearning, particularly camouflaged poisoning attacks, and provide theoretical and empirical evidence of their long-term consequences.

- Objective 3. Develop reinforcement learning-guided federated unlearning frameworks that transform unlearning from a static mechanism into an adaptive decision-making process capable of balancing privacy preservation, robustness, and efficiency.

- Objective 4. Incorporate data importance into adaptive defense by profiling sample contributions and vulnerabilities, enabling reinforcement learning agents to allocate protections dynamically and achieve robustness against multiple concurrent attacks, including backdoor, model stealing, and membership inference.

These objectives form a coherent trajectory that starts with understanding fundamental vulnerabilities, proceeds to uncover emergent risks, and culminates in designing adaptive, generalizable defense strategies for federated systems.

## 1.2 Research Contributions

The main contributions of this thesis can be summarized as follows:

- Contribution 1: Statistical Machine Unlearning for Gradient Inversion Defense

  This thesis introduces a statistical machine unlearning framework that disrupts gradient inversion attacks by training on aggregated statistical queries instead of raw client updates. When combined with knowledge distillation, the proposed method improves both communication efficiency and model performance, and it outperforms noise-based baselines.

- Contribution 2: Discovery of Camouflaged Poisoning in Federated Unlearning

  This thesis reveals a novel class of adversarial strategies in which poisoned updates remain undetectable during training but are activated after unlearning. Theoretical analysis and extensive experiments are conducted to demonstrate that unlearning may inadvertently amplify adversarial influence.

- Contribution 3: Reinforcement Learning-Guided Unlearning Framework

  A Deep Q-Network (DQN)-based defense agent is developed in this thesis to adaptively decide among full unlearning, partial unlearning, or no unlearning. In this framework, state features such as client contribution, privacy cost, and computational overhead are incorporated, thereby enabling adaptive trade-offs among robustness, efficiency, and privacy.

- Contribution 4: Data Importance-Aware Reinforcement Learning for Multi-Attack Defense

  This thesis proposes a data-centric reinforcement learning defense in which per-sample importance and vulnerability are profiled to guide fine-grained protection

strategies. The effectiveness of this method is demonstrated against diverse attack types,Äîincluding backdoor, model stealing, and membership inference,Äîshowing that the defense generalizes beyond single-attack scenarios.

Together, these contributions establish a reinforcement-guided unlearning paradigm as a principled foundation for privacy-preserving and attack-resilient FL.

The security and privacy risks examined in this thesis have direct relevance to real-world FL deployments. In healthcare, attacks such as membership inference and gradient inversion can reveal sensitive diagnostic information, posing risks to patient confidentiality and compliance with regulations including GDPR and HIPAA. In Internet-of-Things (IoT) environments, compromised devices may submit manipulated or malicious updates, potentially causing incorrect global model behaviour and leading to unsafe outcomes in smart-home systems, autonomous sensing, or industrial monitoring. In financial services, model stealing and reconstruction attacks may allow adversaries to infer private customer attributes, disrupt fraud detection mechanisms, or obtain proprietary decision-making logic. These examples illustrate that vulnerabilities in FL systems are not theoretical; they can directly affect the safety, integrity, and reliability of deployed models. Addressing these risks is therefore essential for the wider adoption of FL in practical applications.

## 1.3 Thesis Organization

This thesis explores the challenges of security, privacy, and robustness in FL and federated unlearning, with a particular emphasis on adversarial threats and adaptive defense mechanisms. The main aim is to develop reinforcement learning-driven frameworks that enable dynamic and context-aware protection against a wide range of attacks, while preserving model utility and meeting regulatory requirements.

The contributions of this thesis can be summarized as follows:

- Theoretical analysis: A rigorous investigation of the vulnerabilities in FL and unlearning, including gradient inversion, poisoning, membership inference, and model stealing, thereby establishing the limitations of existing defenses.
- Adaptive defense design: The development of reinforcement-learning agents that dynamically guide defense strategies-such as noise injection, gradient suppression, and selective unlearning-based on client contribution, privacy sensitivity, and computational cost.

- Comprehensive evaluation: Empirical studies conducted on widely used benchmarks (e.g., MNIST, CIFAR-10, FaceScrub, MovieLens), demonstrating that the proposed methods achieve superior trade-offs in accuracy, robustness, and efficiency compared to static defense baselines.

The remainder of this thesis is organized as follows:

- **Chapter 2** introduces the background on FL, unlearning, and adversarial attacks, and surveys related work.
- **Chapter 3** presents the theoretical analysis of vulnerabilities in federated systems.
- **Chapter 4** proposes the reinforcement-learning based defense framework, including algorithmic design and reward modeling.
- **Chapter 5** reports the empirical evaluation of the defense framework under multiple attack scenarios.
- **Chapter 6** extends the defense paradigm to federated unlearning, analyzing unique challenges and presenting RL-guided unlearning solutions.
- **Chapter 7** concludes the thesis and discusses limitations and promising future research directions.

CHAPTER 2

# BACKGROUND

This chapter presents the foundational concepts underpinning the thesis, including federated learning, adversarial threats, machine unlearning, and reinforcement learning. It also reviews related work and identifies gaps that motivate the methods developed in later chapters.

## 2.1 Preliminaries

This section introduces the fundamental concepts that underpin the research in this thesis. It covers the foundations of machine learning and deep learning, the principles of FL, the motivation for machine unlearning, and the adversarial threats commonly considered in distributed learning systems.

### 2.1.1 Fundamentals of Machine Learning

Machine learning (ML) refers to the design of algorithms that improve their performance on a task through experience with data. Formally, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ with inputs $x_i \in \mathbb{R}^d$ and labels $y_i \in \mathcal{Y}$, the goal of supervised learning is to learn a hypothesis function $h : \mathbb{R}^d \to \mathcal{Y}$ that minimizes the expected risk:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(h(x), y)], \tag{2.1}$$

where $\ell(\cdot)$ is a loss function and $\mathcal{P}$ denotes the underlying data distribution.

Deep learning (DL), a subfield of ML, leverages neural networks with multiple layers to approximate complex non-linear functions. Training typically relies on stochastic gradient descent (SGD) or its variants, which update model parameters iteratively to minimize the loss.

### 2.1.2 FL: Concept and Architecture

FL (FL) was proposed to address data privacy and locality constraints by enabling collaborative training without centralized data collection. In FL, $K$ clients participate in learning a global model $w$ by optimizing local objective functions $F_k(w)$, and a central server coordinates aggregation. The global objective is defined as:

$$\min_w F(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell(h_w(x_i), y_i), \tag{2.2}$$

where $n_k$ is the local dataset size of client $k$, and $n = \sum_{k=1}^{K} n_k$ is the total number of samples.

The most common aggregation strategy is Federated Averaging (FedAvg)[64], which updates the global model as the weighted average of local updates. While FL reduces the need for raw data sharing, it introduces challenges such as system heterogeneity, statistical non-IID data, and susceptibility to adversarial manipulation.

### 2.1.3 Machine Unlearning: Definition and Motivation

Machine unlearning (MU) is the process of removing the influence of specific data samples or clients from a trained model, without retraining from scratch. MU has gained prominence due to legal frameworks such as the General Data Protection Regulation (GDPR), which grants individuals the “right to be forgotten.”

Formally, given a model $w$ trained on dataset $\mathcal{D}$, if a subset $\mathcal{D}_r \subset \mathcal{D}$ must be removed, the unlearning process aims to produce a new model $w^{\setminus \mathcal{D}_r}$ such that:

$$w^{\setminus \mathcal{D}_r} \approx \text{Train}(\mathcal{D} \setminus \mathcal{D}_r), \tag{2.3}$$

while minimizing computational overhead. In federated settings, unlearning must be designed carefully to respect decentralization and communication constraints.

### 2.1.4 Adversarial Threats in Distributed Systems

Despite its advantages, FL is exposed to a broad range of adversarial threats:

- **Data poisoning attacks:** Malicious clients inject mislabeled or manipulated data into training to bias the global model.
- **Model poisoning attacks:** Attackers directly manipulate gradient updates before submission to the server, potentially embedding backdoors.
- **Gradient inversion attacks:** By analyzing shared gradients, adversaries reconstruct private training data of honest clients.
- **Membership inference attacks:** Attackers infer whether a given sample was part of the training dataset.
- **Model stealing attacks:** Through repeated queries, adversaries approximate the global model, potentially violating intellectual property.

These threats underline the necessity of robust and adaptive defense mechanisms, which form the core focus of this thesis.

### 2.1.5 Reinforcement Learning: Fundamentals and Applications

Reinforcement learning (RL) is a paradigm of machine learning in which an agent learns to make sequential decisions by interacting with an environment. The agent selects actions based on its current state, receives feedback in the form of rewards, and updates its policy to maximize the cumulative reward over time.

Formally, RL problems are often modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where:

- $\mathcal{S}$ is the state space representing possible situations of the environment.
- $\mathcal{A}$ is the action space, containing all possible actions the agent can take.
- $P(s'|s,a)$ denotes the transition probability from state $s$ to state $s'$ given action $a$.
- $R(s,a)$ is the reward received after taking action $a$ in state $s$.
- $\gamma \in [0,1]$ is the discount factor balancing immediate and future rewards.

The objective of RL is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, \pi\right]. \tag{2.4}$$

Two widely used categories of RL algorithms are:

- **Value-based methods:** e.g., Q-learning and Deep Q-Networks (DQN), which approximate the action-value function $Q(s,a)$ and select actions accordingly.
- **Policy-based methods:** e.g., Policy Gradient and Proximal Policy Optimization (PPO), which directly optimize the policy by gradient ascent on the expected reward.

In the context of this thesis, RL provides a natural framework for adaptive defense in FL and unlearning. The dynamic and uncertain nature of adversarial threats makes static defenses inadequate. By modeling the defense problem as an MDP, the RL agent can learn to select optimal defense actions (e.g., noise injection, gradient suppression, or unlearning) based on observed states (e.g., client contribution, privacy risk, computational cost), thereby achieving robustness and adaptability.

## 2.2 Literature Review

### 2.2.1 FL and Its Challenges

FL (FL) has emerged as a paradigm shift in machine learning, enabling collaborative model training without the need to centralize raw data. The seminal FedAvg algorithm [71] established the foundation of FL by aggregating locally trained models into a global model. Since then, FL has found applications in domains such as healthcare, finance, edge computing, and mobile services, where privacy concerns prevent direct data sharing.

Despite its appeal, FL suffers from several inherent challenges. Data heterogeneity [57] (non-IID distributions) leads to inconsistent updates across clients, slowing down convergence and reducing the accuracy of the aggregated global model. System heterogeneity [16] arises from differences in computational and communication capabilities across devices, complicating synchronization and resource allocation. Furthermore, FL is heavily constrained by communication overhead [55], as iterative parameter exchanges dominate the training cost. Finally, although FL avoids direct data sharing, model updates themselves can leak sensitive information and introduce new attack surfaces. Together, these factors highlight that FL is far from a complete solution to privacy and security, and instead creates new vulnerabilities that adversaries can exploit.

### 2.2.2 Adversarial Attacks on Federated Systems

A substantial body of work has investigated the vulnerabilities of FL. Attacks in federated settings can be broadly categorized as data poisoning, model poisoning, and privacy

attacks.

- Data poisoning attacks corrupt the training process by altering local datasets. Examples include label flipping or trigger-based backdoors, which can significantly bias the global model toward adversarial objectives while remaining hidden in normal operations. [75]
- Model poisoning attacks manipulate gradients or weight updates before transmission. Such Byzantine behaviors allow attackers to introduce malicious objectives directly into the aggregation process, often bypassing detection mechanisms. [112]
- Privacy attacks exploit shared updates. Techniques such as gradient inversion can reconstruct sensitive training samples from model updates, while membership inference attacks expose whether a particular data record was included in the training set. Attribute inference attacks further attempt to deduce private features from observed gradients. [132]
- Model stealing attacks target the functional replication of the global model through repeated adaptive queries, undermining intellectual property protection and exacerbating downstream vulnerabilities. [103]

These attacks demonstrate that FL is not only vulnerable to threats inherited from centralized machine learning but is also exposed to new adversarial strategies due to its distributed, decentralized nature.

### 2.2.3 Defense Mechanisms in FL

In response, multiple defensive strategies have been proposed, typically falling into three categories:

- Robust Aggregation Rules. Algorithms such as Krum, Trimmed Mean, Median, and Bulyan attempt to identify and neutralize malicious updates by analyzing deviations in gradient space. These methods are efficient and model-agnostic, but their effectiveness diminishes under colluding adversaries or adaptive attacks. [92]
- Privacy-Preserving Mechanisms. Techniques such as Differential Privacy (DP) [1], Secure Multi-Party Computation (SMPC) [73], and Homomorphic Encryption (HE) [82] aim to protect sensitive information during model exchange. While they provide strong theoretical guarantees, they incur significant trade-offs in accuracy, efficiency, and scalability, often limiting practical deployment.

- Adversarial Training and Anomaly Detection. [94] Defensive approaches based on outlier detection or robust optimization seek to detect and down-weight malicious clients, while adversarial training strengthens resilience against specific attacks. However, these defenses are typically attack-specific and fail to generalize to new or combined adversarial strategies.

Overall, the surveyed paper emphasizes that most defense mechanisms remain static, narrow in scope, and insufficiently adaptive, leaving FL systems vulnerable to evolving threat landscapes.

### 2.2.4 Machine Unlearning in Federated Contexts

Beyond adversarial defense, there is a growing recognition of the importance of machine unlearning, i.e., the ability to remove the influence of particular data points or clients from trained models. This demand is largely driven by privacy regulations such as GDPR's ‚Äúright to be forgotten.‚Äù [127]

Existing approaches to unlearning can be divided into:

- Exact retraining, which ensures perfect removal by discarding the target data and retraining the model from scratch. While effective, this approach is computationally infeasible for large-scale FL deployments.
- Approximate unlearning, which attempts to estimate and remove the effect of specific data contributions using techniques like influence functions or incremental updates. These methods offer efficiency but often lack formal guarantees of privacy or robustness.

In federated settings, unlearning becomes even more challenging due to distributed architectures, communication costs, and the potential presence of malicious participants. The literature highlights that federated unlearning is still in its infancy and rarely incorporates adversarial-aware mechanisms, leaving a critical gap in ensuring privacy-compliant, robust FL systems.

### 2.2.5 Reinforcement Learning in Federated and Adversarial Settings

Reinforcement Learning (RL) has recently gained attention as a potential tool to address the limitations of static strategies in FL (FL). Unlike conventional optimization meth-

ods, RL explicitly models the interaction between an agent (e.g., the server or defense mechanism) and an environment (e.g., participating clients, communication conditions, or adversarial activities). Through trial-and-error learning, RL agents can dynamically adapt to system states and optimize long-term objectives such as accuracy, robustness, privacy, and efficiency. [3]

Beyond system optimization, RL provides a natural framework for designing adaptive security defenses in FL. Unlike traditional defenses that are static or attack-specific, RL agents can continuously monitor indicators of adversarial behavior-such as gradient divergence, anomaly scores, or shifts in model loss-and choose optimal defensive actions. Examples include:

- Injecting calibrated noise to mitigate privacy leakage.
- Suppressing or re-weighting suspicious updates to resist poisoning attacks.
- Triggering selective unlearning or retraining in response to detected adversarial influence.

This sequential decision-making perspective transforms defense from a one-time static mechanism into a dynamic policy that evolves with the adversarial environment.

### 2.2.6 Security Landscape in Federated Learning

Recent studies show that federated learning faces many security risks in practice. Beyond early poisoning and gradient inversion attacks, researchers have found that differences among clients can create additional weak points. When clients hold data from different distributions, their gradients can vary in ways that allow adversaries to hide harmful updates among normal ones. This means that threats can arise from the design of the training process itself and not only from the model.[59]

Other work shows that the optimization steps used in federated learning can strengthen the effect of malicious updates. Local learning rates, momentum and repeated local steps may cause a small adversarial update to influence later rounds.[2] This allows attacks to remain unnoticed during training and appear only at a much later stage. Researchers have also found that information may leak through update patterns, timing differences and model statistics. These results suggest that the security challenges in federated learning are broader than early studies assumed.[4]

### 2.2.7 Privacy Leakage and Inference Attacks

Privacy risks in federated learning extend far beyond direct data reconstruction.[136] Membership inference attacks can identify whether a sample is part of the training set.[27] Attribute inference attacks may reveal hidden or sensitive features using gradients or prediction changes. [72] Some studies show that attackers can rebuild models that closely match the performance of the original global model. These findings show that privacy leakage can happen even when data remains on the client device because gradients and model updates still carry information that can be exploited.

### 2.2.8 Robust Aggregation and Its Limits

Robust aggregation has been widely explored as a defense against malicious clients. Methods such as Krum, Median and Trimmed Mean aim to remove harmful contributions. Later work shows that these methods have important limits.[6] If multiple attackers coordinate, their updates can stay within the accepted range and bypass the defense. Many methods also rely on strong assumptions about normal client behavior that do not hold in real systems. As a result they may incorrectly remove benign updates or fail to detect harmful ones. [116]

Static defenses do not adjust when attacks change. Some studies show that attackers can learn how an aggregation rule behaves and design updates that avoid detection. Robust aggregation also does not address privacy leakage.[83] These limits have encouraged the development of adaptive defenses that can adjust their behavior over time.

### 2.2.9 Federated Unlearning and Open Challenges

Federated unlearning has become an important research topic driven by privacy requirements and system reliability.[30] Existing approaches include retraining, approximate removal and influence based methods. Retraining is reliable but too costly for large federated systems. [10]Approximate removal aims to reverse the influence of certain clients or samples but becomes less accurate when models are complex or client data is heterogeneous.

Several works point out that repeated aggregation steps make unlearning more difficult. The influence of a single sample becomes mixed with other updates across many rounds. [68] Simple parameter adjustments often cannot fully remove this effect. Unlearning may also introduce new risks. Attackers may create updates that look normal during training but become harmful after unlearning removes certain benign updates.

This shows that unlearning can change the balance of gradients and reveal hidden malicious components.

Another challenge is verifying that unlearning has succeeded. Some studies propose audit logs and influence measurements but practical verification remains difficult. [38] These studies show that federated unlearning is still at an early stage and requires stronger guarantees and more efficient algorithms.

### 2.2.10 Reinforcement Learning for Optimization and Defense

Reinforcement learning has been used to improve both efficiency and security in federated learning. Early work applies reinforcement learning to client selection and communication control. [24] These studies show that reinforcement learning can find strategies that balance accuracy and cost more effectively than fixed rules.

Recent work examines reinforcement learning as a defense against adversarial behavior. [109] Reinforcement learning agents can learn from past rounds and identify unusual updates. Unlike fixed defenses, they can adjust their behavior when the attack changes. However many existing approaches operate only at the client or model level and do not consider the different importance or sensitivity of each sample. [44] Most methods also target one attack type at a time. These gaps motivate the reinforcement learning methods developed in this thesis which use fine grained signals to guide unlearning and defense decisions.

### 2.2.11 Privacy Regulations and Compliance

Privacy laws such as GDPR and CCPA influence how federated learning systems should be designed. GDPR includes the right to be forgotten which requires that data be removed upon request and that models no longer depend on that data. [100] In federated learning data remains on the device but its influence is stored in the model parameters. Removing local data alone does not meet this requirement.

Studies show that compliance requires a method to remove or reduce the model's dependence on specific data. Retraining can achieve this but is usually too costly. [51] This has led to interest in efficient and verifiable unlearning methods.Privacy regulations therefore provide strong motivation for unlearning and show that it is necessary for using federated learning in sensitive fields such as healthcare and finance.

CHAPTER 3

# DEFENDING AGAINST GRADIENT INVERSION ATTACKS IN FL VIA STATISTICAL MACHINE UNLEARNING

This chapter focuses on defending against gradient inversion attacks through a statistical machine unlearning framework. It introduces the threat model, details the proposed defense method, provides theoretical analysis, and evaluates its effectiveness through experiments.

## 3.1 Introduction

FL (FL) has recently emerged as a novel approach in machine learning, enabling the training of large-scale models without the need for individual clients' data access [71], [63], [59]. This system operates through a distributed structure and training process, prioritizing client privacy. In FL, a global model is collaboratively trained over multiple global rounds. Each round involves clients receiving an initial model from the server, which they optimize using their local data to create a local model. Clients then update their local models over several local rounds and send these updates back to the server. The server aggregates these updates to complete a global round, with the overarching aim of preserving client privacy by only requiring the upload of model updates, not raw data.

However, subsequent research has identified vulnerabilities in FL systems, specifically regarding gradient inversion attacks, where training samples can be recovered

from gradients [136], [27], [117]. These techniques aim to minimize the difference between model-predicted gradients and actual gradients. The recovery process involves generating random data and labels, then using an iterative optimization procedure to align virtual gradients with real data gradients. When this optimization converges, private data can potentially be fully reconstructed. Recent studies [23], [134], [15] have even demonstrated the feasibility of reconstructing original data using closed-form algorithms. Understanding the relationships among input data, model parameters, and layer gradients is crucial to finding the most accurate solutions with minimal error.

To combat gradient inversion attacks, extensive research has focused on developing defense mechanisms. These include gradient perturbation methods such as [136], where selectively eliminating certain gradients can make recovered images indiscernible to humans. Other explored strategies include conventional encryption techniques like secure aggregation [7] and homomorphic encryption of gradients. While these methods can mitigate attacks, they often introduce significant encryption overhead. Despite advancements, current defense methods still face various limitations.

There are two main challenges in this work:

- Challenge 1: The precise gradients that are shared between global and clients raise the risk of information leakage, such as gradient inversion attacks. The introduction of gradients subject to techniques such as noise injection can lead to a reduction in model accuracy.

- Challenge 2: Communication efficiency is a crucial component, and reducing the communication load between the client and server sides would constitute an improvement in efficiency.

Knowledge distillation, often referred to as teacher-student learning, involves training a simpler, smaller model (the student) to emulate a more complex one (the teacher) [33]. This technique is known to bolster model robustness and increase resilience to adversarial attacks [87].

In our study, we introduce a novel defense method against gradient inversion attacks by integrating principles from statistical machine unlearning and knowledge distillation techniques. The concept of statistical machine unlearning, introduced in [12], focuses on reducing a trained model's dependence on specific data points. It does this by training models using only statistical queries about the data, instead of individual data samples, thus enabling selective data forgetting for privacy enhancement.

In our approach, the student model is derived from local models present on the client side. We then extract statistical information from the client's data within these student models, deliberately breaking certain links between the original data and the global model. Our method involves using only a subset of this statistical information for training local models, instead of direct client data. This maintains the integrity of the original training steps and avoids additional complexities that could impact model accuracy. Inspired by statistical machine unlearning, we leverage statistical data for training, which protects against information leakage. Moreover, by employing knowledge distillation, we reduce the model size, thus minimizing the updates sent to the server. This not only cuts down on communication overhead but also boosts the efficiency of global model training. Our findings demonstrate the superiority of this strategy in not only preserving data privacy against gradient inversion attacks but also enhancing model accuracy and efficiency compared to other methods. Our work makes three main contributions to the field:

- 1. We propose a novel defense method against gradient inversion attacks, which is inspired by the principles of statistical machine unlearning.
- 2. We design the knowledge distillation to design defense method against gradient inversion attack and have some advantages compared with most state-of-the-art recently proposed methods.
- 3. We conduct extensive experiments to compare the performance of our proposed method with many sate-of-the-art defense methods, providing evidence of its superior performance.

## 3.2 Related Work

### 3.2.1 Gradient Inversion Attacks

Gradient inversion attacks was first introduced by [136] in 2019, highlighting the risk of privacy leakage from the weights and gradients of shared models. Their method, known as DLG, reconstructed authentic images and labels by minimizing the difference between original gradients and synthetic gradients generated from random noise. They preferred the L-BFGS optimizer over Adam for faster convergence and better performance, though their method was limited to small $32x32$ images and a batch size of 8.

Building on DLG, [126] developed iDLG, an improved technique for extracting ground-truth labels from gradients. iDLG showed superiority over DLG, but its label extraction was only effective in single-image restoration scenarios [113].

Significant advancements were made by [27], who successfully reconstructed input images from the ImageNet dataset. Their work compared the effects of trained versus untrained models on image recovery and explored using updated weights for local image restoration instead of gradients. Their introduction of cosine dissimilarity for measuring gradient and total variation regularization for image denoising greatly enhanced image reconstruction quality [129].

[117] expanded label extraction to batch mode with GradInversion, eliminating the need for duplicate labels in a mini-batch. However, this method might be less practical for large batch sizes or limited class numbers. They also examined the influence of random seeds on recovery outcomes and introduced a group consistency regularization using the RANSAC-flow algorithm.

These studies collectively enhance the understanding and techniques of gradient inversion attacks for private information reconstruction. They have explored various optimization strategies, extensions, and scenarios, including different image sizes, batch sizes, and label extraction methods, to refine and broaden the applicability of these attacks.

### 3.2.2 Defense against Gradient Inversion Attacks

Numerous defense strategies have been introduced to mitigate gradient inversion attacks, and these approaches can be categorized into three distinct groups, each tailored to address different objectives and operational aspects.[47][128][137]

#### 3.2.2.1 Encrypt gradients.

To mitigate the risk of gradient inversion attacks, cryptographic methods have been employed to protect gradients. [7] introduced a secure aggregation protocol in FL, utilizing secret sharing to aggregate gradient vectors securely. On another front, [80] suggested the use of homomorphic encryption to safeguard gradients during transmission. While these cryptographic techniques bolster privacy, they require specific configurations and can lead to significant implementation expenses. Furthermore, even with secure aggregation, vulnerabilities to gradient inversion attacks persist, especially from honest-but-curious servers or clients, posing ongoing privacy challenges.

#### 3.2.2.2 Perturb gradients.

Another approach to thwarting inversion attacks involves perturbing gradients. In their study, [136] propose gradient pruning, which involves setting gradients of small magnitudes to zero as a defensive measure. However, the effectiveness of this approach may diminish when confronted with more sophisticated attacks. An alternative strategy is to introduce noise into gradients, inspired by Differential Privacy Stochastic Gradient Descent (DPSGD) [22], as suggested by [106]. By adding Gaussian or Laplacian noise to gradients, privacy can be enhanced, but this may reduce accuracy. Researchers also explore improved pre-training techniques and larger batch sizes for DPSGD training [27] [96].

#### 3.2.2.3 Weak encryption of inputs

The mixup tactic is another defensive strategy where neural networks are trained on composite images created through linear combinations of image pairs [125]. Mixup has been shown to improve the generalization and robustness of neural networks to adversarial examples. Inspired by mixup, [48] propose InstaHide as a lightweight instance encoding scheme for private distributed learning. InstaHide combines an image with other images and coefficients to create a composite image, which is then encrypted using a random sign-flipping pattern. Training on such encoded images maintains good classification accuracy while offering privacy. However, recent research reveals potential vulnerabilities when InstaHide encodings are exposed to attackers [13].

### 3.2.3 Summary of Defense Methods

Defensive methods against gradient inversion attacks encompass a variety of techniques, including encryption-based approaches, perturbing gradients, weak encryption of inputs, as well as data augmentation. The inspiration behind techniques such as gradient perturbation and weak encryption of inputs can be traced back to concepts from differential privacy and secure multi-party computation. These methodologies provide important insights into protecting privacy while maintaining the utility of FL systems. By incorporating elements of differential privacy and secure multi-party computation into our proposed strategy, we aim to enhance the overall security and privacy of FL systems while maintaining efficiency and utility. While these methods aim to enhance privacy, they often come with trade-offs in terms of computational costs and training efficiency. Encryption-based methods and data augmentation may introduce additional computa-

tional overhead, while methods focusing on gradient perturbation and weak encryption of inputs could compromise training efficiency and accuracy due to the privacy-utility trade-off. In selecting an appropriate defense strategy, it is crucial to carefully consider the specific use case and threat model. Given the challenges inherent in these defense methods, we propose a strategy that involves leveraging partial information for model training. This approach serves a dual purpose: it reduces the computational burden on federated systems by processing only a subset of update information, while selectively using a subset of updates during training to mitigate gradient inversion and preserve utility.

### 3.2.4 Recent Advances in Gradient Inversion Attack

In FL, recent studies have delved deeply into the nuances of gradient inversion attacks and their defenses. [98] presents an advanced approach that moves beyond conventional gradient-based inversion by incorporating adversarial data priors. This method demonstrates increased effectiveness in sensitive areas like facial recognition, but it also raises substantial privacy concerns.

[43] explores the practical aspects of these attacks in the healthcare sector, proposing new baseline methodologies for attacks. While these new baselines address practical challenges in healthcare-related gradient inversion attacks, their applicability may be confined to specific contexts.

The research by [66] reveals potential vulnerabilities to external threats in FL systems. This study sheds light on previously unaddressed security gaps, although its effectiveness might depend on certain external conditions.

Furthermore, [28] broadens the conversation by offering a comprehensive framework that covers both enhanced attack methods and defense strategies. This balanced approach promises thorough protection but could entail intricate implementation processes.

Lastly, [77] introduces mixed quantization as an innovative defensive tactic to boost data confidentiality against gradient inversion attacks. This strategy not only improves confidentiality but also reduces communication overhead, though it may affect learning efficiency or accuracy.

Collectively, these studies highlight the evolving and vital importance of security within FL environments.

### 3.2.5 Knowledge Distillation

Knowledge distillation is a technique in machine learning where a small, simpler model (the student) is trained to emulate a larger, more complex model (the teacher). This process enables the student model to leverage the insights and generalizations of the teacher, which might have been trained on more extensive data or for a longer duration. The core principle of knowledge distillation involves a specialized loss function that not only considers the actual labels of the training data but also the soft probabilities (confidence levels) output by the teacher model. This loss function is designed to guide the student model in mimicking the teacher model's predictions, including the certainties associated with each class label [33].

The following is loss for the knowledge distillation:

$$L_{KD} = (1-\alpha)\cdot L_{CE}(y_i, p_i) + \frac{\alpha}{T^2}\cdot \text{KL\_Divergence}(q_i, p_i) \tag{3.1}$$

$T$ denotes the temperature parameter, which controls the softening of the teacher model's probabilities. $y_i$ is the true label (one-hot encoded) of the training example. $q_i$ is the softened or soft target probabilities predicted by the teacher model and $p_i$ is the probabilities predicted by the student model. The first term on the right is the standard cross-entropy loss between the true labels $y_i$ and the student model's predictions $p_i$. The second term is the Kullback-Leibler (KL) Divergence between the softened target probabilities $q_i$ (teacher's predictions) and the student model's predictions $p_i$, scaled by a factor. The factor of $\frac{\alpha}{T^2}$ (usually a small value) controls the trade-off between the two terms, and The parameter $T$ controls the softening effect.

### 3.2.6 Gradient Inversion Attack

In this study, we focus on a gradient inversion attack method that matches gradients between input data and real data, as detailed in [27]. This method can do a better recovery performance compared to other methods, and it is one of the state-of-the-art gradient inversion attack methods. This technique aims to recover input data for image classification tasks. Given a neural network with parameters $\theta$ and gradients $\nabla_\theta L_\theta(x', y')$ computed from private data $(x', y')$, the attacker attempts to approximate $x \in \mathbf{R}^{b\times d}$ by solving the following optimization problem:

$$L_{grad}(x; \theta, \nabla_\theta L_\theta(x', y')) + \alpha R_{aux}(x) \tag{3.2}$$

Here, $L_{grad}(x;\theta,\nabla_\theta L_\theta(x',y'))$ enforces gradient matching with the given gradients $L_\theta(x',y')$, while $R_{aux}(x)$ regularizes the recovered image based on image prior information [80]. Prior work by [136] and [126] attempted similar approaches with limitations.

Notably, [27] significantly improved the attack by using cosine distance as $L_{grad}$ and total variation as $R_{aux}(x)$. They achieved successful reconstruction of low-resolution images, even with a batch size of 100 or a single high-resolution image. Subsequent research [106] explored the impact of different training configurations on attack efficacy.

Furthermore, [117] enhanced the attack for high-resolution images by incorporating a new image prior term based on batch normalization statistics and a regularization term ensuring consistency across multiple attack attempts.

An alternative research direction proposed by [135] formulates the gradient inversion attack as a recursive procedure instead of an optimization problem but is limited to low-resolution images with a batch size of 1.

In our work, we employ one of the state-of-the-art attacks, specifically [27]. This attack optimizes the following objective function:

$$\arg\max_x 1 = \frac{\langle \nabla_\theta L_\theta(x,y), \nabla_\theta L_\theta(x',y') \rangle}{||\nabla_\theta L_\theta(x,y)||||\nabla_\theta L_\theta(x',y')||} + \alpha_{TV} R_{TV}(x) \tag{3.3}$$

Here, $\langle \cdot,\cdot \rangle$ represents the inner product between vectors, and $R_{TV}(\cdot)$ measures the total variation of images.

## 3.3 Proposed Method

### 3.3.1 Threat Model

In this study, we establish a unified threat model and provide definitions for key technical terms. Our focus is on a specific type of adversary known as an honest-but-curious server, which aims to recover clients' data. We assume that the server itself acts as the attacker in this scenario.

**Attackers' Goal:** The primary objective of the attacker is to retrieve the raw data of clients by extracting private information from the trained model, which is generated during the training process.

**Attackers' Ability:** The attacker possesses the capability to independently retain and analyze updates provided by individual clients. However, they are unable to interfere with the collaborative learning process. This means that the attacker cannot modify the

model architecture to suit their attack or provide malicious global parameters that do not correspond to the actual global model learned.

**Attackers' Knowledge:** The attacker is granted access to the batch normalization statistics. Batch normalization (BatchNorm) [50] is a widely used technique in deep learning and neural network training that aims to improve the convergence and stability of training. It works by normalizing the activations of each layer in a neural network using statistics computed over mini-batches of data.

### 3.3.2 Defense Objectives

Our primary objective is to defend against attacks that aim to recover private data without modifying models or federation settings while minimizing the impact of our defense on model performance and preserving the effectiveness of the training process. We address the gradient inversion attack in FL, which reconstructs input data from gradients obtained from local models' raw data. Our key insight is to disrupt the transformation process from raw data to gradients, making it more challenging to invert the gradients during the training stage while ensuring the smooth execution of other stages. By interfering with or bypassing the gradient calculation process, we can confuse adversaries by leading them in the wrong direction when attempting to reconstruct the original data. Specifically, during the training process, we replace the gradients derived from raw data $x^s$ with gradients $\nabla d$ calculated using statistical information. To ensure the model functions normally, we aim to make $\nabla_d$ approximate $\nabla x^s$. Through aggregation, the influence on the global model becomes negligible. As a result, the gradient does not contain information about the original data, obstructing the recognition of the reversed data by adversaries. Our defense strategy aims to achieve three objectives:

- Objective 1: Preserve the efficiency of the training process and avoid adding complex extra stages to the FL framework.
- Objective 2: Protect clients' data by generating visually dissimilar reconstructed data with new gradients.
- Objective 3: Maintain or reduce the drop in model accuracy by ensuring that the new gradient direction remains similar to the original gradient direction.

To achieve our objectives, we introduce a method tailored for a FL system that comprises several local client models. In our approach, these local models act as teacher models in a knowledge distillation process, which begins after a certain number of local

epochs have concluded. We start with a student model initialized with random weights. This student model is then trained using a dual loss function: a classification loss and a knowledge distillation loss. Through iterative training, the student model gradually aligns its predictions with those of the teacher models. This process enhances the student model's generalization and performance, while maintaining a smaller and more efficient size.

When data is input into the student model, it generates an update set comprising various parameters. Instead of uploading the entire set of parameters to the server, we opt for selective retention. Typically, we prioritize retaining parameters from the initial layers of the model, as these are closer to the original data and likely contain more sensitive information. By doing so, we aim to balance the need for privacy with the efficiency and effectiveness of the learning process in the federated system.

In contrast to methods that integrate noise or mixing procedures at or before the training stage, our approach initially applies knowledge distillation to the local models, followed by parameter cropping during the training phase. The idea of cropping parameters in neural networks is not new. For instance, [42] introduced the concept of pruning parameters to create more efficient networks without compromising accuracy. This precedent reinforces the viability of our method in maintaining model performance.

Moreover, our experimental results, which will be discussed in a later section, further affirm that our approach exerts minimal impact on the model's accuracy. By combining knowledge distillation with strategic parameter cropping, we strike a balance between preserving model performance and protecting against gradient inversion attacks.

### 3.3.3 Our Defense Method

Our workflow for defending against gradient inversion attacks encompasses both the training of the model using our defense method and the subsequent execution of the attack on this trained model. The first step involves employing knowledge distillation to simplify the local model. This process not only improves the efficiency of transmitting the distilled model compared to the original but also integrates seamlessly into the FL framework without adding complex steps, thereby fulfilling Objective 1. Moreover, the updates from the distilled student model diverge more from the original data than those from the original local model, aiding in achieving Objective 2. Through knowledge distillation, the model retains key features of the original model, ensuring minimal loss in accuracy and contributing to Objective 3.

Next, we draw inspiration from statistical machine unlearning, training the model using only a subset of statistical information. This approach not only maintains model accuracy but also effectively separates training data from training information, complicating attackers' efforts to extract training data. This step helps in achieving all three objectives, as it upholds accuracy while enhancing data privacy.

Before attackers can initiate their methods, the model is trained using our defense technique. Our method is based on the standard FL algorithm, Federated Averaging (FedAvg), which lays the foundation for our defense strategy. This initial step is crucial in setting up a robust defense against potential gradient inversion attacks.

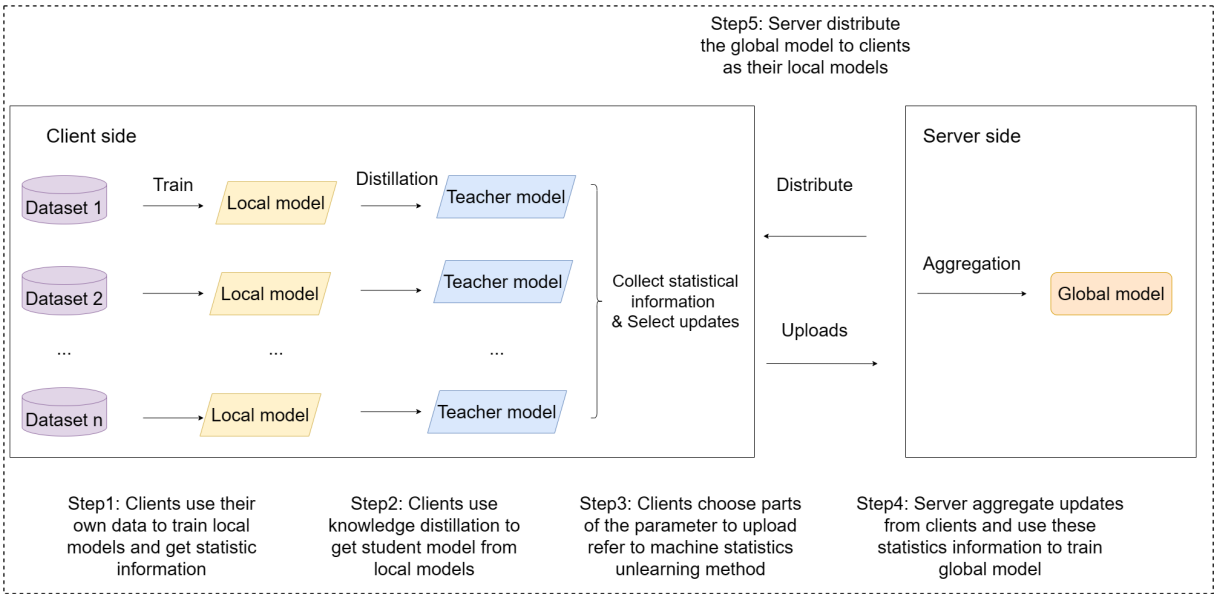


Figure 3.1: Overview of Defense Method

Figure 3.1 provides an overview of our method within the context of a federated system comprising $n$ clients, each in possession of their unique raw data. In the conventional Federated Averaging (FedAvg) algorithm, the model undergoes training via updates received from individual clients. These clients then share their model parameters with a central server for the collective training of a global model. However, our proposed defense mechanism deviates from this standard practice. Here, the data from each client is directed to local models, and their respective updates are transmitted to the server. The statistics machine unlearning method incorporates an intermediary layer comprising a limited number of summations, strategically positioned between the learning algorithm and the training data [12]. Referring to the method, we must determine all the layers information in local models, the number of layers denoted as $l$, and the transformation

g-functions are fixed, denoted $g_1, g_2, ...g_l$. Then we present the method with the following form:

$$Learn(\sum_{x_i \in X} g_1(x_i, y_{x_i}), \sum_{x_i \in X} g_2(x_i, y_{x_i}), ... \sum_{x_i \in X} g_l(x_i, y_{x_i})) \tag{3.4}$$

where $x_i$ is a training data sample and $y_{x_i}$ is its label. For the purpose of unlearning, let $G_h$ be $\sum_{g_h}(x_i, y_{x_i})$, if the $x_q$ is the objective data sample, $G_h$ can be $G_h - g_h(x_i, y_{x_i})$. For our method, we just discard the first layer information $\sum_{x_i \in X} g_1(x_i, y_{x_i})$. This layer serves to disentangle and disrupt the dependencies within the system. To simulate the unlearning method, we take the parameters in models as the training information, for each client in FL, we opt to omit the parameters of the initial layer within the local model, forwarding only the remaining parameters for participation in the global model's training. This process conceptually resembles employing the first segment of a fully connected neural network as the local model and using the output of this segment as input for the subsequent phase.

**1. Train local models:** We initiate our approach by having clients train their respective local models. The updates generated during this process are closely tied to the original data, which is crucial for both defending against attacks and strengthening our overall defense strategy. While traditional defense methods often introduce noise or data manipulation to prevent attackers from extracting sensitive information, such techniques can complicate training and potentially reduce the accuracy of the global model. In contrast, our approach follows standard FL practices, training local models without modifications to preserve the integrity and effectiveness of the global model.

**2. Distill local models:** In the subsequent phase, we apply knowledge distillation to refine the local model. This serves two primary goals: firstly, preserving the performance of the local model while distancing it from the training data, thereby enhancing privacy. The distilled model retains the core competencies of the original but with reduced direct data dependency. Secondly, this streamlined model reduces the load involved in transmitting model updates from local clients to the server. The complexity of the distillation process largely depends on the intricacies of the teacher model and the dataset's size. In a FL environment, where each client typically has access to only a subset of data, the resource demand for training student models is comparatively low, ensuring feasibility even for clients with limited computational capabilities.

**3. Select updates:** Once the student models are obtained through knowledge distillation, we draw upon the principles of statistical machine unlearning and choose to use only a portion of the statistical information as updates for the server. Specifically, in

our method, only the statistics information from the first layer of the model is uploaded. This selective approach is crucial in reducing the amount of potentially sensitive data transmitted.

**4. Aggregate updates:** Upon receiving these selective updates from various clients, the server performs aggregation, a standard procedure in FL. Despite some updates being omitted in this process, the global model's accuracy remains impressively high, often comparable to that of a traditional federated system.

**5. Distribute global model:** Following the aggregation phase, the server updates the global model and redistributes it to each client. This updated global model serves as the starting point for each client's local model in the subsequent training round.

By following these steps, we complete the cycle of training and updating the global model within a FL framework. This approach not only adheres to the core principles of FL but also integrates enhanced privacy measures, ensuring a balance between model performance and data confidentiality. Each client receives the updated global model, which then forms the basis for further local training in the next round, maintaining the ongoing collaborative learning process. By repeating the aforementioned steps over $T$ epochs, we attain a protected model, effectively backing the privacy and security of the model while capitalizing on the advantages of collaborative.

**Algorithm 3.1** Training Method of Local Model

**Input:** train set $D = \{(x^{(n)}, y^{(n)})_{n=1}^{N}\}$, Local epoch $T$;
Model from server $\theta$;
$\theta_s$=knowledge distillation($\theta$)
**repeat**
  **for all** $(x, y) \in D$ **do**
    $S_p \leftarrow \theta_s(x, y)$;
    $t \leftarrow t + 1$;
  **end for**
**until** $t = T$
**for all** $i \in S_p$ **do**
  **if** $i \in$ layer1 **then**
    discard;
  **else**
    $S_s \leftarrow i$;
  **end if**
**end for**
**return** $S_s$;

Algorithm 3.1 delineates the training procedure employed within our methodology,

which takes into account the training dataset and the specified count of local training epochs as its primary inputs. These inputs constitute the foundational elements of a FL system. We usually use the training data and local epoch to train the local model. Initially, the algorithm retrieves the model from the central server and subsequently leverages the technique of knowledge distillation to derive the student model $\theta_s$. This initial step in our approach to model training through knowledge distillation serves the purpose of simplifying the local model, effectively distancing it from its original data representation before its incorporation into the FL process. The training process on the student model $\theta_s$ is then iterated until the local training epoch $T$ is successfully completed. At this juncture, the local model reaches a state of completion and is seamlessly replaced by the student model following the knowledge distillation process. Typically, we exclude the initial layer parameters in the local model and package the remaining information as updates for transmission to the server. The initial layer parameters typically encapsulate the highest volume of information derived from the original data within neural networks. To strike a balance between model utility and data privacy, it is deemed sufficient to selectively exclude only these first-layer parameters.

**Algorithm 3.2** Defense Gradient Inversion Attack on FL

Initialize $w_0$
**Server executes**:
**for** each round $t = 1, 2, ...$
  $S_t$ =(all clients)
**for** each client $k \in S_t$ **do**
  $w_{t+1}^k \leftarrow$ Client Update$(k, w_t)$
  $w_{t+1} \leftarrow \sum_{t=1}^{K} \frac{n_k}{n w_{t+1}^k}$
**end for**
**Client Update** $k, w$
**for** each local epoch $i$ from 1 to $E$ **do**
  batches←(data $P_k$ split into batches of size B)
**end for**
**for** batch b in batches **do**
  $w \leftarrow w - \nabla l(w; b)$
**end for**
**for all** $i \in S_p$ **do**
  **if** $i \in$ layer1 **then**
    discard;
  **else**
    $W_s \leftarrow i$;
  **end if**
**end for**
**return** $w_s$ to server

Algorithm 3.2 outlines the complete workflow within the FL (FL) system. From the client-side perspective, executing Algorithm 1 is critical. This involves local model training and the subsequent submission of updates to the server infrastructure. During this phase, we implement our defense mechanism on the client side. This effectively disconnects the uploaded updates from the original training data, resulting in updates that are substantially lighter than those in traditional FL.

On the server side, the primary task is to aggregate updates from all participating clients. These collective updates are essential for progressively refining and training the global model. This process embodies the collaborative learning essence of the FL framework, where the server plays a crucial role in synthesizing individual contributions into a cohesive, improved global model.

### 3.3.4 Intuitive Explanation of Statistical Queries

Statistical queries provide a way for the server to learn from clients without receiving gradients or any information that depends on individual samples. A statistical query defines a simple function that is applied to each data point on a client. Instead of returning the result for each point, the client reports only an aggregate such as an average value, a count or another summary statistic computed over its entire local dataset.

This mechanism removes the detailed structure that reconstruction attacks rely on. Gradient inversion requires access to complete gradient vectors because these vectors capture how the loss changes around specific training examples. When the server receives only coarse summaries, this relationship is no longer present. The mapping from the local dataset to the returned statistic becomes a many to one relationship, and the server cannot isolate the contribution of any individual sample.

In this sense statistical queries do not hide gradients but rather replace them with information that is too coarse to enable reconstruction. The server still receives useful global signals that guide learning, yet these signals do not contain the level of detail required to infer the underlying data points. This characteristic explains why statistical queries substantially reduce the risk of reconstruction based attacks.

### 3.3.5 Theoretical Analysis

#### 3.3.5.1 Use knowledge distillation to protect privacy

In knowledge distillation, a teacher model, which is typically a larger and well-trained model, is used to transfer its knowledge to a smaller student model. The teacher model's knowledge is distilled into the student model by encouraging the student model's outputs to match the softened probabilities produced by the teacher model. This process allows the student model to learn from the teacher model's expertise without directly accessing the original training data. Consequently, knowledge distillation can potentially alleviate privacy concerns by reducing the need to share sensitive training data with third parties.

From a theoretical standpoint, knowledge distillation can offer privacy benefits in several ways: Let's consider the scenario where a teacher model is trained on a large dataset $D$ and a student model is trained using knowledge distillation from the teacher model. Instead of sharing the entire dataset $D$, only the teacher model's predictions (i.e., softened probabilities) are shared. The key idea behind data protection is that the student model can achieve similar performance to the teacher model while using a

smaller dataset. It can be expressed as:

$$\min_{\theta_s} \frac{1}{N} \sum_{n=1}^{N} \mathscr{L}\left(f(\theta_s(x^{(n)})), f(\theta_t(x^{(n)}))\right) \tag{3.5}$$

where $\theta_s$ represents the parameters of the student model, $\mathscr{L}$ is the loss function, $f$ is the softmax function, and $\theta_t$ denotes the parameters of the teacher model. By minimizing this loss function, the student model acquires the ability to replicate the teacher model's predictions without requiring direct access to the training data. This illustrates how knowledge distillation safeguards data privacy by reducing the need to disclose sensitive training data.

In knowledge distillation, the teacher model acts as an abstraction layer, capturing important information from the original data and transferring it to the student model. This abstraction process allows the student model to leverage the teacher model's knowledge without directly accessing the specific training examples, thereby preserving data privacy.

Mathematically, the model abstraction can be demonstrated by considering the divergence minimization between the softened probabilities produced by the teacher model and the student model. The objective is to minimize the discrepancy between the two models' predictions for each training example.

Let's denote the softened probabilities produced by the teacher model as $P_t(y|x)$ and the softened probabilities produced by the student model as $P_s(y|x)$. These probabilities represent the distribution over the possible output labels $y$ given the input $x$.

The loss function of knowledge distillation typically measures the discrepancy between these two distributions. One common approach is the cross-entropy loss:

$$\mathscr{L} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{y} P_t(y|x^{(n)}) \log P_s(y|x^{(n)}) \tag{3.6}$$

Minimizing this loss function encourages the student model to match the teacher model's predictions, abstracting the crucial knowledge contained in the teacher model.

To prove the model abstraction, we can show that minimizing the knowledge distillation loss leads to the student model learning the teacher model's predictions.

Let's consider the optimization objective:

$$\frac{\partial \mathscr{L}}{\partial \theta_s} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{y} P_t(y|x^{(n)}) \frac{1}{P_s(y|x^{(n)})} \frac{\partial P_s(y|x^{(n)})}{\partial \theta_s} = 0 \tag{3.7}$$

By taking the derivative of the loss function with respect to the student model's parameters $\theta_s$ and setting it to zero, we can find the optimal parameters that minimize the

loss.

$$\sum_{n=1}^{N} \sum_{y} P_t(y|x^{(n)}) \frac{\partial \log P_s(y|x^{(n)})}{\partial \theta_s} = 0 \tag{3.8}$$

Simplifying the equation, we get:

$$\log P_s(y|x^{(n)}) \quad \text{(logits of the student model)} \tag{3.9}$$

Since $\log P_s(y|x^{(n)})$ represents the logits (output before softmax) of the student model, this equation indicates that the weighted average of the gradients of the logits with respect to the teacher model's predictions is zero. The gradients of the logits align with the teacher model's predictions.

Therefore, by minimizing the knowledge distillation loss, the student model effectively learns the teacher model's predictions, abstracting the important knowledge contained in the teacher model.

#### 3.3.5.2 Use part information to train global model

To analyse how the defense method affects the attack, we should understand how the attack works. To get the raw data by gradient inversion attack, let us first analyze the question if data $x \in \mathbf{R}^n$ can be recovered from its gradient $\nabla_\theta L_\theta(x, y) \in \mathbf{R}^p$ analytically. $x$ and $\nabla_\theta L_\theta(x, y)$ have distinct dimensions, reconstruction quality is surely a question of the number of parameters $p$ versus input pixels $n$. If $p < n$, then reconstruction is at least as difficult as image recovery from incomplete data [11], but even when $p > n$, which we would expect in most computer vision applications, the difficulty of regularized ‚Äú√∫inversion" of $\nabla_\theta L_\theta$ relates to the non-linearity of the gradient operator as well as its conditioning [5].

There are several preconditions that need to be met before we can begin making any deductions. The input to a fully-connected layer can always be estimated analytically from the parameter gradients, regardless of where the layer is located in a neural network. This is true even if the layer is not completely linked. A single input to a fully-connected network can always be reconstructed analytically without the need to solve an optimisation problem [27]. In particular, the analytic reconstruction is not dependent on the specific types of layers that come before or after the fully connected layer. The proof can also be described as that in the situation of a neural network with a biased fully-connected layer is preceded by only (potentially unbiased) fully-connected layers. Additionally, suppose that for any of these completely connected layers, the derivative of

the loss $L$ with respect to the layer's output has at least one non-zero element. The input to the network may then be uniquely recovered from the gradients of the network.

Nonetheless, this argument must satisfy two significant assumptions [27]: the first is that the BatchNorm statistics are known, which is a procedure for training models that normalises the inputs to each mini-batch. As described in the threat model section, knowledge of BatchNorm would enable an attacker to use the same batch normalisation employed by a private batch to his recovered batch, resulting in a more accurate reconstruction.

Refer to statistical machine unlearning method[12], our strategy employs part of statistical information to train models, which conceals certain information in the BatchNorm component to impact the reconstruction results. The second premise is knowledge of or the ability to infer private labels. Which is also utilised to enhance the effectiveness of attacks. In our method, all private labels are extracted to features, and the absence of private labels is also a reason for superior performance in the face of gradient inversion attacks.

The original purpose of the statistical machine unlearning method is to transform data into statistical information to enhance the efficiency of the retraining process. In our approach, statistical information serves as a demarcation between the original data and the training data. This segregation restricts attackers from deducing the data from information with weaker correlations.

In a typical FL system, the upload of a subset of local model parameters for training the global model involves a trade-off between model privacy and training efficiency. First, for the privacy of model.To quantitatively analyze model privacy, we can consider the concept of differential privacy. Let's assume that the parameter update operations of each participant on their local data adhere to $\epsilon - differential$ privacy, where $\epsilon$ represents the privacy budget for differential privacy. The impact of uploading a subset of local model parameters on model privacy can be expressed as follows:

Let $\epsilon_{global}$ denote the differential privacy budget of the global model, and $\epsilon_{local}$ denote the differential privacy budget of the local model. Uploading the parameters of the last two layers affects $\epsilon_{global}$ while keeping $\epsilon_{local}$ constant:$\epsilon_{global} = \epsilon_{local} - \epsilon_{upload}$. Here, the magnitude of $\epsilon_{upload}$ is contingent on the number of uploaded parameters, the method used for introducing differential privacy noise, and the knowledge level of potential adversaries. Uploading fewer parameters typically results in a larger $\epsilon_{upload}$, thereby increasing the differential privacy level of the global model and reducing $\epsilon_{global}$.

Uploading the part layers of the local model parameters impacts training efficiency in

terms of reduced communication and computation costs. Specifically, if the dimension of the global model's parameters is denoted as $P$, and the number of uploaded parameters is denoted as $M$: Reduced communication overhead can be expressed as $(P - M)/P$. Reduction in aggregation computation overhead depends on specific considerations but is typically linearly related to the number of uploaded parameters, resulting in lowered aggregation computational complexity. Uploading fewer parameters reduces communication and computation costs, thereby enhancing training efficiency.

## 3.4 Experiment

### 3.4.1 Experiment Setup

#### 3.4.1.1 Dataset

**CIFAR-10:** CIFAR-10 dataset is a popular benchmark in computer vision and machine learning. It contains 60,000 small 32x32 pixel images divided into 10 classes, making it useful for testing image classification algorithms. This dataset covers a wide range of objects and scenes and is known for its diversity and balanced class distribution. It's widely used for training and evaluating machine learning models, particularly in deep learning research.

**MNIST:** MNIST is a widely used dataset in machine learning and computer vision, featuring 28x28 pixel grayscale images of handwritten digits (0-9). It has 10 classes and is commonly employed for digit recognition and OCR tasks.

**Adult:** The Adult dataset, also known as the Census Income dataset, contains demographic information about individuals alongside a target variable indicating income levels. It is widely used in machine learning for binary classification tasks, particularly in predicting whether an individual's income exceeds $50,000 annually. The dataset serves as a valuable resource for exploring socioeconomic trends and patterns, and it is commonly employed in social science research and demographic analysis.

We train CIFAR-10 on ResNet-18 architecture. We train MNIST with a simple 6-layer ConvNet model, the simple ConvNet does not contain BatchNorm layers. And we use a simple 3-layer ConvNet model to train Adult dataset.

#### 3.4.1.2 Baseline of Defense

We have chosen three cutting-edge defense methods as the foundational strategies for our research. These approaches have exhibited outstanding efficacy in protecting against

Table 3.1: Performance on CIFAR-10

| | **None** | **GradPrune**($p$) | | | | | | **MixUp** ($k$) | | **Intra-InstaHide** ($k$) | | **Our method** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parameter | - | 0.5 | 0.7 | 0.9 | 0.95 | 0.99 | 0.999 | 4 | 6 | 4 | 6 | - |
| Test Acc. | 93.37 | 93.19 | 93.01 | 90.57 | 89.92 | 88.61 | 83.58 | 92.31 | 90.41 | 90.04 | 88.20 | 93.33 |
| Time(train) | 1× | 1.04× | | | | | | 1.06× | | 1.06× | | 0.98× |
| Attack batch size = 1 | | | | | | | | | | | | |
| Avg. LPIPS | 0.19 | 0.19 | 0.22 | 0.35 | 0.42 | 0.52 | 0.52 | 0.34 | 0.46 | 0.58 | 0.61 | 0.78 |
| Best LPIPS | 0.02 | 0.02 | 0.05 | 0.14 | 0.22 | 0.32 | 0.36 | 0.12 | 0.25 | 0.41 | 0.42 | 0.62 |
| (LPIPS std.) | 0.16 | 0.17 | 0.16 | 0.13 | 0.11 | 0.08 | 0.06 | 0.08 | 0.07 | 0.06 | 0.09 | 0.11 |
| Attack batch size = 16 | | | | | | | | | | | | |
| Avg. LPIPS | 0.45 | 0.46 | 0.47 | 0.51 | 0.55 | 0.58 | 0.61 | 0.34 | 0.31 | 0.62 | 0.63 | 0.76 |
| Best LPIPS | 0.18 | 0.19 | 0.19 | 0.31 | 0.43 | 0.47 | 0.51 | 0.11 | 0.13 | 0.41 | 0.44 | 0.66 |
| (LPIPS std.) | 0.12 | 0.12 | 0.11 | 0.07 | 0.05 | 0.04 | 0.03 | 0.09 | 0.09 | 0.08 | 0.08 | 0.10 |
| Attack batch size = 32 | | | | | | | | | | | | |
| Avg. LPIPS | 0.45 | 0.46 | 0.48 | 0.52 | 0.54 | 0.58 | 0.63 | 0.50 | 0.49 | 0.69 | 0.69 | 0.88 |
| Best LPIPS | 0.18 | 0.18 | 0.22 | 0.31 | 0.43 | 0.48 | 0.54 | 0.31 | 0.28 | 0.56 | 0.56 | 0.74 |
| (LPIPS std.) | 0.11 | 0.11 | 0.09 | 0.07 | 0.05 | 0.04 | 0.04 | 0.10 | 0.10 | 0.06 | 0.07 | 0.10 |

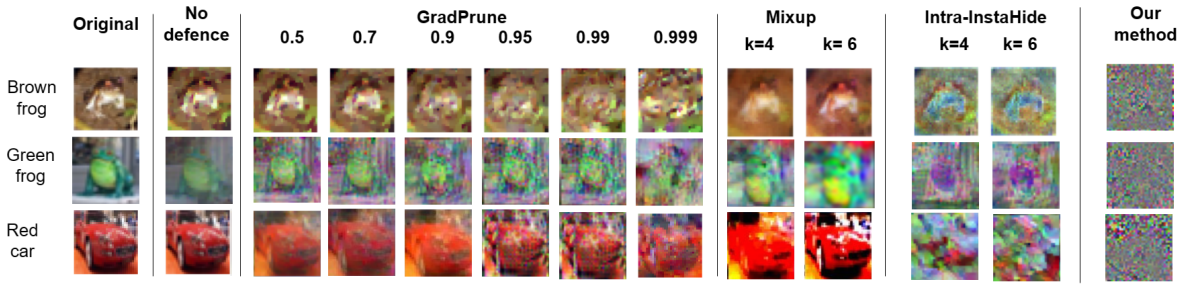


Figure 3.2: Reconstruction Results

a wide range of gradient inversion attacks, positioning them as robust contenders for our investigation.

**GradPrune** (gradient pruning)[136]: gradient pruning set gradients of small magnitudes to zero. We vary the pruning ratio $p$ in $0.5, 0.7, 0.9, 0.95, 0.99, 0.999$.

**MixUp**[125]: MixUp encodes a private image by linearly combining it with $k-1$ other images from the training set. Following [48], The value of $k$ is from $4, 6$, and set the upper bound of a single coefficient to 0.65 (coefficients sum to 1).

**Intra-InstaHide**[48]: InstaHide [48] proposes two versions: Inter-InstaHideand Intra-InstaHide. The only difference is that at the mixup step, Inter-Instahide mixes up an image with images from a public dataset, whereas Intra-InstaHide only mixes with private images. Both versions apply a random sign-flipping pattern on each mixed image. Similar to the evaluation of MixUp, we vary $k$ in $4, 6$, and set the upper bound of a single coefficient to 0.65. Note that InstaHide flips signs of pixels in the image, which destroys the total variation prior. However, the absolute value of adjacent pixels should still be

Table 3.2: Performance on MNIST

| | **None** | **GradPrune**($p$) | | | | | | **MixUp** ($k$) | | **Intra-InstaHide** ($k$) | | **Our method** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parameter | - | 0.5 | 0.7 | 0.9 | 0.95 | 0.99 | 0.999 | 4 | 6 | 4 | 6 | - |
| Test Acc. | 98.89 | 96.11 | 96.04 | 93.25 | 90.26 | 90.02 | 89.41 | 91.25 | 90.56 | 89.42 | 88.50 | 98.45 |
| Time(train) | 1× | 1.04× | | | | | | 1.06× | | 1.06× | | 0.99× |
| Attack batch size = 1 | | | | | | | | | | | | |
| Avg. LPIPS | 0.20 | 0.21 | 0.25 | 0.34 | 0.44 | 0.52 | 0.53 | 0.34 | 0.47 | 0.58 | 0.60 | 0.76 |
| Best LPIPS | 0.02 | 0.04 | 0.06 | 0.14 | 0.23 | 0.31 | 0.34 | 0.11 | 0.25 | 0.41 | 0.44 | 0.63 |
| (LPIPS std.) | 0.15 | 0.16 | 0.15 | 0.13 | 0.10 | 0.09 | 0.07 | 0.08 | 0.06 | 0.05 | 0.09 | 0.10 |
| Attack batch size = 16 | | | | | | | | | | | | |
| Avg. LPIPS | 0.45 | 0.45 | 0.49 | 0.52 | 0.55 | 0.61 | 0.62 | 0.34 | 0.36 | 0.62 | 0.63 | 0.77 |
| Best LPIPS | 0.17 | 0.19 | 0.23 | 0.32 | 0.44 | 0.46 | 0.51 | 0.12 | 0.12 | 0.42 | 0.46 | 0.67 |
| (LPIPS std.) | 0.11 | 0.11 | 0.11 | 0.08 | 0.04 | 0.04 | 0.05 | 0.12 | 0.09 | 0.08 | 0.09 | 0.15 |
| Attack batch size = 32 | | | | | | | | | | | | |
| Avg. LPIPS | 0.45 | 0.46 | 0.50 | 0.51 | 0.53 | 0.55 | 0.66 | 0.54 | 0.52 | 0.66 | 0.72 | 0.84 |
| Best LPIPS | 0.17 | 0.19 | 0.24 | 0.34 | 0.44 | 0.49 | 0.56 | 0.34 | 0.31 | 0.54 | 0.56 | 0.78 |
| (LPIPS std.) | 0.10 | 0.12 | 0.09 | 0.06 | 0.04 | 0.04 | 0.06 | 0.12 | 0.07 | 0.05 | 0.07 | 0.15 |

close. Therefore, for the InstaHide defense, we apply the total variation regularizer on $|x|$, i.e. taking the absolute value of each pixel in the reconstruction.

**Our method**: The training process begins with initializing the student models on the client devices. Local training is performed iteratively, where each client trains its model using its local dataset. During each training iteration, the selected subset of parameters is uploaded to the server for aggregation, without disclosing specific details of the local data. The aggregated parameters are then sent back to the respective clients for updating their models based on the distillation loss between the student and teacher predictions.

#### 3.4.1.3 Evaluation metrics

LPIPS (Learned Perceptual Image Patch Similarity): LPIPS is a perceptual similarity metric that measures the perceptual difference between two images. It is commonly used to evaluate the quality or similarity of generated or reconstructed images compared to the original images. LPIPS takes into account various low-level visual features and captures the human perception of image similarity. It computes the distance between the feature representations of the images using a deep neural network, such as a pre-trained VGG network. Lower LPIPS scores indicate higher perceptual similarity between images.

Model Accuracy Metric: Model accuracy is a commonly used evaluation metric to measure the performance of machine learning models. It quantifies the model predicts or classifies data correctly. The accuracy metric is often calculated as the percentage of correctly predicted samples from the total number of samples in the dataset. Higher model accuracy indicates better performance in capturing the patterns and characteristics of

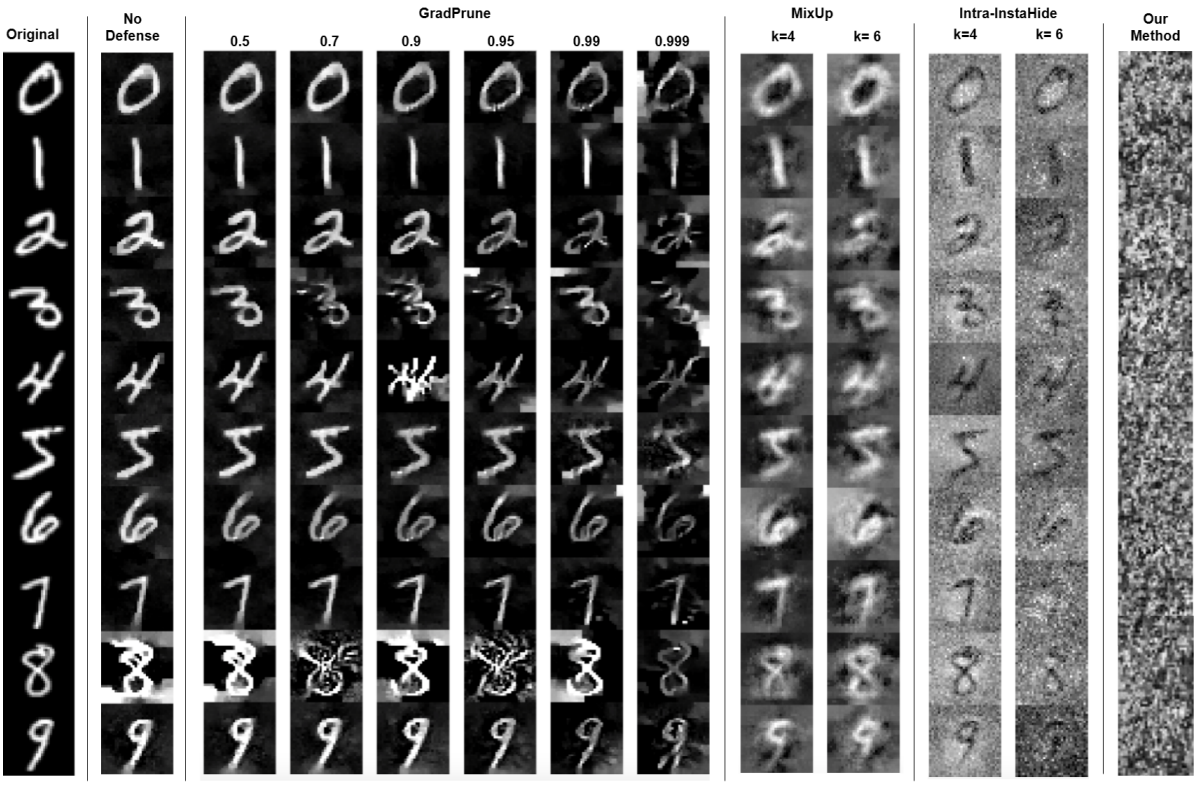


Figure 3.3: Performance on MNIST Dataset

the data.

#### 3.4.1.4 Attack setting

In order to test the effectiveness of the attack, we utilize a subset of 50 CIFAR-10 pictures. Take note that there is an additional step involved in attacking MixUp and InstaHide, which is to decode private photographs from encoded images. We use the attack described in[13] as the decode step. In this attack, the adversary is required to eavesdrop on training over a total of $T$ epochs, rather than on a single training step. In our assessment, we will assume that $T = 20$. For the sake of determining the top limit of privacy leaking, we also give the attacker the most powerful possible power for the decode phase. We make the assumption that the attacker is aware of the following given a MixUp or Intra InstaHide picture that encodes $k$ private images:

1. The indices of the $k$ photos contained within the private dataset. In the context of the attack described by [13], the attacker would train a neural network to assess the similarity of encodings and subsequently employ a combinatorial algorithm to solve an approximation of this mapping, assuming a scenario based on real-world conditions.

2. The mixing coefficients for each of the $k$ private images individually. In a true FL environment, you would not have access to this information.

Metrics to evaluate the quality of the reconstruction. We illustrate reconstructions achieved under varied defenses. Following the methodology described in [117], we additionally make use of the learnt perceptual image patch similarity (LPIPS) score described in [125] to determine the degree to which the reconstructed pictures differ from the original ones: Higher numbers indicate a greater degree of mismatch (less privacy leakage).

Part of the attack settings and experiment data is from [47], Which has a comprehensive evaluation of attack and defense methods on gradient inversion attacks. Our baseline methods are also from this paper.

Reconstruction results under different defenses with batch sizes being 1, 16 and 32. Larger batch size usually has better performance of attack, as a result, figure 3.4 shows the attacks with or without defense methods when the batch size is 1. Gradient pruning has better performance when the prune ratio is close to 1. Mixup and Intra-InstaHide also make the image hard to recognize, and our method makes the reconstruction unrecognizable to humans (the last column).

### 3.4.2 Results

When the batch size is set to 1, the attack stands a high likelihood of successfully reconstructing the image from the gradient, even in the absence of any defensive measures. However, as the batch size is increased, the task of successful data recovery becomes more challenging. Nevertheless, the recovered images closely resemble those that were originally lost in terms of aesthetics.

In the second baseline, the Gradient pruning (GradPrune). The findings indicate that an increase in the pruning ratio $p$ leads to an increase in the number of artefacts present in the reconstructions. However, even when the pruning ratio is set to 0.9, the reconstructions may still be recognised; as a result, the proposal made in the past by [136] to use a value of 0.7 is no longer secure against the most recent attack methods. According to the studies in [47], in order to defend CIFAR-10 from the most severe attack using gradient pruning, the pruning ratio may need to be more than or equal to 0.999. A decrease in accuracy of around 10% would be the result of adopting such a high pruning ratio as a trade-off.

MixUp adds a little more complex computational burden to the training process. MixUp with $k = 4$ only has a marginal influence on test accuracy (2%), but this is not enough to protect against the gradient inversion attack. The leakage can be partially reduced by increasing $k$ from 4 to 6; nevertheless, the reconstruction can still be easily recognised even after this change. It seems from this that MixUp by itself may not be effective protection against the most recent gradient inversion attack.

When compared with MixUp, Intra-InstaHide with $k$ equal to 4 suffers from an additional accuracy loss of 2%. However it achieves better defense performance: when the batch size is 32, there are obvious artifacts and color shift in the reconstruction.

In our approach, the simplification of the training process will result in a reduction in the amount of time needed for training, while maintaining an accuracy that is sufficient to identify figures and a reconstruction that is no longer recognizable by the human eye. Figures 3.2 show the reconstruction performance of each defense method under batch size 1. For GradPrune, a large value of the prune ratio means better performance of defense on two different datasets. And Intra-InstaHide also has better performance than Mixup. Moreover, our method has better performance than other methods. Figure 3.6 shows that our method is similar to the baseline without any defense method in efficiency and accuracy. With the epoch increase, our method fast convergence. The slowest method is Intra-InstaHide, this approach needs to encode images during the process which will cost more time.

Table 3.3 shows the performances on Adult dataset. Figure 3.4 and Figure 3.5 show the recovery performance on no defense, GradPrune method and our method.

Table 3.3: Performance on Adult dataset

| | **None** | **GradPrune**($p$) | | | | | | **MixUp** ($k$) | | **Intra-InstaHide** ($k$) | | **Our method** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Param. | - | 0.5 | 0.7 | 0.9 | 0.95 | 0.99 | 0.999 | 4 | 6 | 4 | 6 | - |
| Test Acc. (%) | 85.45 | 85.32 | 84.84 | 82.42 | 80.33 | 79.42 | 77.76 | 80.52 | 80.2 | 78.64 | 77.56 | 85.34 |
| Time (train) | 1× | 1.01× | | | | | | 1.03× | | 1.03× | | 0.99× |

In knowledge distillation, there are several hyperparameters that can significantly influence the effectiveness of the distillation process and the performance of the student model. Here are some important hyperparameters and their influence: Temperature ($T$): The temperature parameter controls the softening of the teacher model's predictions. Higher temperatures ($> 1$) make the probabilities more uniform and emphasize the relative differences between classes, while lower temperatures ($< 1$) sharpen the probabilities and focus on the most probable class. A higher temperature can encourage exploration during training, while a lower temperature can lead to more confident predic-

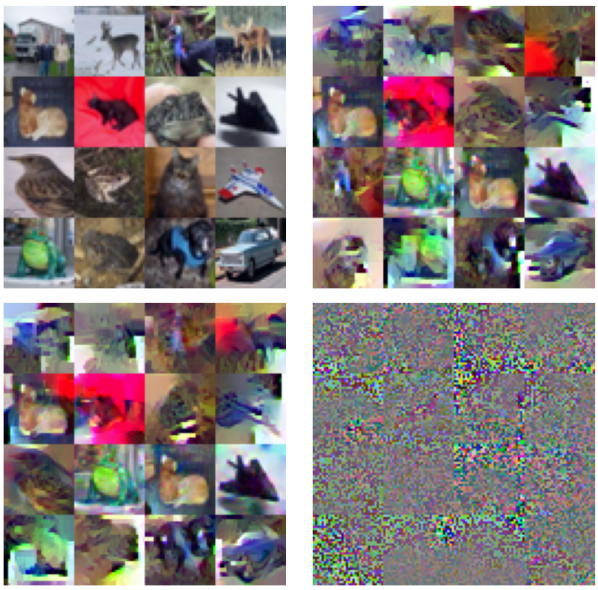

Figure 3.4: Comparison of No Defense and Defense Methods

Figure 3.5: Comparison of No Defense and Defense Methods

tions. Generally, higher temperatures can result in a smoother optimization landscape and better knowledge transfer.

Distillation Loss Weight ($\lambda$): The distillation loss weight determines the balance between the knowledge distillation loss and the original training loss. It controls the relative importance of matching the teacher's predictions compared to fitting the ground

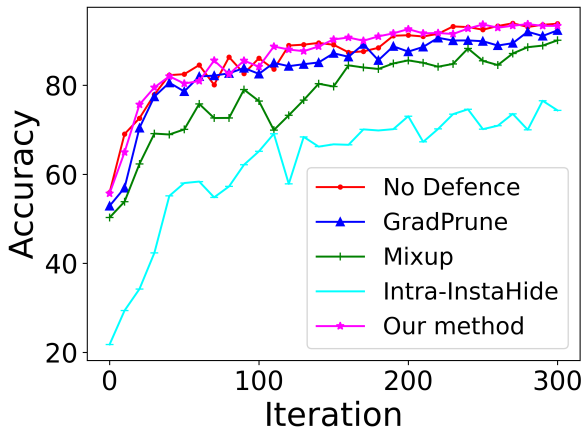


Figure 3.6: Accuracy Changes with Iterations

truth labels. A higher weight on the distillation loss places more emphasis on transferring knowledge from the teacher model, potentially at the cost of fitting the ground truth labels. It is important to find an appropriate balance between the two losses to ensure effective knowledge transfer without sacrificing performance.

Figure 3.7 shows the accuracy change with temperature.

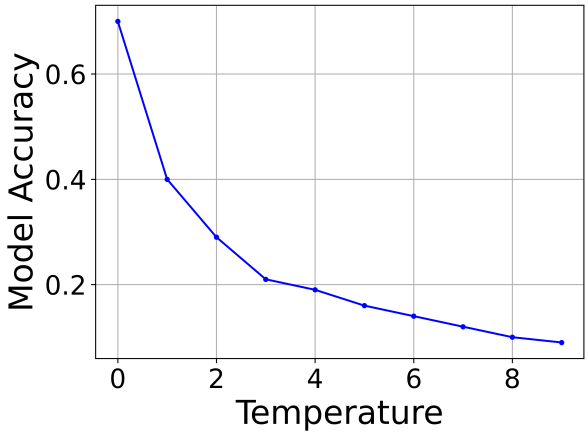


Figure 3.7: Accuracy Changes with Temperature

## 3.5 Ablation Study

This section evaluates the contribution of the three main components used in the proposed defence framework, namely statistical aggregation (SA), knowledge distillation

(KD) and noise injection (NI). The goal is to understand how each module affects both predictive performance and resistance to reconstruction attacks. We conduct a controlled study by progressively enabling these components and measuring test accuracy, reconstruction quality and attack success rate under a standard gradient inversion attack. Reconstruction quality is reported using PSNR, where lower values indicate weaker ability to recover meaningful input features.

Table 3.4 summarises the results. The baseline system without defense reaches a PSNR of 29.4 dB, which allows the recovered image to retain clear structure, leading to an attack success rate of 92.0 percent. Introducing statistical aggregation significantly reduces the information available to the attacker by replacing full gradients with coarse summary statistics. This alone reduces the PSNR to 18.6 dB and decreases the attack success rate to 58.3 percent.

Knowledge distillation also provides meaningful protection by smoothing the representation space and reducing the dependence on individual samples. When applied alone, it lowers the PSNR to 21.2 dB. Noise injection adds numerical uncertainty to the aggregated statistics, further diminishing the consistency needed for reconstruction. With noise as the only defence, the PSNR decreases to 17.4 dB and the attack success rate falls to 52.1 percent.

Stronger protection emerges when components are combined. When SA and KD are used together, the PSNR falls to 14.8 dB and the attack success rate decreases to 41.0 percent. Combining SA with NI provides even stronger protection, reducing the PSNR to 12.3 dB. The complete system that integrates all three components achieves the lowest PSNR of 9.7 dB and an attack success rate of only 22.5 percent, while maintaining a test accuracy comparable to other configurations.

These results show that although each component contributes to model robustness, the full combination delivers the strongest privacy protection with minimal impact on predictive performance. The observed trend supports the design choice of using multiple complementary mechanisms to limit the ability of reconstruction attacks to exploit fine level information hidden in model updates.

## 3.6 Conclusion

In summary, this paper tackles the pressing issue of defending against gradient inversion attacks in FL (FL), with a keen focus on preserving high model accuracy. FL offers a valuable framework for privacy-sensitive learning, but it remains vulnerable to

| Method | Test accuracy (%) | PSNR (dB) | Attack success rate (%) |
|---|---|---|---|
| No defence | 86.1 | 29.4 | 92.0 |
| SA only | 85.7 | 18.6 | 58.3 |
| KD only | 86.4 | 21.2 | 66.7 |
| NI only | 84.9 | 17.4 | 52.1 |
| SA + KD | 86.0 | 14.8 | 41.0 |
| SA + NI | 85.1 | 12.3 | 36.4 |
| Full model (SA + KD + NI) | 85.8 | 9.7 | 22.5 |

Table 3.4: Ablation study of individual components and their combinations. Lower PSNR and lower attack success rate indicate stronger protection.

attacks where adversaries can reconstruct clients' training data from shared gradients. Traditional defense mechanisms often involve perturbing training data or gradients, a strategy that unfortunately tends to compromise model accuracy.

To address this challenge, our paper introduces a novel defense method rooted in the concept of machine unlearning. This approach involves training local client models using statistical data information, rather than direct training data. By incorporating knowledge distillation, we effectively create a buffer between the original model and a derived student model, managing this without significant losses in efficiency. Subsequently, only selective information from the student model is utilized for updating the global model, adding a second layer of separation between the original training data and the global model updates. This innovative method not only strengthens data privacy in FL but also maintains the accuracy and effectiveness of the global model.

Our method rooted in machine unlearning can be further developed to detect and mitigate poisoning attacks orchestrated by malicious clients. Techniques such as anomaly detection algorithms and outlier rejection mechanisms can be integrated into our approach to identify and neutralize the impact of poisoned data injected by adversaries. Additionally, incorporating robust aggregation methods can help in aggregating model updates from trustworthy clients while filtering out potentially malicious contributions.

To defend against membership inference attacks, our method can be enhanced to preserve the privacy of individual participants in FL systems. Techniques such as differential privacy mechanisms and FL with cryptographic privacy protections can be integrated to prevent adversaries from inferring sensitive information about specific data contributors. By incorporating privacy-preserving algorithms, our approach can ensure that the global model updates released to clients do not reveal any information that could be exploited to infer membership status.

For future work, the focus could be on integrating advanced machine learning models to improve FL systems' robustness against inversion attacks. Exploring innovative GAN architectures could simulate and counteract potential threats, while ensuring model fairness and consistency across diverse datasets. Efficiency and scalability in larger networks, alongside applying these methods in varied domains like healthcare, are vital. Long-term studies on the impact of these strategies on learning processes in FL will provide deeper insights into their efficacy and evolution.

CHAPTER 4

# HIDDEN THREATS IN FEDERATED UNLEARNING: CAMOUFLAGED POISONING ATTACKS AND THEIR UNLEARNING CONSEQUENCES

This chapter reveals vulnerabilities in federated unlearning by studying camouflaged poisoning attacks. It formalizes the problem, presents attack methodology and theoretical insights, and demonstrates the consequences through empirical evaluation.

## 4.1 Introduction

Building upon the core principles of machine unlearning and the concept of right to be forgotten, federated unlearning can generally be categorized into several main types based on what aspect of the federated system is being unlearned. Class unlearning targets an entire class label across all clients [108]. When a specific category need to be excluded or corrected in the model, this type of unlearning works well. Client unlearning is a special type of unlearning in FL, it removes all the contributions of particular clients [25, 40], reflecting scenarios where a participant withdraws completely from the federated system. There is also one kind of unlearning that aims to forget specific features or sensitive attributes called feature unlearning, which ensures that certain types of information are no longer part of the model's knowledge [34] [88]. Data unlearning involves the removal of specific data points, allowing the model to adjust as if the samples had never participated in [69] [123]. This type of unlearning is necessary for

privacy or regulatory compliance. Our method mainly focus on data unlearning. Our goal is to leverage the unlearning of specific data points, which we called camouflaged data to trigger the adversarial effects of the poisoned data that remain in the model. [101]

While pioneering in addressing privacy and data autonomy, federated unlearning also introduces security and privacy challenges. The process of unlearning inherently modifies the model, potentially exposing it to new forms of adversarial attacks, where malicious clients exploit the unlearning mechanism to degrade model performance or induce computational inefficiencies [17] [121]. In addition, ensuring that the data is thoroughly and verifiably removed complicates the task, raising concerns about the influence of residual data and the adherence to strict privacy regulations. [25] [61] Additionally, the balance between maintaining model performance and achieving effective unlearning poses significant challenges, underscoring the complexity of designing machine learning systems that are both robust and respectful of user privacy in a dynamic data landscape [111].

To exploit these vulnerabilities, we propose a novel attack, called camouflage poisoning attack, specifically targeting the client level to enable the selective unlearning of particular training data points to degrade the performance of the model while avoiding being detected. Current attack methods face several limitations: they are often detectable during training due to significant changes in model performance, lack persistence as their effects become apparent throughout the training process, and may depend on test-time triggers, which restricts their applicability and effectiveness. In contrast, our camouflaged data poisoning attack overcomes these limitations by introducing a camouflage set that neutralizes the poison's effect during training, rendering the attack more covert. It specifically leverages the unlearning process to activate the latent adversarial influence once the camouflage is removed, unlike existing methods that continuously affect the model. Furthermore, it does not rely on specific test-time triggers, making it more adaptable and effective.

One of the recent research also proposed a new camouflaged attack method called backdoor attacks (UBA-Inf) [49], The method focuses on embedding dormant backdoors in machine learning models during training, which can be activated later through specific unlearning requests. Compared to methods like UBA-Inf, which rely on test-time backdoor triggers in a centralized MLaaS environment, our approach operates in the FL context and is uniquely tailored to the challenges of decentralized training, such as non-IID data and distributed aggregation. While UBA-Inf exploits influence-driven camouflage for stealth, it is confined to backdoor scenarios, whereas our attack directly

targets the global model's accuracy and misclassification rates, creating broader impacts beyond backdoor-specific use cases. Moreover, UBA-Inf requires persistent backdoor activation mechanisms, whereas our method achieves stealth and effectiveness by fully exposing the poisoned influence only after unlearning the camouflage set.

Our attack has two steps. Initially, prior to the model's training, the malicious clients introduce a crafted set of data points into their own training dataset, comprising both a poison set and a camouflage set. The objective is to ensure that the model has similar behavior no matter whether there are some malicious clients or not. The second phase is launched by attackers, the malicious clients request for unlearning operation after training phrase. After unlearning the camouflaged part, the attack is fully exposed. Fig.4.1 shows an illustration of our method. The performance of models will be poor because of the attack. The main purpose of our attack method is to misclassify one particular point in the training set and decrease the accuracy of the model. Our contributions are as follows.

- We present the concept of camouflaged data poisoning attacks in FL, unveiling a novel attack paradigm within dynamic environments, particularly in scenarios involving federated unlearning.
- We implement these attacks as a targeted poisoning attack in the FL framework, employing an algorithm based on the gradient matching technique. To simulate the behavior of the model to hide the attacks, we devised the camouflage set by creating a new series of points that counteract the influence of the poison set.
- We evaluated the effectiveness of these attacks on a wide range of model architectures and datasets.

## 4.2 Problem Statement

We define a data sample as $(x, y)$, where $x$ represents the feature vector and $y$ is the class label. In FL, a model is represented by a function $f(\cdot)$, which maps the input $x$ to an output $Y = f(x)$, a vector of probabilities across classes. Each entry $y_i$ in $Y$ indicates the posterior probability of the model assigning $x$ to class $c_i \in C$, where $C$ is the set of all possible classes.

Let $D$ represent the training dataset used by the clients, and let $D_u \subseteq D$ be the subset of camouflage data intended for unlearning. We denote the original global model

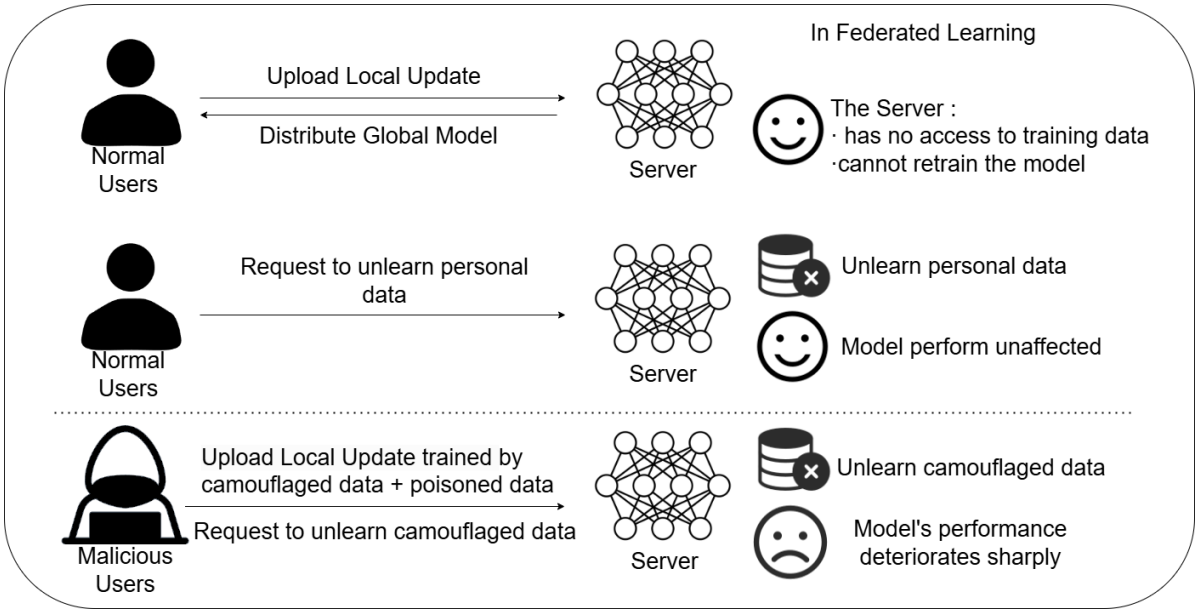


Figure 4.1: Illustration of Poisoning Unlearning Attack in FL

parameters trained on the full dataset as $\theta_{cpc}$, and the parameters after unlearning the camouflage data as $\theta_u$.

The adversarial clients introduce both poisoned and camouflage samples into their local datasets, which are used to train their local models. These models, when aggregated, form a global model ($\theta_{cpc}$) that performs well on validation data because the camouflage data neutralize the impact of poisoned data. Once the unlearning request for the camouflage data ($D_u$) is initiated, the model retrains, removing the camouflage data and revealing the previously hidden effects of the poisoned data. This leads to the global model misclassifying the targeted data points, as the poisoned data now influence the model's behavior, degrading its performance. The attacker's primary objective is to exploit this unlearning process to deliberately activate the effects of the poisoned data, causing misclassifications or a significant drop in overall accuracy, thereby compromising both the privacy and integrity of the FL system.

### 4.2.1 Threat Model

Assume a scenario where the cross-silo FL system has few clients and needs to aggregate all the updates from them. The threat model operates in an environment where adversarial clients are part of the FL system and can interact with the central server.

**Attacker's Goal:** The attacker's goal is to degrade the model's performance in the

FL system by injecting maliciously crafted data that appears benign during the initial training phase, allowing it to blend in undetected. However, once the camouflage data is removed during the unlearning process, the hidden toxic data reveals its adversarial effects, leading to reduced accuracy in the model.

**Attacker's Knowledge:** The attacker possesses knowledge about the FL system's architecture, communication protocols, and update aggregation mechanisms.

**Attacker's Capability:** The attacker can generate updates that strategically contribute to the model training while remaining indistinguishable from benign updates.

The attacker has the ability to submit unlearning requests to exploit vulnerabilities in the unlearning process.

The attacker may collaborate with other adversarial clients to amplify the impact and reduce the risk of detection.

The attacker can craft updates that mimic the statistical properties of normal updates, ensuring they contribute to a seemingly accurate global model.

## 4.3 Methodology

This section outlines the method for implementing a Camouflaged Poisoning Attack in a FL environment, highlighting the key phases: adversarial poisoning, camouflage integration, and activation through unlearning requests.

### 4.3.1 Camouflage Poisoning Attack

The core idea of our camouflaged poisoning attack is to inject adversarial updates into the FL process in such a way that they remain hidden during training. By masking poisoned data points with carefully crafted camouflage data, the attack ensures the global model's performance does not degrade during training, but significant performance degradation is triggered when the unlearning process is initiated.

The workflow for the attack consists of three main phases: Adversarial Poisoning, Camouflage Data Integration and Activation via Unlearning which are displayed in Fig.4.2.

#### 4.3.1.1 Adversarial Poisoning Phase

In this phase, the attacker injects poisoned data into the FL process through selected client devices. The primary objective is to subtly alter the global model's parameters,

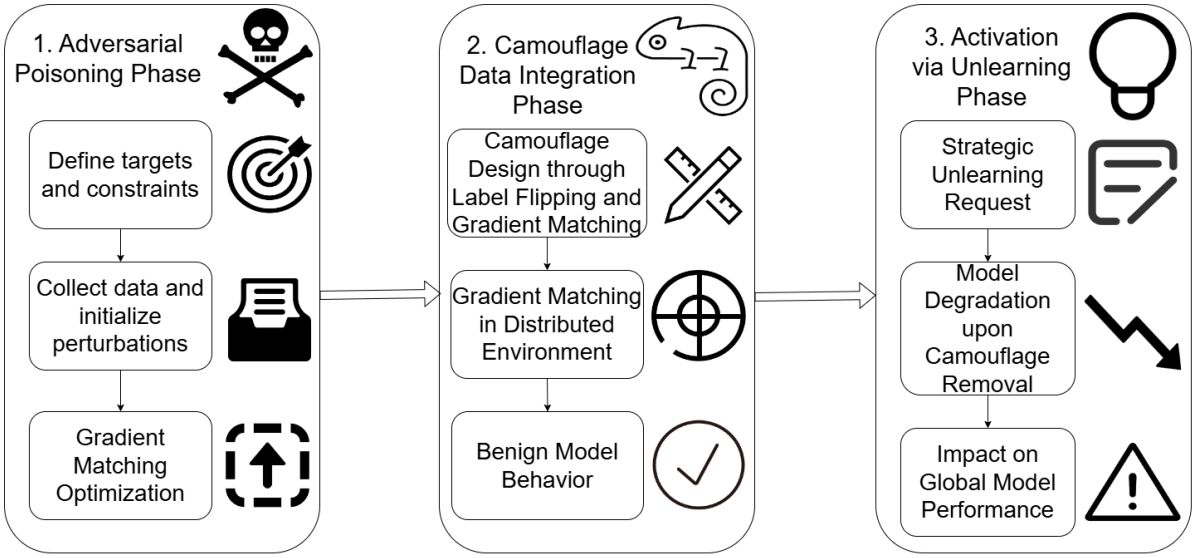


Figure 4.2: Workflow of Proposed Method.

leading to misclassifications or specific model behavior.

Target Definition and Perturbation Constraints: The attacker selects target classes and defines the type of perturbations (e.g., bounded by $l_\infty$ norm) to be applied to clean data points.

Data Collection and Perturbation Initialization: The attacker collects a set of images or data points corresponding to the target class. Initial adversarial perturbations are applied to these data points within the predefined budget. The perturbations are optimized over several iterations to ensure alignment between the gradients of poisoned data and the global model's objective.

Gradient Matching Optimization: Using an optimization algorithm, such as Adam, the attacker refines the perturbations to align the gradients of the poisoned data with those of normal data points.

#### 4.3.1.2 Camouflage Data Integration Phase

To ensure that the adversarial poisoning remains undetected during training, we introduce camouflage data. This data is carefully designed to neutralize the poisoned data's impact on the global model during the initial training phase.

Camouflage Design through Label Flipping and Gradient Matching: The attacker creates camouflage data points that either flip the labels of the poisoned data or align their gradients with clean data. For each poisoned data point $(x_i, y_i)$, a corresponding

camouflage point $(x_i, -y_i)$ is added to neutralize the overall gradient effect of the poisoned data.

Gradient Matching in a Distributed Environment: The attacker ensures that the gradients from both the poisoned and camouflage data are aligned across different clients. This is particularly effective in non-IID (non-independent and identically distributed) data settings, where data distributions may differ significantly among clients. By controlling the gradients of both sets, the attacker ensures that the global model remains accurate during the training phase.

Benign Model Behavior: Throughout training, the global model aggregates updates from both normal and adversarial clients. The camouflage data neutralizes the poisoned updates, allowing the model's performance on validation data to remain stable, which helps avoid detection.

#### 4.3.1.3 Activation via Unlearning Phase

The attack is fully realized when the attacker initiates an unlearning request to remove the camouflage data from the global model. This leverages the federated unlearning mechanism to undo the camouflage effects, exposing the underlying poisoned data.

Strategic Unlearning Request: The attacker issues an unlearning request to remove the camouflage data points from the global model. FL environments typically support such requests to comply with privacy regulations (e.g., GDPR). The attacker exploits this mechanism to remove the camouflage data and activate the latent effects of the poisoned data.

Model Degradation upon Camouflage Removal: Once the camouflage data is removed, the adversarial effects of the poisoned data become fully visible. The global model, which had relied on the balancing effect of the camouflage data, now experiences performance degradation.

Impact on Global Model Performance: In FL, the timing of removing camouflage updates is critical. When these updates are taken away, the global model loses their balancing effect, which exposes the influence of the poisoned data. This shifts the model's parameters toward the attacker's goal and results in a significant decline in performance.

### 4.3.2 Techniques for Generating Poisoned and Camouflage Data

Camouflaged poisoning attacks in machine learning are designed to subtly corrupt a model's training process. These attacks involve introducing 'poisoned' data points that,

when removed (unlearned), lead to significant misclassification errors in the model. The most crucial parts in our method is the generation of poisoned data and camouflage data. Therefore, considering the mechanisms and limitations of these attacks, we focus on two main strategies for generating such data, label flipping and gradient matching.

#### 4.3.2.1 Gradient Matching

Gradient matching is a key technique in camouflaged poisoning attacks. The attacker aligns the gradients of poisoned data points with those of clean data points, ensuring that poisoned updates are not detected during training. The optimization problem can be formalized as minimizing the cosine similarity loss between the gradients of poisoned data and the target clean data:

$$\phi(\Delta,\theta) = 1 - \frac{\left\langle \nabla_\theta \ell(f(x_{\text{target}};\theta),y_{\text{target}}), \sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta),y_i) \right\rangle}{\left\| \nabla_\theta \ell(f(x_{\text{target}};\theta),y_{\text{target}}) \right\| \left\| \sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta),y_i) \right\|}. \tag{4.1}$$

This loss function ensures that the gradients of the poisoned data align closely with those of the target data, effectively masking the adversarial influence during training. Gradient matching offers a more sophisticated approach, aiming to align the gradients of poisoned and clean data sets. The goal is to make training on the poisoned set mimic training on the clean set. This method relies on minimizing the cosine similarity loss between the gradients of the target and poisoned data. Although more complex than label flipping, gradient matching also faces challenges, particularly in its requirement to hold for all model parameters, a condition often intractable in practice.

Algorithm 4.1 generates poisoned data by iteratively optimizing perturbations applied to a selected set of data points, aligning their gradients with those of the target data. The optimization process ensures that the gradients of the poisoned data mimic those of clean data, making it difficult to detect during training. The algorithm uses techniques such as Adam optimization and multiple restarts to find the best perturbations that minimize the cosine similarity loss, effectively embedding adversarial influence into the model without compromising short-term performance.

#### 4.3.2.2 Label Flipping

Label flipping is another technique used in binary classification tasks, where the attacker flips the labels of poisoned data points. By introducing corresponding camouflage data points with flipped labels, the attacker neutralizes the effect of poisoned updates during

training. The total loss from the camouflage set can be shown to be equivalent to the clean set:

$$\sum_{(x,y)\in S_{\text{cpc}}} \ell(f(x,\theta),y) = \sum_{(x,y)\in S_{\text{cl}}} \ell(f(x,\theta),y) \tag{4.2}$$

This ensures that the overall training process remains unaffected by the poisoned updates until the unlearning process removes the camouflage. Besides, label flipping is a straightforward approach where the labels of certain data points in a binary classification problem are reversed. This method is effective for models trained with linear loss functions. For instance, in a model with loss function $\ell(f(x,\theta),y) = -yf(x,\theta)$, flipping the labels of poisoned data creates a camouflage set that neutralizes the poison's effect. Mathematically, this results in the loss from the camouflage set equating to the loss from a clean set, effectively rendering the poison harmless. However, label flipping has notable limitations. It's applicable only in binary classification scenarios with linear loss functions. Moreover, the method is detectable in validation phases as it involves assigning opposite labels to ground truth. This detectability makes it vulnerable to simple data purification techniques, where a data validator might remove data points with conflicting labels, rendering the attack ineffective.

Algorithm 4.2 is designed to create camouflage data that neutralizes the impact of poisoned data during the training phase, effectively hiding the adversarial updates. By aligning the gradients of the camouflage data with those of the poisoned data, the overall gradient update remains indistinguishable from normal client updates. The camouflage data is carefully optimized to ensure it only affects the global model when the poisoned data is activated via unlearning requests. The algorithm also uses iterative optimization and gradient matching techniques to minimize the adversarial footprint, ensuring the model behaves as expected during training until the unlearning process is triggered.

### 4.3.3 Comparison with Similar Method

The proposed method, camouflage poisoning attack shares conceptual similarities with the unlearning-activated backdoor attack (UBA-Inf) [49] in that both methods exploit vulnerabilities in the unlearning process to achieve adversarial objectives. However, the two approaches differ significantly in the scope, methodology, and impact.

UBA-Inf operates in a centralized Machine-Learning-as-a-Service (MLaaS) environment, where unlearning is used to activate dormant backdoors embedded during the

training phase. The attack relies on specific test-time triggers and carefully camouflaged poisoned data, aiming to achieve stealth and persistence in activating the backdoor. By contrast, our camouflage poisoning attack targets the FL (FL) framework, leveraging the decentralized nature of FL systems. Unlike UBA-Inf, which is designed specifically for backdoor attacks, our approach focuses on degrading the overall model performance and misclassifying targeted data points after unlearning.

Our method is uniquely tailored to the challenges of FL, such as complex data distributions and the decentralized aggregation of updates, which complicate the detection of adversarial behavior. Besides, instead of continuously affecting the model during training or relying on test-time triggers, our method strategically activates the adversarial influence only after the camouflage data is unlearned. This enhances the attack's stealth and adaptability. Unlike UBA-Inf, which relies on specific inputs to activate the backdoor, our method directly influences the model's performance post-unlearning, making it applicable to broader adversarial objectives. We employ gradient alignment strategies to craft poisoned and camouflage data that remain undetected during training, as well as label flipping to further mislead the model's decision boundaries.

In conclusion, both methods work well in their own environment. UBA-Inf demonstrates the potential risks associated with unlearning in centralized systems, emphasizing the need for robust defenses to address such sophisticated adversarial strategies while our camouflage poisoning attack exploit the unlearning process in FL systems, leveraging the decentralized and dynamic nature of FL to achieve stealth and effectiveness.

## An Intuitive Example of Post-Unlearning Amplification

Consider a malicious client that submits a mixed update of the form

$$g_{\text{cam}} = (1-\lambda)g_{\text{benign}} + \lambda g_{\text{attack}},$$

where $0 < \lambda \ll 1$. During normal FL training, the benign component dominates, causing the update to appear statistically similar to honest client gradients. As a result, standard aggregation rules accept it.

When unlearning is later performed to remove the benign client contributions, the global model update effectively becomes

$$g' = g_{\text{cam}} - g_{\text{benign}} \approx \lambda\, g_{\text{attack}}.$$

**Algorithm 4.1** Gradient Matching to generate poisoned data

**Require:** Clean network $f(\cdot;\theta_{\text{clean}})$ trained on uncorrupted base images $S_{\text{cl}}$, a target $(x_{\text{target}}, y_{\text{target}})$, and an adversarial label $y_{\text{adversarial}}$. Poison budget $P$, perturbation bound $\varepsilon$, number of restarts $R$, optimization steps $M$.

Collect a dataset $S_{\text{po}} \equiv \{(x_i, y_i)\}_{i=1}^{P}$ of $P$ images whose true label is $y_{\text{adversarial}}$.
**for** $r = 1$ to $R$ **do**
  Randomly initialize perturbations $\Delta$ s.t. $\|\Delta\|_\infty \leq \varepsilon$.
  **for** $k = 1$ to $M$ **do**
    Compute the loss $\phi(\Delta, \theta_{\text{clean}})$ as: $\phi(\Delta, \theta) = 1 - \frac{\langle \nabla_\theta \ell(f(x_{\text{target}};\theta), y_{\text{adversarial}}), \sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta), y_i) \rangle}{\|\nabla_\theta \ell(f(x_{\text{target}};\theta), y_{\text{adversarial}})\| \|\sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta), y_i)\|}$
    Update $\Delta$ using an Adam update to minimize $\phi$ and project onto the constraint set $\Gamma$.
  **end for**
  Amongst the $R$ restarts, choose the $\Delta^*$ with the smallest value of $\phi(\Delta^*, \theta_{\text{clean}})$.
**end for**
**return** the poisoned set $S_{\text{po}} \equiv \{x_i + \Delta_i^*, y_i\}_{i=1}^{P}$.

**Algorithm 4.2** Gradient Matching to generate camouflages

**Require:** Network $f(\cdot;\theta_{cp})$ trained on $S_{cl} + S_{po}$, the target $(x_{target}, y_{target})$, Camouflage budget $C$, perturbation bound $\varepsilon$, number of restarts $R$, optimization steps $M$

Collect a dataset $S_{ca} = \{x_j^{'}, y_j^{'}\}_{j=1}^{C}$ of $C$ many images whose true label is $y_{target}$.
**for** $r = 1$ to $R$ **do**
  Randomly initialize perturbations $\Delta$ s.t. $\|\Delta\|_\infty \leq \varepsilon$.
  **for** $k = 1$ to $M$ **do**
    Compute the loss $\phi(\Delta, \theta_{CP})$ with the function $\phi(\Delta, \theta) = \phi(\Delta, \theta) = 1 - \frac{\langle \nabla_\theta \ell(f(x_{\text{target}};\theta), y_{\text{adversarial}}), \sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta), y_i) \rangle}{\|\nabla_\theta \ell(f(x_{\text{target}};\theta), y_{\text{adversarial}})\| \|\sum_{i=1}^{P} \nabla_\theta \ell(f(x_i+\Delta_i;\theta), y_i)\|}$ using the base camouflage images in $S_{ca}$.
    Update $\Delta$ using an Adam update to minimize $\psi$, and project onto the constraint set $\Gamma$.
  **end for**
  Amongst the $R$ restarts, choose the $\Delta_*$ with the smallest value of $\psi(\Delta_*, \theta_{cp})$.
**end for**
Return the poisoned set $S_{ca} = \{x_j^{'} + \Delta_*, y_j^{'}\}_{j=1}^{C}$.

Although the original poisoning component was small, the removal of its benign cover shifts its relative influence, allowing the attack to manifest only *after* unlearning. This phenomenon requires two assumptions: (1) the malicious update includes a benign-like masking component during training, and (2) unlearning selectively removes contributions that were used as the mask.

## 4.4 Theoretical Analysis

In this section, we analyze the success of camouflaged poisoning attacks in FL by proving two key claims: (1) the attack remains undetected during the initial training phase due to the camouflage effect, and (2) once the camouflage is removed, the poisoned data leads to significant model performance degradation. This analysis demonstrates how these attacks conceal adversarial updates and trigger substantial misclassifications once the camouflage is unlearned.

### 4.4.1 Claim 1: Undetectability during Training (Gradient Matching)

To demonstrate why camouflaged poisoning attacks remain undetectable during training, we explain how adversarial updates blend with normal updates using the technique of gradient matching, thereby minimizing detectability.

In FL, the global model parameters are updated based on the aggregation of local updates from participating clients. If some clients introduce poisoned data, they aim to make these updates appear indistinguishable from the updates of normal clients.

Aggregation of Updates: Let $\theta_t$ be the global model parameters at time $t$, and let $\Delta\theta_i$ represent the updates from normal clients. For an adversarial client, we denote the poisoned update as $\Delta\theta_{\text{att}}$. The global model's parameters are updated by aggregating these updates:

$$\theta_{t+1} = \theta_t + \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_i + \frac{1}{M}\sum_{j=1}^{M}\Delta\theta_{\text{att}}, \tag{4.3}$$

where $N$ is the number of normal clients and $M$ is the number of adversarial clients.

Creating Undetectable Poisoned Updates via Gradient Matching: To ensure that $\Delta\theta_{\text{att}}$ remains undetected, the adversarial updates are designed to mimic the average behavior of normal updates. This is achieved by aligning the gradients of poisoned data with the gradients of clean data. Specifically, the adversary aims to minimize the difference

between the gradients of poisoned data $(x_j + \Delta, y_j)$ and the gradients of normal data $(x_i, y_i)$. Mathematically, they seek to solve:

$$\min_{\Delta} \left\| \nabla_\theta \ell(f(x_j + \Delta, \theta), y_j) - \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \ell(f(x_i, \theta), y_i) \right\|_2^2, \tag{4.4}$$

where $\Delta$ is a small perturbation added to the adversarial data to align its gradients with those of the normal data.

Bounding Gradient Difference: By carefully optimizing $\Delta$, the difference between the gradients of poisoned and clean data can be minimized to a small value, denoted by $\epsilon$. This small $\epsilon$ indicates that the gradient differences are minimal, which makes it difficult for the aggregation mechanism to distinguish between normal and adversarial updates. Essentially, the poisoned updates ‚Äúblend in‚Äù with the normal updates. Impact on Global Model Parameters: Since $\Delta\theta_{\text{att}}$ is crafted to closely match the normal updates, its contribution to the global model's parameters is similar to that of a normal update. Thus, the model parameters after aggregation (with adversarial updates included) remain statistically similar to the parameters that would result from only normal updates. This can be represented as:

$$\|\theta_{t+1}^{\text{att}} - \theta_{t+1}^{\text{normal}}\|_2 \approx 0, \tag{4.5}$$

where $\theta_{t+1}^{\text{att}}$ is the global model with adversarial updates, and $\theta_{t+1}^{\text{normal}}$ is the model with only normal updates.

Minimal Effect on Model Performance: The small difference in parameters results in minimal impact on the model's overall performance. This ensures that during the training phase, the model behaves as expected and does not show any signs of degradation or anomalous behavior, keeping the attack hidden.

### 4.4.2 Claim 2: Model Degradation after Unlearning

In this part, we demonstrate that once the camouflage data is removed through the unlearning process, the hidden effects of poisoned data become apparent, leading to a measurable degradation in the model's performance.

Camouflaged poisoning attacks rely on the presence of both poisoned data and carefully crafted camouflage data during training. The purpose of the camouflage data is to mask the influence of poisoned data, maintaining the model's performance and preventing detection during initial training. However, once an unlearning request removes the

camouflage data, the poisoned data's adversarial effects are fully exposed, causing a shift in model behavior.

Effect of Removing Camouflage Data: Let $S_{\text{po}}$ represent the set of poisoned data, and let $S_{\text{ca}}$ represent the camouflage data, which together create the camouflaged attack set. During initial training, both $S_{\text{po}}$ and $S_{\text{ca}}$ are included in the training process, where $S_{\text{ca}}$ helps to balance the gradients of $S_{\text{po}}$, effectively hiding the attack.

Unlearning of Camouflage Data: An unlearning request removes the camouflage set $S_{\text{ca}}$ from the model, leaving only the poisoned set $S_{\text{po}}$ in the training data. This removal breaks the balance between poisoned and camouflage data, allowing the gradients of $S_{\text{po}}$ to have an unmasked influence on the model's parameters.

Model Parameter Shift: After the removal of $S_{\text{ca}}$, the global model's parameters are updated without the neutralizing effect of the camouflage data. Let $\theta_{t+1}^{\text{unlearned}}$ denote the model parameters after the unlearning process. The shift in parameters can be expressed as:

$$\theta_{t+1}^{\text{unlearned}} = \theta_{t+1} - \Delta\theta_{\text{ca}}, \tag{4.6}$$

where $\Delta\theta_{\text{ca}}$ represents the influence of the camouflage data that has been removed. This shift allows the poisoned gradients to dominate, which alters the model's decision boundaries.

Performance Degradation (Impact on Accuracy): With the parameters shifted by the influence of $S_{\text{po}}$ alone, the model's accuracy on clean validation data decreases, leading to an increase in misclassifications. The performance degradation can be represented by a change $\Delta P$ in accuracy:

$$\Delta P = |P(\theta_{t+1}^{\text{unlearned}}) - P(\theta_{t+1})|, \tag{4.7}$$

where $P(\theta_{t+1})$ represents the model's performance before unlearning and $P(\theta_{t+1}^{\text{unlearned}})$ is the performance after unlearning. Due to the dominance of poisoned gradients, $\Delta P$ becomes significant, resulting in degraded model accuracy.

## 4.5 Evaluation

The efficacy of our attack methods was assessed across various datasets, examining the influence of different parameters and both vanilla and robust FL system. Additionally, we conducted comparative analyses with established attacks to benchmark the performance of our strategies. We have applied CIFAR-10 and MNIST datasets in the experiment.

And Machine Learning Classifiers ResNet-18, VGG11 and ResNet-34 as the local model in our experiment. For FL setting, we use FedAvg as the aggregation algorithm and the number of clients from 0 to 200, a batch size of 32 is used for local training, the initial learning rate is set to 0.01 and decayed over time.

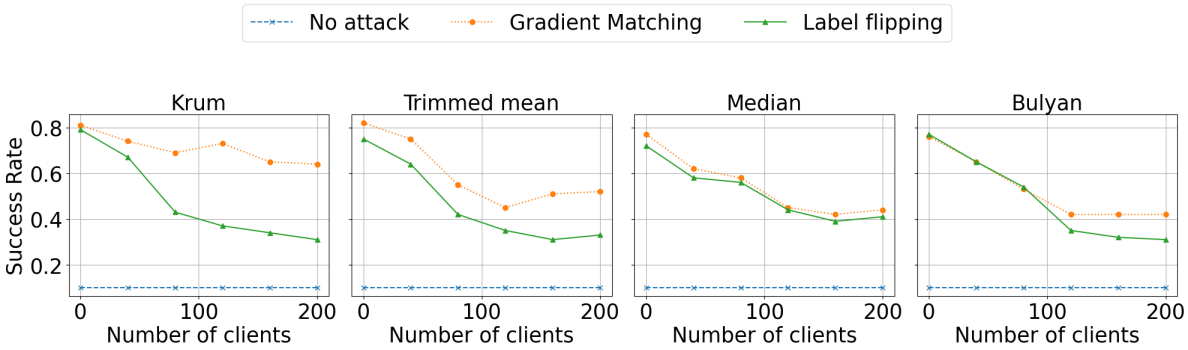


Figure 4.3: Success Rate vs. Number of Clients on MNIST

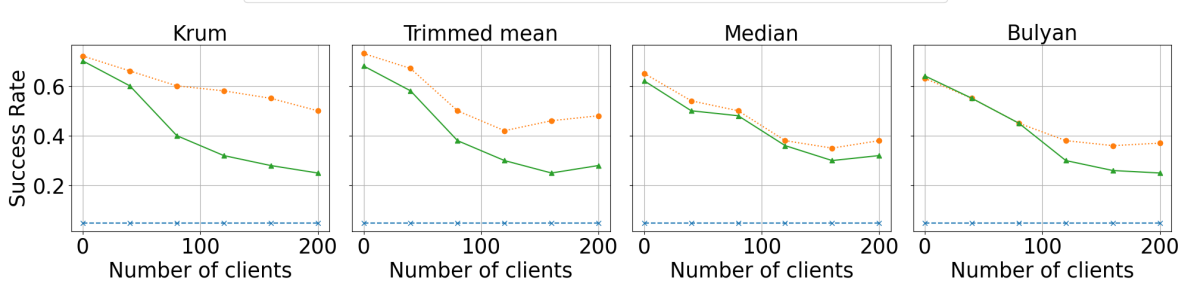


Figure 4.4: Success Rate vs. Number of Clients on CIFAR-10

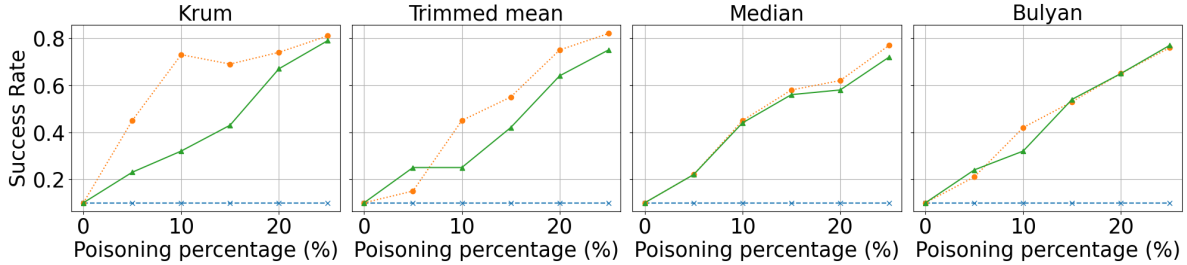


Figure 4.5: Attack Success Rates with Different Defenses on MNIST

*Baseline methods:*

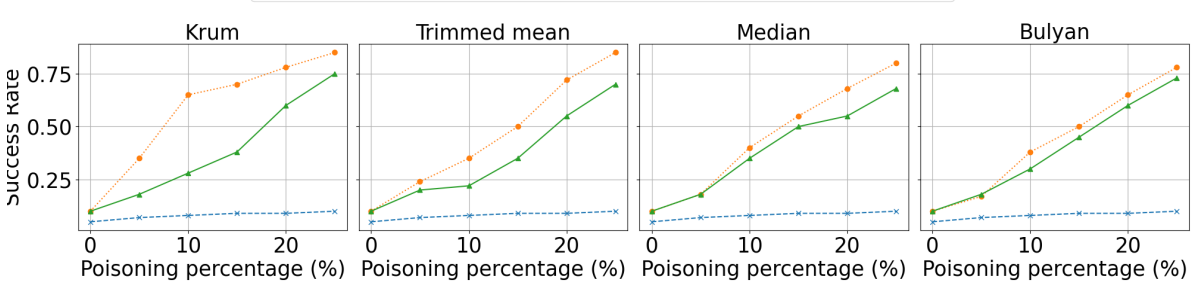


Figure 4.6: Attack Success Rates with Different Defenses on CIFAR-10

Table 4.1: Summary of Threat Models and Validation Accuracy

| **Threat model** | **Attack success** | | **Validation Accuracy (%)** | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | **bp** | **bc** | **Poisoning** | **Camouflaging** | **Clean** | **Poisoned** | **Camouflaged** |
| 16 | 1% | 1% | 85% | 82% | 82.13 | 81.88 (± 0.21) | 82.22 (± 0.25) |
| 8 | 1% | 1% | 80% | 77% | 82.13 | 82.12 (± 0.25) | 82.31 (± 0.19) |
| 16 | 2% | 1% | 92% | 35% | 82.13 | 82.15 (± 0.28) | 82.24 (± 0.15) |
| 8 | 2% | 1% | 95% | 45% | 82.13 | 82.31 (± 0.35) | 82.35 (± 0.16) |

Table 4.2: Attack Success Rates on Different Datasets and Models

| Dataset | Model | Attack Success | |
|---|---|---|---|
| | | Poisoning | Camouflaging |
| CIFAR-10 | VGG-16 | 45% | 100% |
| MNIST | Resnet-18 | 40% | 70% |

Krum, Bulyan, Trimmed Mean, and Median are aggregation methods used in FL to mitigate the impact of outliers or adversarial data, especially against poisoning attacks.

**Krum:** Selects the most reliable update by computing the sum of distances to nearest neighbors and choosing the one with the smallest sum, effectively excluding adversarial updates [6].

**Bulyan:** Builds on Krum by selecting a subset of updates using Krum, then applies a robust aggregation rule, making it more resilient to adversarial attacks [37].

**Trimmed Mean:** Discards the highest and lowest updates and computes the mean of the remaining ones, removing extreme values caused by noise or adversaries [116].

**Median:** Takes the median of all updates, providing robustness against outliers, as the median is less affected by extreme values compared to the mean [116].

*Metrics:*

**Success rate:** Measures the percentage of targeted adversarial samples that are misclassified as intended by the attacker, indicating the effectiveness of the attack.

Table 4.3: Attack Success and Validation Accuracy

| Attack type | Attack success | | Validation Accuracy | | |
|---|---|---|---|---|---|
| | Poisoning | Camouflaging | Clean | Poisoned | Camouflaged |
| Label Flipping | 70% | 71.5% | 81.5 | 81.4 | 81.5 |
| Gradient Matching | 95% | 75% | 81.3 | 81.5 | 81.6 |

**Validation accuracy:** Represents the accuracy of the model on a separate validation dataset.

### 4.5.1 Hyperparameter Definitions

This section discusses the roles of critical hyperparameters in machine learning models vulnerable to adversarial attacks and data poisoning.

$\epsilon$ (Epsilon): $\epsilon$ is a critical hyperparameter in machine learning models that are susceptible to adversarial attacks or data poisoning. In many contexts, $\epsilon$ denotes the maximum allowable perturbation that can be added to the input data to create adversarial examples or poison instances. This perturbation is designed to be small enough that it doesn't make the input obviously incorrect to human observers or simple validation checks, but sufficient to mislead the model into making errors. The roles of $\epsilon$ include:

Role in Poisoning:$\epsilon$ controls the intensity and stealthiness of the modifications applied to the data points intended as poison. The smaller the $\epsilon$, the less detectable the poison, but potentially less effective it is at manipulating the model's behavior.

In Camouflage: For camouflage data points, $\epsilon$ similarly governs how these points are perturbed to mask the effects of the poison data, ensuring that they do not alter the model's behavior until the camouflage is removed.

$\boldsymbol{b_p}$ **(Camouflage-to-Clean Data Ratio)**: $b_c$ describes the ratio of camouflage data points to clean data points within the training dataset. It is a measure of how much of the dataset is made up of data intended to obscure the effects of the poison data:

Purpose: The primary purpose of $b_c$ is to ensure that the camouflage data is sufficient to prevent the detection of the poison data during initial model training and validation phases.

Optimization: Determining the optimal $b_c$ is crucial. Too high a value might lead to unnecessary complexity or potential detection, while too low might not effectively hide the poison.

$\boldsymbol{b_p}$**(Poison-to-Clean Data Ratio)**: $b_p$ quantifies the proportion of poison data points

relative to the clean data points in the dataset. This ratio is pivotal in determining the effectiveness of the poisoning:

Effectiveness: $b_p$ needs to be high enough to ensure that the poison has a tangible effect on the model once the camouflage is unlearned, but not so high as to cause noticeable performance degradation or to trigger alerts during the initial training.

Balance: Like $b_c$, balancing $b_p$ is crucial. An optimal $b_p$ should be strategically set to avoid early detection but ensure substantial impact when activated.

Balancing $\epsilon$, $b_c$, and $b_p$ is essential for crafting effective and stealthy attacks in FL or other distributed learning environments. These hyperparameters must be tuned based on the specific characteristics of the model and the dataset to optimize the attack's stealth and impact.

Table 4.4: Comparison of Different Defense Mechanisms

| Defense Mechanism | Attack Success Rate | Clean Accuracy (%) | Poisoned Accuracy (%) |
|---|---|---|---|
| None | 85% | 82.13 | 81.88 |
| Krum | 60% | 81.95 | 81.50 |
| Bulyan | 45% | 82.00 | 81.70 |
| Trimmed Mean | 50% | 81.98 | 81.60 |
| Median | 55% | 81.90 | 81.55 |
| Differential Privacy | 30% | 81.80 | 81.30 |

Table 4.5: Computation Time and Resource Consumption

| Process | Time (s) | Resource Usage (CPU/GPU) |
|---|---|---|
| Training without Attack | 500 | 20 CPU, 1 GPU |
| Training with Poisoning | 700 | 20 CPU, 1 GPU |
| Training with Camouflaging | 800 | 20 CPU, 1 GPU |
| Applying Defense (Krum) | 850 | 20 CPU, 1 GPU |
| Applying Defense (Bulyan) | 870 | 20 CPU, 1 GPU |
| Applying Defense (Differential Privacy) | 900 | 20 CPU, 1 GPU |

### 4.5.2 Results

The effectiveness of the camouflaged poisoning strategy employed on the CIFAR-10 dataset and MNIST is detailed through a series of experiments. The experimental setup was designed to evaluate the impact of specific adversarial parameters on various neural network architectures.

Table 4.3 Gradient Matching emerges as a significantly more potent attack method compared to Label Flipping, achieving a 95% success rate in poisoning the model. Both attack methods are also effective in camouflaging their presence, with Gradient Matching having a 75% success rate. Despite their effectiveness, the validation accuracy of the model remains largely unaffected across clean, poisoned, and camouflaged conditions, which underscores the subtlety of these attacks. This indicates that even when the attacks succeed, they do so without significantly degrading the model's overall performance, making them harder to detect.

Table 4.4 shows when it comes to defending against these attacks, Differential Privacy stands out as the most robust mechanism, reducing the attack success rate to 30%. Bulyan also offers strong protection, with a 45% success rate for attacks. Both methods maintain relatively high model accuracy, though there is a slight trade-off in terms of performance when compared to a model with no defense. Krum, Trimmed Mean, and Median provide moderate levels of protection, with success rates ranging from 50% to 60%, but they do not match the effectiveness of Differential Privacy or Bulyan. This indicates that while these methods are helpful, they may not be sufficient against more sophisticated attacks like Gradient Matching.

Table 4.5 show that implementing defenses increases both computation time and resource consumption. Differential Privacy, while highly effective, is the most computationally expensive defense, requiring 900 seconds of processing time. Bulyan and Krum also incur additional computational costs, reflecting the trade-off between enhanced security and resource efficiency. Even the attack strategies themselves (poisoning and camouflaging) add to the computational burden, suggesting that both attackers and defenders in a FL environment must account for these increased costs.

Figure 4.7 shows the Success rates of the camouflaged poisoning attack across different neural network architectures with threat model parameters $\epsilon = 16$, $b_p = 0.6\%$, and $b_c = 0.6\%$.

Since there are no existing studies on unlearning-based poisoning attack in FL, we take the some of the defense method which are used to defense against normal poisoning attack in Byzantine-Robust FL. We evaluate the attacks under the four types of Byzantine-robust aggregation rules:

In Byzantine-robust FL, the primary challenge is to ensure the integrity of the aggregated global model, despite potential adversarial attacks or failures from participant nodes. This section details several aggregation rules designed to mitigate such risks through robust statistical methods.

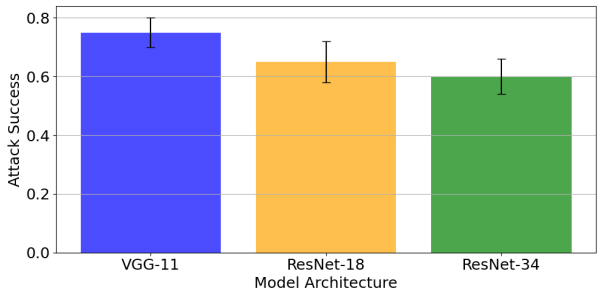


Figure 4.7: Success Rates of Camouflaged Poisoning Attack.

The effectiveness of FL relies significantly on the aggregation method used to compile local updates into a global model. While simple averaging is common, it is vulnerable to attack. As such, more resilient methods have been developed:

**Krum [6]**: Krum is designed to select the most reliable update among a set of candidates from different workers. The selection criterion is based on the squared Euclidean distance. Mathematically, for a set of vectors $\{v_1, v_2, \ldots, v_n\}$, Krum selects the vector $v_i$ that minimizes the sum of distances:

$$\text{Krum}(v_i) = \arg\min_{v_i} \sum_{v_j \in \text{nearest } (n-f-2) \text{ of } v_i} \|v_i - v_j\|^2, \tag{4.8}$$

where $f$ is the number of Byzantine workers, and $n$ is the total number of workers.

**Bulyan [37]**: Bulyan extends Krum by applying it iteratively to exclude outliers progressively and then taking a trimmed mean of the selected models. The Bulyan operator acts on the subset chosen by Krum:

$$\text{Bulyan}(\mathscr{S}) = \text{TrimmedMean}(\text{Krum}(\mathscr{S}, k), l), \tag{4.9}$$

where $k$ is the number of iterations and $l$ is the trim parameter.

**Trimmed Mean [116]**: This method involves calculating the mean of the parameters after excluding the highest and lowest $\beta\%$ of the values to reduce the impact of outliers:

$$\text{Trimmed Mean}(x_1, \ldots, x_n) = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_{(i)}, \tag{4.10}$$

where $x_{(i)}$ are the order statistics, and $k = \lfloor n\beta/100 \rfloor$.

**Median [116]**: Median aggregation simply computes the median of all submitted updates for each parameter, providing robustness against extreme values:

$$\text{Median}(x_1, \ldots, x_n) = \text{median}(x_1, \ldots, x_n). \tag{4.11}$$

These robust aggregation methods are critical for maintaining the reliability of the learning process, especially in environments where the presence of malicious actors is assumed.

Figure 4.3 shows the changes in Success Rate with Increasing Number of Clients in FL on MNIST Dataset. Under the Krum aggregation method, the success rate of attacks significantly decreases as the number of clients increases from 0 to 200. Initially, Gradient Matching is more effective, but its success rate gradually declines with more clients. The success rate of the Label Flipping attack is consistently low and decreases further as the number of clients increases. With the Trimmed Mean aggregation method, the success rate of attacks also decreases as the number of clients increases, though the decline is more gradual compared to Krum. Gradient Matching remains more effective than Label Flipping, but its success rate also diminishes over time. For the Median aggregation method, the decline in attack success rate is similar to that of Trimmed Mean, but with slightly lower overall success rates. The impact of Gradient Matching decreases over time, and Label Flipping remains at a consistently low success rate. Bulyan shows the strongest defensive capability, maintaining the lowest success rates for attacks across all client numbers. Both Gradient Matching and Label Flipping see significant reductions in success rates, with Bulyan providing the highest resistance to both attacks.

Figure 4.4 shows the changes in Success Rate with Increasing Number of Clients in FL on CIFAR-10 Dataset. Similar to the MNIST dataset, the success rate of attacks under Krum decreases significantly as the number of clients increases. Gradient Matching starts with a higher success rate but drops sharply with more clients. Label Flipping remains less effective with consistently low success rates. The success rate of attacks under Trimmed Mean decreases with more clients, with Gradient Matching being more effective than Label Flipping, though both diminish as client numbers grow. Under the Median aggregation method, the pattern is similar to the MNIST results, with a decreasing success rate for attacks as client numbers increase. Gradient Matching remains more potent, but its effectiveness decreases over time. Bulyan again demonstrates the strongest defense, showing the lowest attack success rates as the number of clients increases, especially against both Gradient Matching and Label Flipping attacks.

Figures 4.5 and Figure 4.6 show the attack success rates for different defenses versus our method in MNIST and CIFAR-10. As the poisoning percentage rises from 0% to 20%, the success rate of attacks increases significantly. Gradient matching is particularly effective, with its success rate rising sharply as the percentage of poisoning increases.

Label Flipping also shows an increase, though to a lesser extent. The Trimmed Mean method shows a similar trend, with increasing success rates for both Gradient Matching and Label Flipping attacks as the poisoning percentage rises. Gradient Matching remains the more effective of the two. The Median aggregation method also exhibits an increase in attack success rates with higher poisoning percentages. Gradient Matching is more successful, but Label Flipping also poses a considerable threat as the percentage increases. Bulyan remains the most effective defense, showing the smallest increase in attack success rates as the percentage of poisoning increases. Even at higher poisoning percentages, Bulyan's defense is more robust compared to other methods.

The figures illustrate how different aggregation methods in FL respond to Gradient Matching and Label Flipping attacks under varying conditions. Figure 4.3 and Figure 4.4 demonstrate that as the number of clients increases, the success rates of attacks decrease across all methods, with Bulyan showing the greatest resistance. Figure 4.5 highlights the challenge posed by increasing poisoning percentages, where the success rates of attacks rise, especially for Gradient Matching.

From an attack method perspective, Gradient Matching poses a greater threat to FL models, especially when fewer clients are involved or when a higher percentage of data is poisoned. It is more sophisticated and adaptable, making it a challenging attack to defend against. In contrast, Label Flipping is less effective and poses a smaller threat, though it can still cause significant harm if not properly mitigated, particularly in environments with fewer clients or higher poisoning percentages.

## 4.6 Related Work

Research on attacks in FL has primarily focused on poisoning and adversarial attacks that manipulate model updates, while robust FL methods, such as Byzantine-tolerant algorithms, have been developed to defend against these threats by ensuring the security and integrity of the global model in adversarial environments.

### 4.6.1 Federated Unlearning

Federated unlearning is an emerging subfield within FL, focused on meeting the growing demand for privacy-preserving techniques that enable the removal of specific data from machine learning models without compromising the global model's integrity or performance. The key challenge of federated unlearning stems from its decentralized

nature. Unlike traditional machine unlearning, which involves retraining a model after excluding specific data, federated unlearning requires the model to forget data without access to the full dataset and must operate efficiently across distributed nodes that contribute independently to model training with varied data subsets. The goal is to ensure the global model behaves as if the unlearned data were never part of its training, all while avoiding the computational cost of full retraining [54, 111].

Various strategies for federated unlearning have been proposed, such as selective retraining, where only the affected parameters are updated [20, 101], and gradient reversal techniques that neutralize the influence of the data to be forgotten [105]. These methods aim to strike a balance between computational efficiency and the completeness of the unlearning process, ensuring that the model forgets specific data while minimizing computational overhead.

However, federated unlearning also introduces new security challenges. Malicious clients can exploit the unlearning process through attacks such as camouflaged data poisoning, where adversaries introduce specially crafted data points during training and later request their removal. These attacks are designed to degrade the global model's performance after the unlearning process, exposing a critical vulnerability in current FL systems.

In conclusion, federated unlearning is a vital advancement for addressing privacy and compliance concerns in distributed machine learning. However, it brings complex challenges that require careful attention. The need to efficiently remove data from models while protecting against adversarial attacks highlights the delicate balance necessary in implementing federated unlearning. As research progresses, the development of more advanced and secure unlearning techniques will be crucial for ensuring both privacy and robustness within FL frameworks in a data-driven world.

### 4.6.2 Robust FL

Robust FL addresses the challenges of building reliable and secure models in distributed environments, especially in the presence of adversarial threats [29]. Its primary goal is to ensure the efficiency, privacy, and resilience of the FL process, even when faced with corrupted or malicious data. Since FL involves data from diverse sources, robust mechanisms are required to detect and mitigate the impact of poisoned data, ensuring that the global model is trained on reliable inputs [24].

To counter adversarial updates, robust FL uses advanced aggregation techniques and anomaly detection to reduce the influence of malicious contributions [130]. These

strategies help maintain the model's integrity and accuracy, even with non-IID data across different clients [133]. Privacy-preserving methods, such as differential privacy, are also integrated to protect individual updates from inference attacks, ensuring security even in adversarial settings [118].

Moreover, robust FL includes resilience against Byzantine faults, where some nodes behave maliciously or arbitrarily, ensuring the system continues to function properly [90]. This paper leverages robust FL as a defense mechanism to evaluate the effectiveness of our proposed attack, contributing to the ongoing efforts to secure FL systems.

### 4.6.3 Comparison with Exiting Approaches

The growing interest of research on adversarial exploration and secure in unlearning in both centralized and distributed machine learning system highlights the challenges and opportunities. Some researchers focus on Machine-Learning-as-a-Service (MLaaS), emphasizing unlearning-activated backdoor attacks (UBA-Inf) in centralized systems [49]. In this method, adversaries embed dormant backdoor into models during training, which are activated through unlearning requests. They focuses on ensuring that backdoor remain stealthy and persistent, bypassing standard defenses such as anomaly detection. In contrast, our work diverges significantly from UBA-Inf by focusing on FL environments, where data and model updates are distributed across clients. Unlike UBA-Inf, which exploits unlearning to activate backdoors in centralized systems, our study highlights the risks posed by camouflaged poisoning attacks and proposes tailored mitigation strategies for FL settings. While UBA-Inf provides valuable insights into the stealthy exploitation of unlearning in MLaaS systems, our work uniquely contributes to the understanding and mitigation of federated unlearning vulnerabilities in decentralized learning frameworks. By addressing the interplay between poisoning attacks and unlearning, our study offers a foundation for improving the robustness and security of FL systems under adversarial conditions.

### 4.6.4 Potential Countermeasures

Although the analysis in this chapter focuses on exposing the vulnerability created by camouflaged updates, several defenses could mitigate this behaviour. One possible direction is the use of multi-stage aggregation rules that combine robust statistics with temporal consistency checks. These methods examine how a client‚Äôs update evolves across rounds rather than relying on a single snapshot. A malicious client that hides

its behaviour behind benign patterns would therefore have a harder time maintaining consistency.

Another direction is to incorporate contribution auditing. By estimating per-client influence, the server can identify updates whose long-term effect deviates from their apparent short-term behaviour. Influence-based auditing does not need to reconstruct the full gradient and can detect situations where an update appears benign but gradually alters the decision boundary.

It is also possible to introduce layered anomaly detection. Instead of examining only the aggregated update, the server may analyse intermediate features, loss profiles or local prediction behaviour. These additional signals may reveal subtle patterns introduced by camouflaged poisoning.

Finally, incorporating unlearning-aware training pipelines can reduce the risk that a benign masking component becomes essential for model stability. By ensuring that the removal of any single client does not disproportionately affect the model, the system becomes less vulnerable to attacks that rely on post-unlearning amplification.

These ideas do not eliminate the challenge but provide starting points for reducing the impact of the attack demonstrated in this chapter.

### 4.6.5 Novelty and Positioning Within Existing Poisoning Threats

Existing poisoning research in federated learning largely focuses on attacks that introduce harmful gradients directly into the aggregation process. Examples include model replacement, directional attacks, and trigger-based backdoors, all of which attempt to inject adversarial behaviour at training time. These attacks usually rely on large gradient deviations or detectable shifts in the update distribution, and the defense literature assumes that malicious behaviour must manifest during the training stage.

The threat examined in this chapter differs in a fundamental way. Rather than introducing an overtly harmful gradient, the adversary constructs a mixed update that combines a benign-like component with a small adversarial signal. During normal training the masking component dominates and the aggregated update appears statistically similar to those of honest clients. As a result the attacker can avoid detection by both robust aggregation and anomaly-based defenses.

The novelty lies in the delayed emergence of the malicious effect. Once unlearning removes the benign masking component, the previously hidden adversarial part becomes proportionally amplified. This creates a form of post-unlearning activation in which harmful behaviour appears only after a legitimate removal request is processed. To

the best of our knowledge, existing poisoning attacks do not rely on the interaction between training-time masking and unlearning-time amplification, nor do they exploit the stability assumptions that typical defenses implicitly depend on.

This shift in perspective highlights that unlearning is not only a privacy mechanism but also a new attack surface. Camouflaged poisoning demonstrates that an adversary can shape how its contribution behaves under future model modifications, revealing a class of threats that is absent from traditional poisoning scenarios.

## 4.7 Conclusion

In summary, we explore the significant implications of federated unlearning. Our investigation highlights critical security vulnerabilities inherent to the federated unlearning process, particularly through the lens of deceptive data perturbation attacks. We have demonstrated that while federated unlearning aims to address privacy and compliance effectively, it inadvertently vulnerable to novel attacks. These attacks, executed by injecting and later requesting the removal of strategically designed data points by adversarial clients, capitalize on the unlearning mechanism to manipulate the model's predictions significantly. Our findings suggest that there is an urgent need to develop more sophisticated security measures to protect against such attacks. Future research should therefore focus on advancing detection techniques and designing resilient FL architectures that can not only respond to changes in data privacy requirements but also defend robustly against these emerging security threats. This will ensure that FL can continue to be a viable and secure option for handling sensitive data across distributed networks.

CHAPTER 5

# FEDERATED UNLEARNING WITH REINFORCEMENT LEARNING: ADAPTIVE PRIVACY PRESERVATION FOR CLIENTS

This chapter introduces a reinforcement-learning-based unlearning mechanism that adapts to client contribution, privacy sensitivity, and efficiency. It describes the DQN-based framework, analyses its operation, and validates performance against poisoning and other adversarial behaviours.

## 5.1 Introduction

Although the current research has proposed many methods to solve the problems in FL, there are still many challenges left. The accuracy degradation is a important problem when some specific clients' contributions are removed, particularly when informative data is eliminated, thereby reducing the model's generalization capability[85]. In addition, there are still difficulties in most FL frameworks, especially in the unlearning process, which can be exploited by adversarial clients to perform targeted attacks, such as poisoning attacks that manipulate the data removal process to degrade model performance[56]. Although federated unlearning improves user privacy, excessive or frequent unlearning requests can compromise overall model stability, requiring a balance between privacy guarantees and learning efficacy[46]. As a result, the privacy trade-offs also play an important role in the urgently needed to be solved troubles.

In summary, we propose an effective federated unlearning approach that improves not only the training model's performance, but also its robustness against poisoning. In order to reduce degradation in accuracy due to purging of client contributions, we propose a reinforcement learning-based method to dynamically select unlearning actions, such that the generalizability of the model is maintained. Depending on the importance of a client's data to the global model, we train deep Q-Networks (DQNs) to decide whether the optimal action is partial unlearning, full unlearning, or no unlearning. To alleviate possible threats in security, while unlearning we incorporate adaptive defenses to identify and offset possible poisoning in the unlearning process. Our system dynamically adjusts the unlearning strategy to prevent malicious clients from hijacking the process to degrade the model performance by continuously learning adversary behaviors. To enable trade-offs between privacy guarantees and model stability, we design a smart trade-off knob that dynamically controls the frequency of unlearning to optimize privacy cost vs. computation overhead. With the dynamic unlearning rate of actions, our method can balance the model accuracy, efficient learning and privacy protection.

We conduct thorough experiments on some datasets to verify that the proposed approach is more effective in enhancing the model performance and defending poisoning attacks. Our findings show that embedding reinforcement learning for federated unlearning results in gains on model accuracy and robustness against adversarial manipulation in the scenario of decentralized learning. In summary, we make the following contributions:

- Contribution 1.We present a defense mechanism against poisoning attacks by integrating unlearning techniques that mitigate the impact of malicious data on the global model.

- Contribution 2.We propose a reinforcement learning-based method for federated unlearning by dynamically selecting unlearning actions of clients.

- Contribution 3.We validate the effectiveness of the proposed approach through experiments on multiple datasets, demonstrating its potential to improve FL performance and improve robustness.

## 5.2 Background

### 5.2.1 Poisoning Attack in FL

FL is vulnerable to poisoning attacks, where malicious clients manipulate data or model updates to degrade performance [95]. Data poisoning introduces adversarial samples or mislabeled data, affecting aggregated updates:

$$\min_{w} f(w) = \sum_{k=1}^{m} p_k F_k(w|D_k) \tag{5.1}$$

where for adversarial clients. A more advanced model poisoning attack modifies local updates to steer global parameters:

$$w_{t+1} = w_t + \frac{1}{m} \sum_{i=1}^{m} \Delta w_i \tag{5.2}$$

A critical example is backdoor attacks, where an embedded trigger activates malicious behavior:

$$f(x|w) = \begin{cases} \text{normal outcome}, & x \neq t \\ \text{malicious outcome}, & x = t \end{cases} \tag{5.3}$$

These attacks persist after training, making detection and mitigation essential. [102].

### 5.2.2 Problem Statement

Unlike that in traditional machine unlearning which focuses on effectively filtering the influence of certain client data out of a collaboratively trained model while preserving model performance and complying with privacy laws, federated unlearning brings in an extra challenge due to the decentralized learning in FL: client data are local and not directly available at the central server.

The basic goal is to update the global model so that its behavior corresponds to that of a not yet seen by the model data, based on a global model previously trained on such different models on an old data, while not retraining the model at all. Formally, given a FL system with $N$ clients, each contributing local updates $\Delta w_i$ to the global model, federated unlearning seeks an approximation $\tilde{w}$ satisfying:

$$\tilde{w} \approx w^{(-i)} \tag{5.4}$$

where $w^{(-i)}$ represents the global model obtained if client $i$'s data had never been included in the training process. The objective is to achieve this approximation efficiently while minimizing computational overhead and ensuring model stability.

- Unlearning Effectiveness: The level to which forgotten data influence is completely removed from the model.
- Computational Efficiency: We should be able to avoid relearning the model from scratch, which is infeasible in large-scale federated scenarios.
- Model Performance: Mitigating accuracy decay due to unlearning while preserving generalization.
- Security and Robustness: Stopping adversarial manipulation, by which malicious client does not make the unlearning process of the model to run smoothly for degrading its accuracy.

Given these constraints, federated unlearning requires adaptive strategies. As a result, our method brings reinforcement learning-based improvement, to dynamically determine unlearning actions based on client contributions, privacy risks, and computational costs.

## 5.3 Proposed Method

In this section, we present the proposed method for FL improvement by reinforcement learning (RL) with Deep Q-Networks (DQN) and the incorporation of unlearning strategies to enhance model quality in the presence of poison attackers. Our mechanism can choose to take either no unlearning action, partial unlearning, or complete unlearning on the fly for different clients, based primarily on key statistics including contribution, privacy cost and computational cost. Furthermore, we introduce the methods used to defend the federated model against adversarial poisoning during the unlearning procedure.

Fig. 5.1 represents the workflow of reinforcement learning-based federated unlearning, divided into three main stages:

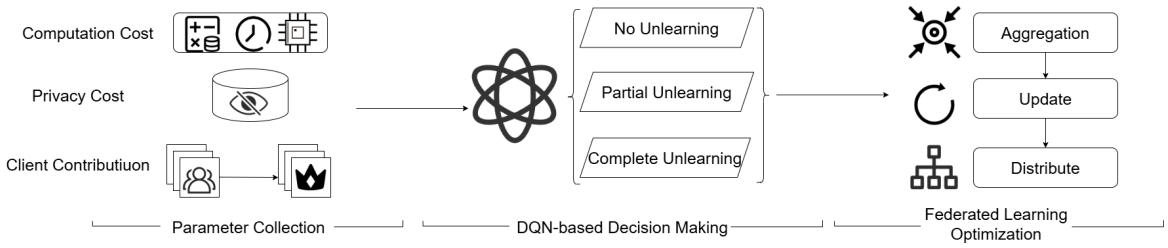


Figure 5.1: Overview of Proposed Method

- Parameters Collection : The client contribution, privacy cost and computational cost are collected by the system to determine unlearning decisions.

- DQN Guidance Policy: A Deep Q-Network (DQN) decides for each client, which unlearning action (No Unlearning, Partial Unlearning, or Complete Unlearning) should be performed to achieve an accuracy-privacy-efficiency trade-off.

  FL Optimization: The updated model is subsequently optimized by aggregation, global model updates, and deployment to clients, to facilitate both on-going learning and unlearning adjustments.

### 5.3.1 Parameters Collection

In FL, the decision of whether and how to unlearn a client's data must be made carefully to balance model performance, privacy protection, and computational efficiency. To guide this decision-making process, we define three key metrics that capture the contribution of each client to the global model, the privacy cost associated with their data, and the computational overhead involved in unlearning actions. These metrics are crucial for ensuring that unlearning decisions are both effective and efficient, especially in dynamic FL environments. This three metrics are the mainly parameters we need to collect in phase one.

#### 5.3.1.1 Client Contribution

Client contribution measures the extent to which the local data of a client has contributed to the performance of the global model. It is an important criterion to evaluate the contribution of a client's data in the training. Large client contribution: This means if we remove the data of that client, it might have significant impact on the performance of global model while small contribution means client data does not effect the model.

We measure client contribution by analyzing the gradient update $\Delta W_i$ that the client $i$ sends to the server after local training. Specifically, the contribution is computed as the magnitude of the client's gradient relative to the total gradient from all clients. Formally, the contribution of client $i$ can be expressed as:

$$\text{Contribution}_i = \frac{\|\Delta W_i\|}{\sum_{j=1}^{N} \|\Delta W_j\|} \tag{5.5}$$

Where:

- $\Delta W_i$ is the gradient update from client $i$,
- $N$ is the total number of clients participating in the current FL round.

Client contribution measures the extent to which the local data of a client has contributed to the performance of the global model. It is an important criterion to evaluate the contribution of a client's data in the training. Large client contribution: This means if we remove the data of that client, it might have significant impact on the performance of global model while small contribution means client data does not effect the model.

#### 5.3.1.2 Privacy Cost

The privacy cost measures what damage it would do to a subject if their data (encoded or not) would be part of the global model. This quantity is important to determine if a client's data should be unlearned in order to comply with privacy laws or client wishes.

We present privacy cost as a composite function of:

- Data Sensitivity: How sensitive the data inherently is, possibly conditioned on the type of data (e.g., medical records, financial data) and the risk associated with the leakage o fthe data through the exposure of model updates.
- Unlearning Requests: If a user is explicitly submitting a forget-me-request, their privacy cost is increased.

The privacy cost for client $i$ can be represented as:

$$\text{Privacy Cost}_i = \alpha \cdot \text{Data Sensitivity}_i + \beta \cdot \mathbb{1}_{\{\text{unlearning request from client } i\}} \tag{5.6}$$

Where:

- $\alpha$ and $\beta$ are weighting factors that control the influence of data sensitivity and unlearning requests, respectively.
- $\mathbb{1}_{\{\text{unlearning request from client } i\}}$ is an indicator function that takes the value 1 if the client has requested unlearning and 0 otherwise.

High privacy cost suggests that the client's data is sensitive or they have requested its removal, making them a strong candidate for unlearning, either partially or completely. In scenarios where privacy regulations (such as GDPR) mandate data removal, this metric can help prioritize unlearning decisions based on regulatory requirements.

#### 5.3.1.3 Computational Cost

The computational cost metric is indicative of the resources needed to perform unlearning for a client and includes the cost of retraining, communication, and processing. Although unlearning can be from zero effort (that is no unlearning) to a lot of effort (totalization relearning), the computation cost is a crucial element in determining the best strategy for unlearning.

Computational cost is influenced by:

- Training Cost: The resources required to retrain the global model after unlearning the client's data. Complete unlearning generally incurs higher costs, as the influence of all the client's data must be removed.
- Communication Cost: The overhead associated with sending updated models or gradients during unlearning, especially in cases where multiple rounds of communication are necessary.
- Unlearning Cost: The specific cost of applying the chosen unlearning action (partial or complete).

The computational cost for unlearning client $i$ is given by:

$$\begin{aligned} \text{Computational Cost}_i = \gamma_1 \cdot \text{Training Cost}_i \\ + \gamma_2 \cdot \text{Communication Cost}_i \\ + \gamma_3 \cdot \text{Unlearning Cost}_i \end{aligned} \tag{5.7}$$

Where $\gamma_1$, $\gamma_2$, and $\gamma_3$ are weight factors that control the influence of each type of cost.

This metric ensures that unlearning decisions take into account not only the client's contribution and privacy needs but also the efficiency of the overall system in terms of resource usage.

### 5.3.2 DQN-based Decision Making

In classic FL, clients train a global model by training local models with their own local private data and transmit the updates to the central server. However, FL does not itself offer built-in support for the removal of data that a client might want to retract or ‚Äúunlearn‚Äù. For this purpose, we propose three unlearning techniques that can be performed on individual clients during FL:

In **No Unlearning**, the client's data remains fully intact in the global model, with no modifications made in response to an unlearning request. The client's contribution $\Delta W_i$ is preserved in its entirety, and the global model is updated as usual without removing any data. This approach ensures maximum model accuracy and incurs no additional computational cost, as no unlearning operations are performed. However, it provides no privacy protection for clients seeking to remove their data from the model.

In **Partial Unlearning**, a subset of the client's data $D_i'$ is removed from the global model, while the remaining data $D_i^{\text{kept}}$ continues to contribute to the model. This is achieved by reweighting the client's update $\Delta W_i$, allowing only a fraction $\alpha$ of the client's contribution to persist in the model, with the unlearned subset being discounted. This approach balances privacy protection and performance preservation, offering a middle ground between fully retaining and completely removing the client's data. While it reduces computational cost compared to complete unlearning, it still involves the complexity of determining which data subset should be removed.

In **Complete Unlearning**, all client-donated beliefs in $B_i$ are extracted, the same way that those beliefs (and all projected references on input features) are removed from the beliefs when training the model. The global model is updated by reverse the updates of the client data so to remove the client data from the model. This is the strongest protection of privacy, appropriate for compliance with very prohibitive privacy laws, or defense from adversarial clients. But full unlearning is computationally costly and can degrade the task performance, especially when the client's data contributed significantly to the accuracy of the model.

To improve the potential of the unlearning decisions for FL, we formulate the unlearning problem as a reinforcement learning (RL) problem and we propose a DQN based agent to adaptively select a proper unlearning action for each client. The reinforcement learning task consists of:

**State ($S$)**

The state captures the current status of each client in the FL process. Specifically, the state vector $S_i^t$ for client $i$ at round $t$ consists of three components:

- Contribution$_i^t$: The contribution of client $i$ to the global model's performance.
- Privacy Cost$_i^t$: The privacy cost associated with client $i$'s data.
- Computational Cost$_i^t$: The computational cost of applying an unlearning action to client $i$.

Thus, the state of client $i$ at time $t$ is represented as:

$$S_i^t = [\text{Contribution}_i^t, \text{Privacy Cost}_i^t, \text{Computational Cost}_i^t] \tag{5.8}$$

**Action ($A$)**

The action space consists of the possible unlearning actions that can be taken for each client. The action $A_i^t$ for client $i$ at round $t$ can be one of the following:

- $A_i^t = 0$ (No Unlearning): No changes are made to the client's contributions.
- $A_i^t = 1$ (Partial Unlearning): A portion of the client's data is selectively removed from the model.
- $A_i^t = 2$ (Complete Unlearning): The client's data is entirely removed from the global model.

**Reward ($R$)**

The reward function is designed to balance the trade-off between improving model performance and minimizing computational cost, while also protecting client privacy. The reward $R$ for taking action $A$ at time $t$ is defined as:

$$R_i^t = \Delta\text{Accuracy}_i^t - \lambda \cdot \text{Computational Cost}_i^t \tag{5.9}$$

Where:

- $\Delta\text{Accuracy}_i^t$ is the change in the global model's accuracy after applying the unlearning action.
- $\lambda$ is a weighting factor that controls the trade-off between model performance and computational overhead.
- Computational Cost$_i^t$ represents the resource cost associated with applying the unlearning action to client $i$.

**Policy**

The DQN agent's objective is to learn a policy $\pi$ that selects the optimal action $A_i^t$ for each client at each time step, maximizing the expected cumulative reward:

$$\pi(S_i^t) = \arg\max_A Q(S_i^t, A) \tag{5.10}$$

Where $Q(S_i^t, A)$ is the action-value function, which estimates the expected reward for taking action $A$ in state $S_i^t$.

### 5.3.3 FL Optimization

The algorithm 5.1 shows our method that integrates reinforcement learning into federated unlearning to dynamically determine optimal unlearning actions for each client. At the beginning of each communication round, the server distributes the global model to participating clients, who then train their local models. Each client evaluates its contribution, privacy cost, and computational overhead, forming a state representation that is fed into a Deep Q-Network (DQN). The DQN selects one of three actions-no unlearning, partial unlearning, or complete unlearning-based on the client's state. The selected action modifies the client's update before aggregation. The server then updates the global model by aggregating the received updates and computes rewards for the DQN based on the trade-offs between model performance, privacy, and computational efficiency.

**Algorithm 5.1** FL with DQN-Based Unlearning Decisions

**Input:** Global model $W$, number of communication rounds $T$, DQN parameters $\theta$, learning rate $\alpha$, step size $\beta$
**Output:** Updated global model $W$ after unlearning
Initialize global model $W$
**for** $t = 1$ to $T$ **do**
  Server sends global model $W^t$ to all clients
  **for** each client $i$ **do**
    Train local model $W_i^t$ using client data $D_i$
    Compute metrics: contribution$_i$, privacy cost$_i$, computational cost$_i$
    Define state $S_i = [\text{contribution}_i, \text{privacy cost}_i, \text{computational cost}_i]$
    DQN selects action $A_i$ based on state $S_i$
    **if** $A_i = 0$ **then**
      No unlearning is performed
      Set $\Delta W_i$ to $W_i^t - W^t$
    **else if** $A_i = 1$ **then**
      Perform partial unlearning
      Select subset $D_i'$ for partial unlearning
      Compute $\Delta W_i$ for partial unlearning
    **else if** $A_i = 2$ **then**
      Perform complete unlearning
      Set $\Delta W_i$ to $-(W_i^t - W^t)$
    **end if**
    Update global model: $W^t = W^t + \Delta W_i$
  **end for**
  Aggregate global model updates: $W^{t+1} = \text{Aggregate}(W^t)$
  **for** each client $i$ **do**
    Compute reward for client $i$ based on contribution$_i$ and computational cost$_i$
    Update DQN policy with reward$_i$
  **end for**
**end for**
**Return:** Final global model $W$

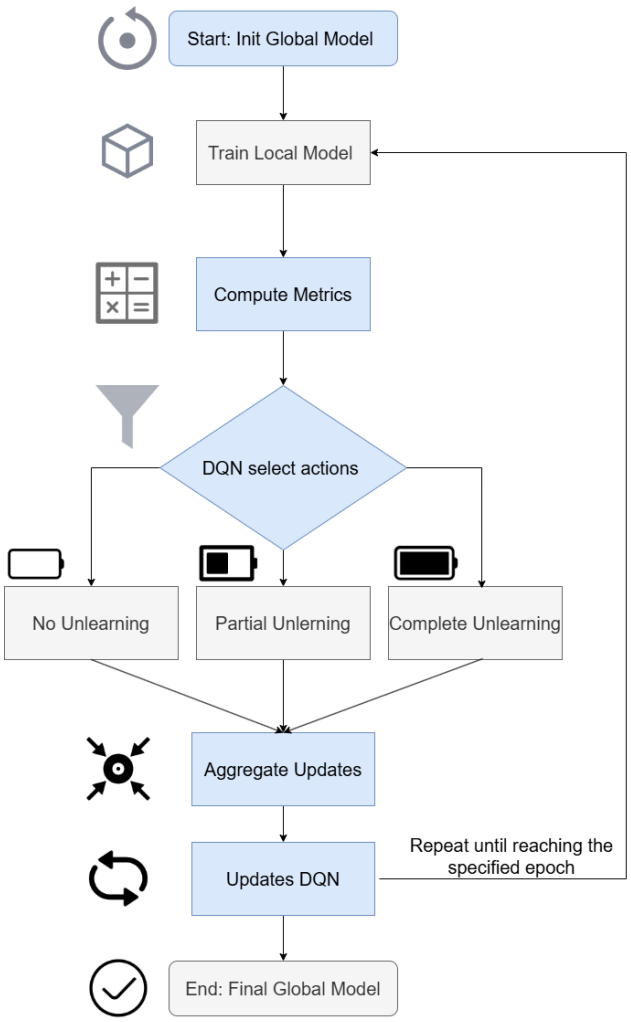


Figure 5.2: Workflow of Proposed Method

Fig. 5.2 represents the FL Process with DQN-Based unlearning Decisions.

- The process begins by initializing the global model $W$ ("Start: Init Global Model $W$").
- Each client trains a local model $W_i^t$ based on their data ("Train Local Model $W_i^t$").
- Clients then compute metrics like contribution, privacy cost, and computational cost ("Compute Metrics").
- The DQN agent selects an action based on these metrics ("DQN Selects Action"). The action could be one of three:
  - No Unlearning: The client's contribution is fully retained.

  - Partial Unlearning: A subset of the client's data is unlearned.
  - Complete Unlearning: The client's entire contribution is removed.

- After the unlearning action is applied, the system aggregates updates from the clients ("Aggregate Updates").
- The DQN agent's reward is computed based on the outcome, and the DQN policy is updated to improve future decisions.
- The process ends with the final global model being returned ("End: Final Global Model").

This flowchart outlines the interaction between FL, unlearning strategies, and reinforcement learning for optimizing decisions.

## 5.4 Theoretical Analysis

### 5.4.1 How the RL Method Works

The integration of Deep Q-Networks (DQN) for unlearning decisions in FL systems enhances the global model's performance by improve the trade-off between privacy, accuracy, and computational cost. This section formalizes the problem setup and explains how DQN achieves this improvement.

#### 5.4.1.1 Problem Setup: FL with Unlearning Requests

Let $W^t$ be the global model at communication round $t$, and each client $i$ has a local dataset $D_i$. Each client trains a local model $W_i^t$, and the global model is updated via a weighted average:

$$W^{t+1} = \sum_{i=1}^{N} \alpha_i W_i^t \tag{5.11}$$

where $\alpha_i$ represents the weight of each client's contribution.

When a client requests unlearning, the updated global model can be represented as:

$$W_{\text{new}}^{t+1} = W^{t+1} - \sum_{i=1}^{N} \beta_i \Delta W_i^t \tag{5.12}$$

where $\beta_i$ represents the weight of the unlearning effect for client $i$.

#### 5.4.1.2 Optimization via DQN-based Decision Making

The DQN agent seeks to optimize the unlearning decision by maximizing long-term rewards, which are based on model performance:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{5.13}$$

where $r_t$ is the reward at time $t$, and $\gamma \in [0,1]$ is the discount factor.

The state for each client $i$ is defined as:

$$S_i = \left[C_i, P_i, \text{Comp}_i\right] \tag{5.14}$$

where:

- $C_i$: Contribution to the global model.
- $P_i$: Privacy cost.
- $\text{Comp}_i$: Computational cost.

The DQN selects an **action** $A_i$ from three possible choices:

$$A_i \in \{0,1,2\} \tag{5.15}$$

where:

- $A_i = 0$: No Unlearning.
- $A_i = 1$: Partial Unlearning.
- $A_i = 2$: Complete Unlearning.

The reward function is defined as:

$$r_t = \Delta\text{Accuracy} - \lambda \cdot \Delta P_i \tag{5.16}$$

where:

- $\Delta$Accuracy: Change in model accuracy.
- $\Delta P_i$: Change in privacy risk.
- $\lambda$: Weighting parameter for privacy.

#### 5.4.1.3 Impact on Model Accuracy

Without unlearning, the model's accuracy at time $t+1$ is given by:

$$\text{Accuracy}(W^{t+1}) = f(W^{t+1}) \tag{5.17}$$

When unlearning is applied, the accuracy changes as:

$$\Delta\text{Accuracy} = f(W^{t+1}) - f\left(W^{t+1} - \sum_{i=1}^{N} \beta_i \Delta W_i^t\right) \tag{5.18}$$

The DQN agent minimizes $\Delta$Accuracy, preserving as much of the model's performance as possible.

#### 5.4.1.4 Trade-off Between Privacy and Accuracy

The DQN agent learns to balance the privacy-accuracy trade-off by maximizing the reward function:

$$r_t = \Delta\text{Accuracy} - \lambda \cdot \Delta P_i \tag{5.19}$$

The optimal policy $\pi^*$ is:

$$\pi^*(S_i) = \arg\max_{A_i \in \{0,1,2\}} \left[\Delta\text{Accuracy} - \lambda \cdot \Delta P_i - \mu \cdot \text{Comp}_i\right] \tag{5.20}$$

where $\mu$ is a weighting factor for computational costs.

#### 5.4.1.5 Performance Improvement Proof

Under the optimal policy $\pi^*$, the following performance improvements are ensured:

- Minimized Accuracy Loss: The agent selects *No Unlearning* or *Partial Unlearning* for high-contributing clients, thus minimizing loss in accuracy.
- Privacy Preservation: For clients with high privacy risks, the agent selects Complete Unlearning to reduce privacy exposure while maintaining as much accuracy as possible.
- Efficient Resource Usage: The agent considers computational costs, avoiding unnecessary resource consumption while retaining accuracy.

Thus, the DQN-based method leads to better performance in FL systems, as it balances privacy, accuracy, and computational efficiency.

### 5.4.2 Why DQN Improves Federated Unlearning Performance

The FL with the DQN based unlearning decision can improve performance of the global model as follows: - There is Emerging behavior of the user preferences variability and the quick adaptation to the new dynamics and It is possible that the Deep Q-Network (DQN) would lead to stable performance as itself.

#### 5.4.2.1 Reinforcement Learning for Adaptive Decision Making

Distributed/FL is a very dynamic setting: Clients may have different data distributions, computing resources, and privacy concerns. This variety of factors might not be mitigated simply by a standard, or one-size-fits-all unlearning approach.

In particular DQN is capable of adaptively learning and customizing the unlearning strategy w.r.t. the state of each client, that is, weigh each elements of the state, such as the contribution, the privacy cost and computational cost. This flexibility allows for per–client optimal unlearning action selection and helps to reduce unnecessary data removal while preserving model accuracy. When the contribution of a client is large, and the risk to privacy is small, the DQN will choose No Unlearning, in order to overfit to that client's data for the global model. If privacy risk is more dominant than its contributions, the DQN chooses the Complete Unlearning algorithm in which sensitive information is degraded while the local model is not corrupted at different steps. For the latter, an in-between approach of Partial Unlearningcan be applied, whereby the trade-off between privacy and allowing the data to still contribute to model accuracy is maintained. This state-action flexibility lets unlearning keep only the most important data, and mostly maintain the useful training information, and this directly contributes to model performance.

#### 5.4.2.2 Performance-Metrics Driven Client Contributions Optimization

In the case of FL, some clients' contributions may not be as useful as compared to others. There is a subset of clients with irrelevant or duplicative data, while others have unique contributions. A non-selective unlearning process may delete the data that is necessary for the model to generalize. The DQN agent prioritizes contributions with respect to performance metrics, e.g. the effect of the client on the global model. If the client's contribution is crucial for better accuracy with respect to its impact on the global model, the DQN will minimize the unlearning action and either force to select Partial Unlearning or No Unlearning. Mitigating the impact of irrelevant and harmful data:

By unlearning data from clients that contribute little to model performance, the global model can concentrate learning on more relevant sources, thus improving the accuracy and robustness overall. By filtering what is unlearnt, this selective unlearning process allows the model to keep useful information and prevents it from being disadvantaged by unlearning important information.

#### 5.4.2.3 Tradeoff Privacy vs Accuracy

Privacy of client's data vs. Model accuracy is a desired trade-off in FL. In the absence of a systematic approach to determining what to forget, privacy-driven unlearning request may erase useful data casually, leading to model performance degradation. The DQN agent respects a trade-off between privacy and accuracy: given training on historical actions and their consequences, the DQN agent knows how to handle the cases in which un-learning could lead to a loss in the model's accuracy.

#### 5.4.2.4 Adversarial Attacks Defense

Unlearning can be used by adversarial clients to poison data (b) and degrade the global model. Non-intelligent unlearning system just identifies the misleading examples directly, so that the could not be identified and treated as the attacks and the model performance degrading, if the non-intelligent unlearning system is adopted. The DQN agent can be trained to identify patterns of adversarial behavior (e.g., repeated requests for unlearning from particular clients, or unreasonably large contributions followed by unlearning). This enables it to reduce exposure to potential attacks by being better able to decide when to forget and when to retain. By reducing the unlearning of poisoned or malicious data, the DQN agent may serve to preserve the integrity of the model, and, thus, the performance of the model when subject to adversarial attack.

### 5.4.3 Reinforcement Learning Design and Stability Considerations

The defence mechanism introduced in this chapter uses reinforcement learning to choose appropriate actions for individual samples. The action space contains a small set of discrete defence operations, including the addition of controlled noise, the reduction of sample influence and the temporary omission of samples from the aggregation process. Since these actions are discrete and the state representation is relatively compact, methods that rely on learning action values provide an effective solution.

We adopt Deep Q Networks as the main reinforcement learning method. Deep Q Networks estimate the expected value of each discrete action directly and do not require sampling from a separate policy. This matches our setting, where the reward signal reflects the improvement in robustness and privacy produced by different defensive choices. Deep Q Networks perform reliably when the number of possible actions is small, which aligns with the design of our defence system.

Other reinforcement learning methods were also examined, including Proximal Policy Optimisation and Advantage Actor Critic. These approaches are able to represent more flexible policies and are often used when the action space is large or continuous. However they introduce additional challenges that are less suited to the requirements of a sample level defence. Proximal Policy Optimisation relies on clipped objectives and trust region ideas that require careful tuning to prevent unstable updates. In settings where the reward distribution varies across training rounds, such as federated learning, this method may require repeated adjustment of training parameters before reaching stable behaviour. Advantage Actor Critic methods experience variance in the estimation of advantages and often require additional regularisation terms. They also maintain both an actor and a critic network, which increases the computational cost.

In comparison, Deep Q Networks offer a simpler learning objective and do not require maintaining two separate components. The update rule is efficient and the computational cost remains low, which is important because the defence must evaluate actions for a large number of samples. For these reasons Deep Q Networks represent a practical compromise between model expressiveness, computational efficiency and training stability.

Training stability is an important factor because reinforcement learning methods can be sensitive to changes in data distribution and can exhibit oscillation or slow convergence. To address these issues we employ several techniques that are well established in reinforcement learning. First, we use an experience replay buffer to reduce the correlation among training samples. The agent learns from batches of past transitions rather than from consecutive observations, which lowers the variance of the estimated action values. Second, we maintain a separate target network that is updated at fixed intervals. This stabilises the target values used in learning and prevents rapid shifts in the value estimates. Third, we normalise the rewards to keep them within a consistent range across training stages. Finally, we use a gradual exploration schedule in which the probability of random actions decreases slowly over time.

Together these choices allow the reinforcement learning agent to operate in a stable and predictable manner. The combination of a straightforward value learning method

and standard stabilisation techniques ensures that the agent can identify effective defence strategies without suffering from the instability that often appears in more complex reinforcement learning algorithms.

## 5.5 Experiment

### 5.5.1 Experiment Setup

FL Framework: The setup consists of a central server coordinating with multiple clients (we set clients from 10 to 200), each with its own local dataset. At each communication round, the clients train local models and send updates to the server, which aggregates these updates to form a global model.

Here are datasets we have tested:

- CIFAR-10: A popular benchmark dataset with $60,000$ $32 \times 32$ images across 10 classes, widely used for testing image classification algorithms, especially in deep learning.
- MNIST: A classic dataset of $28 \times 28$ grayscale images of handwritten digits $(0-9)$, commonly used for digit recognition tasks.

Here are baseline methods we have compared:

- Random Unlearning: This method randomly selects clients for unlearning without considering their contributions, privacy costs, or computational impact.
- Retrain from Scratch: This method retrains the entire model from scratch after removing poisoned or undesired data.
- No Unlearning: This method serves as a control baseline, where no action is taken to remove poisoned or undesired data from the model.

Poisoning attacks:

- Label Flipping Attack: Adversarial clients intentionally mislabel a subset of their data (e.g., flipping labels of one class to another) to misguide the global model during training[53]. [116].

- Backdoor Attack: Adversaries insert specific patterns (e.g., a pixel patch) into their training data, ensuring that the global model associates the pattern with a particular label[31].

To evaluate the effectiveness of the DQN-based unlearning method, we use the following metrics:

- Model Accuracy: The accuracy of the global model on the test set, measuring how well the model retains performance after unlearning actions. [116].
- Attack Success Rate(ASR): Measures the effectiveness of poisoning attacks by quantifying the percentage of manipulated data points that successfully influence the model. A lower ASR indicates a more robust system.
- Computational Cost: The time and memory usage required to perform each unlearning action, evaluating the efficiency of the method compared to baselines. [116].

Krum, Bulyan, Trimmed Mean, and Median are aggregation methods used in FL to mitigate the impact of outliers or adversarial data, especially against poisoning attacks.

- Bulyan: Builds on Krum by selecting a subset of updates using Krum, then applies a robust aggregation rule, making it more resilient to adversarial attacks [37].
- Trimmed Mean: Discards the highest and lowest updates and computes the mean of the remaining ones, removing extreme values caused by noise or adversaries [116].
- Median: Takes the median of all updates, providing robustness against outliers, as the median is less affected by extreme values compared to the mean [116].

### 5.5.2 Results

In this sub-section, we present the experimental results of our methods with the comparison of baseline methods. Firstly, we set two groups of simple controlled experiments to prove the impact of unlearning and the performance of two attack methods. And then, according to the characteristic of FL and poisoning attack, we set a different number of clients and a different proportion of poisoning. Besides, we also record the computational cost of each method.

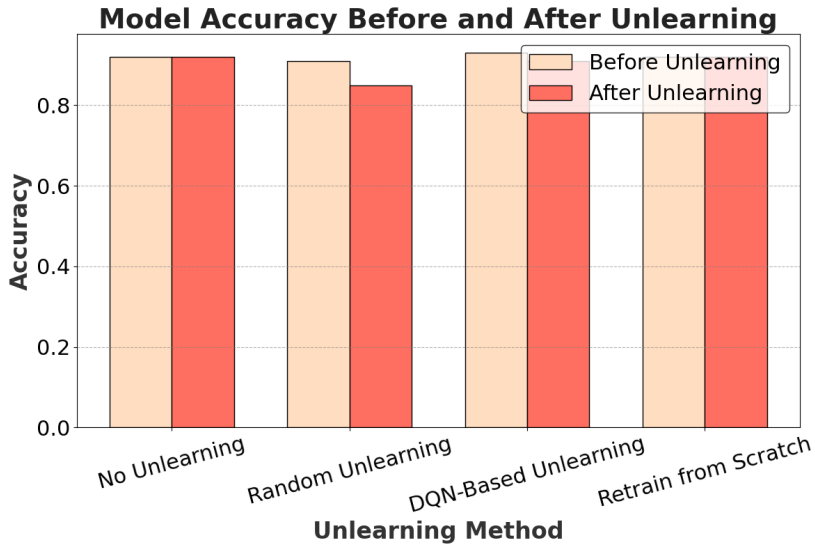


Figure 5.3: Model Accuracy.

Fig. 5.3 compares the model accuracy before and after applying different unlearning methods. Our method demonstrates the highest post-unlearning accuracy, indicating its ability to selectively remove poisoned contributions while preserving model utility. Retrain from scratch achieves comparable accuracy, but at a significantly higher computational cost. Random unlearning shows moderate improvements, whereas no unlearning results in the lowest accuracy, as it fails to address the poisoned data's adverse effects.

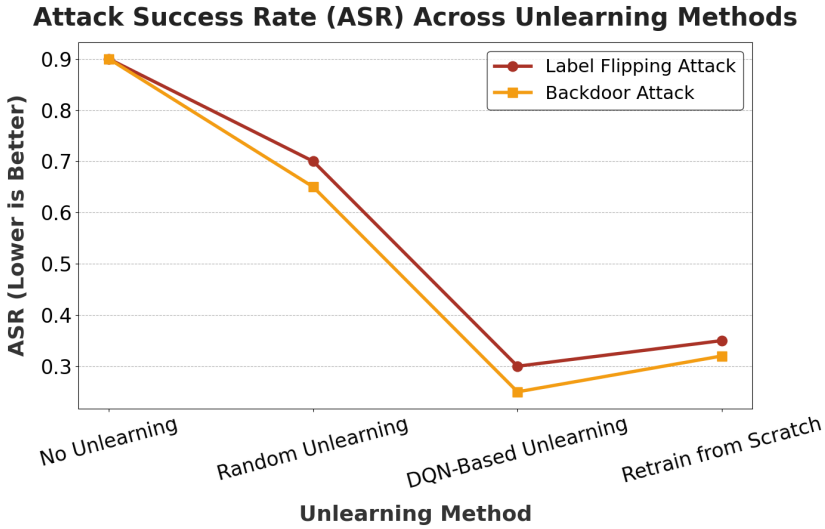


Figure 5.4: Attack Success Rate (ASR).

Fig. 5.4 shows ASR of two attack type. We can see that our method has best perfor-

mance compare with other methods. For no unlearning, it can not clear the poisoned data in the model, random unlearning can clear part of poisoned data and the train from scratch and our method can eliminate most influence from poisoned data, moreover, our method can also filter some low quality clients, which can make our method perform better.

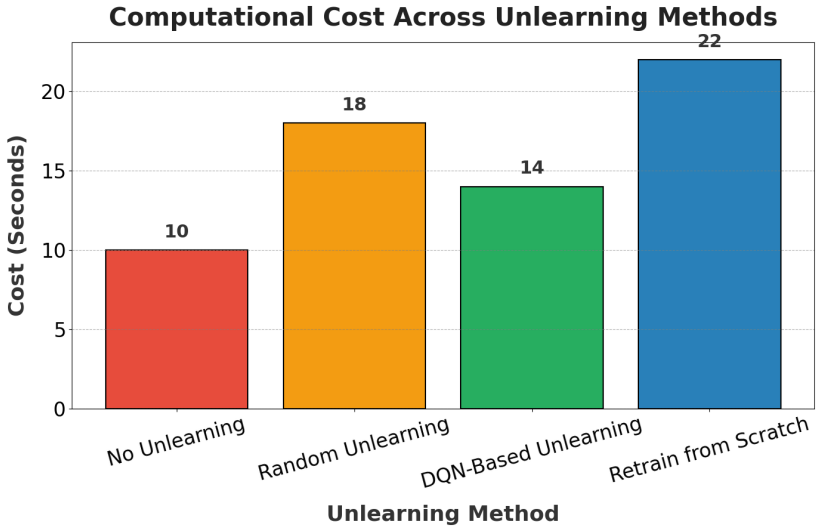


Figure 5.5: Computational Cost.

Fig. 5.5 compares the computational cost of the unlearning methods, highlighting the trade-offs between effectiveness and efficiency. Our method is the lowest computational cost among the effective methods, showcasing its practicality for large-scale deployments. By selectively targeting poisoned data, it avoids the overhead of retraining while maintaining strong attack mitigation.

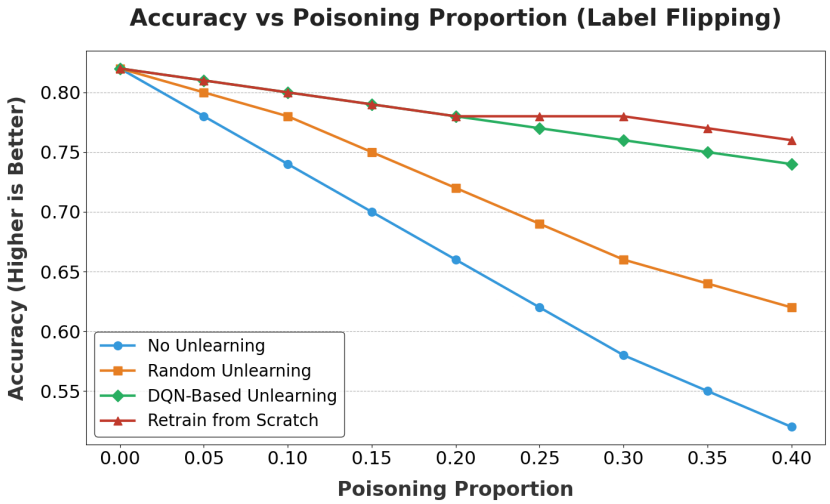


Figure 5.6: Accuracy under Label Flipping

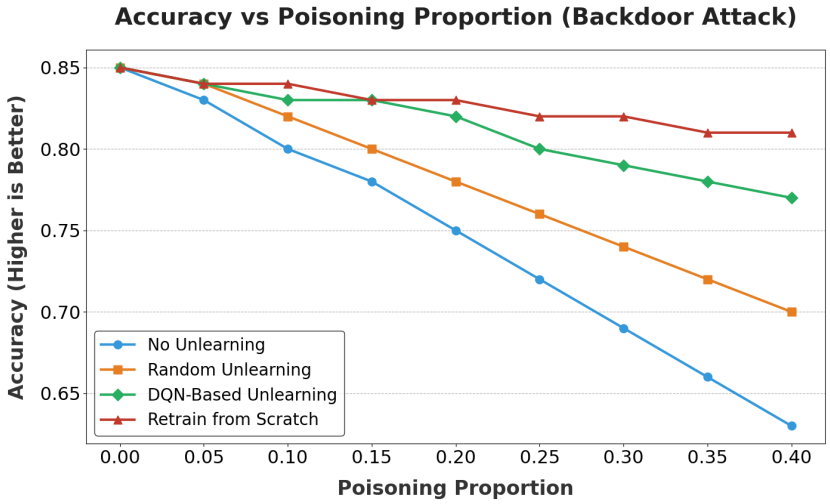


Figure 5.7: Accuracy under Backdoor

Fig. 5.6 and Fig. 5.7 illustrate the effectiveness of the unlearning methods in mitigating poisoning attacks as the number of clients increases in two type attacks. Random Unlearning shows moderate improvements, as it occasionally removes poisoned data by chance. However, its lack of strategy limits its overall effectiveness. No Unlearning maintains a consistently high ASR, indicating its inability to counteract poisoning attacks even as the number of clients increases. Our method achieves the most significant reduction in ASR for both types of attacks. Its ability to dynamically identify and unlearn

poisoned contributions ensures robust mitigation, with ASR decreasing steeply as client diversity improves. Retrain from Scratch performs similarly to our method in reducing ASR but at a much higher computational cost.

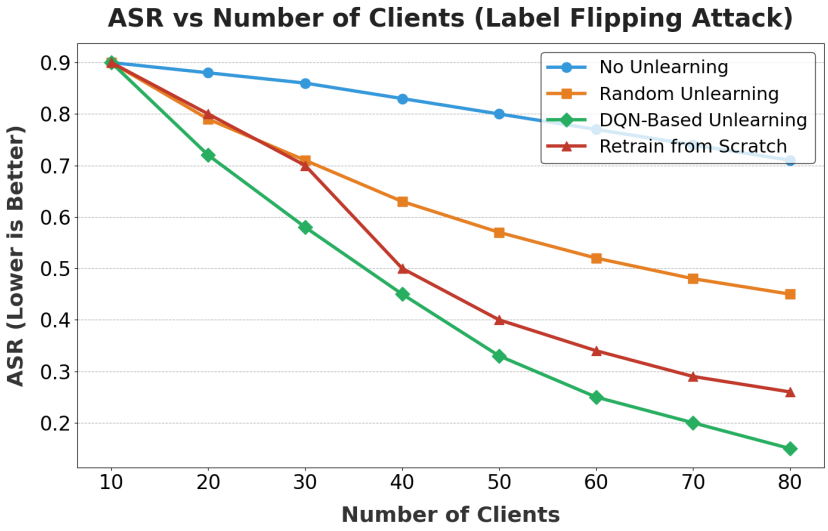


Figure 5.8: ASR under Label Flipping

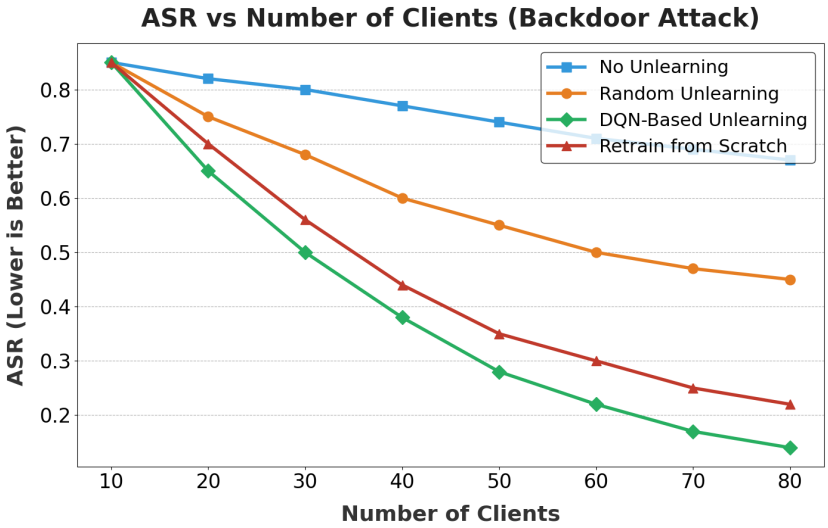


Figure 5.9: ASR under Label Backdoor

Fig. 5. and Fig. 5.9 depict the relationship between model accuracy and the proportion of poisoned data for label flipping and backdoor attacks. For both types of attack, as the proportion of poisoned data increases, the accuracy of the model decreases across

all methods. However, our method maintains the highest accuracy, closely matching retrain from scratch. This highlights its ability to preserve model performance while addressing poisoning attacks. Random unlearning demonstrates moderate resilience, as it reduces poisoned contributions by chance. In contrast, no unlearning results in the steepest accuracy decline, particularly under higher poisoning proportions, indicating its ineffectiveness in countering such threats. The results further validate our method's ability to balance attack mitigation and model utility, outperforming other approaches in preserving accuracy even at high levels of poisoning.

Table 5.1: Results on Poisoning Attacks - MNIST

| **Defense Method** | **Train Accuracy (%)** | **Test Accuracy (%)** |
|---|---|---|
| No Defense | 78.3 | 65.7 |
| Our Method | 96.7 | 94.4 |
| Krum | 84.2 | 79.3 |
| Bulyan | 91.6 | 87.1 |
| Trimmed Mean | 94.5 | 88.2 |
| Median | 93.8 | 86.4 |

Table 5.2: Results on Poisoning Attacks - CIFAR-10

| **Defense Method** | **Train Accuracy (%)** | **Test Accuracy (%)** |
|---|---|---|
| No Defense | 57.2 | 51.4 |
| Our Method | 66.8 | 64.5 |
| Krum | 60.5 | 53.2 |
| Bulyan | 55.3 | 52.1 |
| Trimmed Mean | 58.4 | 55.8 |
| Median | 57.9 | 54.9 |

Table 5.1 and table 5.2 show the results across the MNIST and CIFAR-10 datasets highlight the effectiveness of various defense mechanisms against poisoning attacks. Our Method consistently outperforms all other approaches, achieving the highest training and testing accuracy.

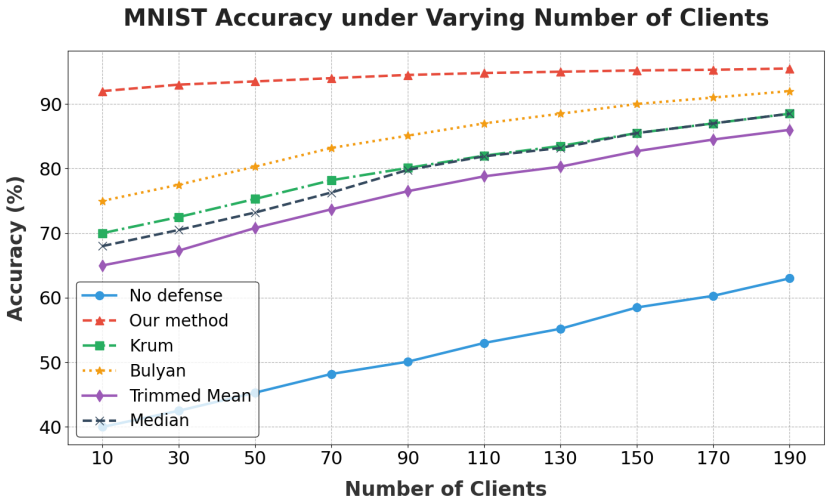


Figure 5.10: Accuracy vs. Number of Clients on MNIST

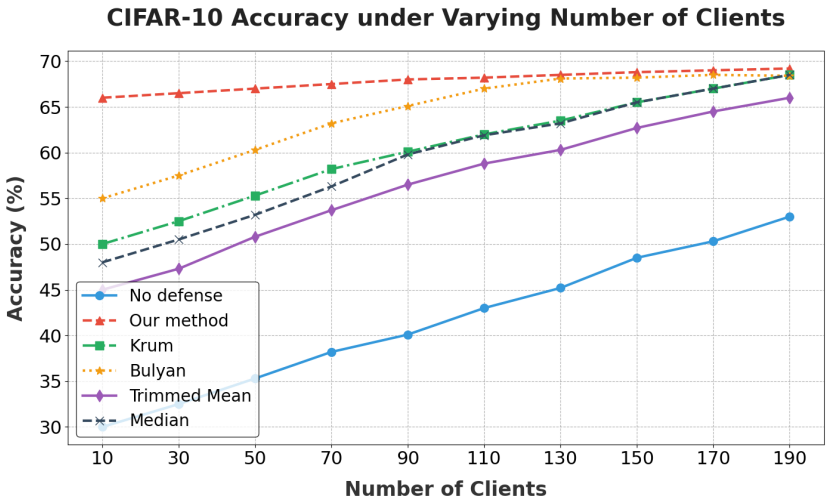


Figure 5.11: Accuracy vs. Number of Clients on CIFAR-10

Fig. 5.10 and Fig. 5.11 examine the impact of increasing poisoning proportions. As the proportion of poisoned data rises, accuracy declines across all methods and datasets. Our method again stands out, showing resilience with a relatively slower decline in accuracy compared to others. No Defense experiences the steepest accuracy drop, underscoring the need for robust defenses. Methods like Bulyan and Trimmed Mean achieve better results than Krum and Median but remain less effective than Our Method.

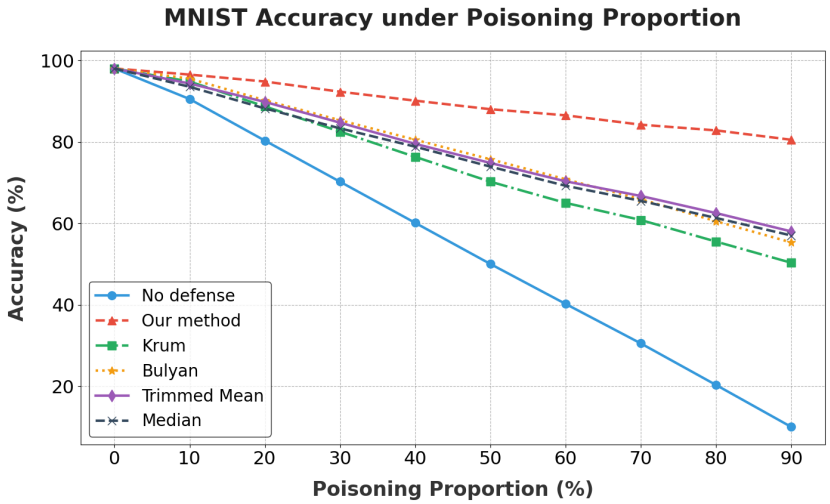


Figure 5.12: Accuracy vs. Poisoning Proportion on MNIST

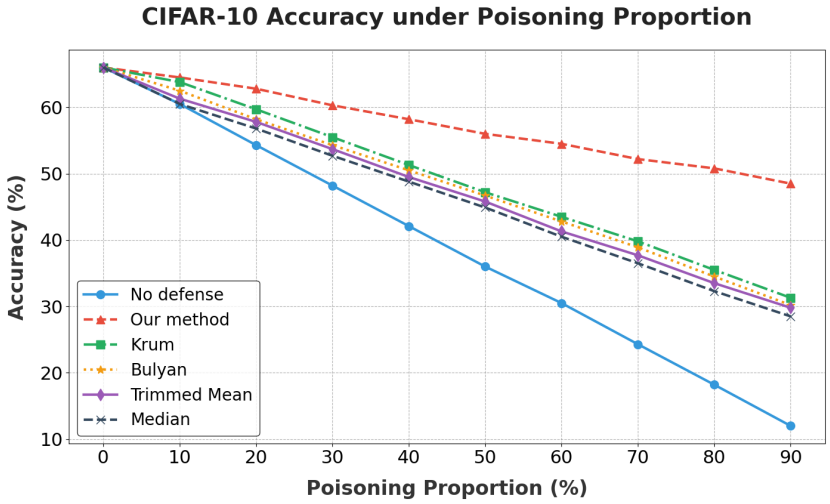


Figure 5.13: Accuracy vs. Poisoning Proportion on CIFAR-10

Fig. 5.12 and Fig. 5.13 have the curves of root-mean-squared as a function of the size of the set of clients. In both data, No Defense always achieves the worst accuracy at the beginning of all the defenses, and its performance becomes better as the number of clients involved increases. In comparison, Our Method shows very stable and robust prediction performance and less variations (small plateaus), indicating its strong robustness and scalability. The other methods: Krum, Bulyan, Trimmed Mean, and Median show the medium performance on a large number of clients.

## 5.6 Related Work

Federated unlearning aims to remove the influence of specific client data from a trained federated model while maintaining overall model performance and compliance with privacy regulations. Existing approaches can be categorized into exact unlearning and approximate unlearning. Exact unlearning methods, such as model retraining from scratch without the designated data [10, 124], ensure complete removal but incur prohibitive computational costs. Approximate unlearning techniques leverage techniques such as model inversion, gradient perturbation, and weight adjustments [30, 119], providing a trade-off between computational efficiency and unlearning efficacy. However, these methods often struggle with performance degradation and may be vulnerable to adversarial interference. Our work extends this line of research by introducing a reinforcement learning-based approach to dynamically optimize unlearning actions, balancing efficiency, privacy, and model stability.

The decentralized nature of FL systems makes them inherently vulnerable to adversarial attacks. Adversarial attacks Eg., model poisoning [2, 24] and inference [89] can abuse unlearning to change model behavior or learn sensitive content. Namely, unlearning operations may inadvertently leak private data, as shown in gradient leakage and model inversion studies [136]. Additionally, attackers can design poison updates to cause a model to perform badly or make wrong decisions which is a great security challenge[41]. Our method mitigates these two threats by adopting an adaptive unlearning strategy to prevent the model performance from decreasing and enhance the robustness against adversarial perturbations.

For example, reinforcement learning (RL) has been extensively used to learn how to optimize different aspects of FL, such as client selection [74], resource allocation [67] and secure aggregation [62]. Meanwhile, there are some recent research works that investigate the possibility of utilizing deep reinforcement learning techniques, such as deep Q-networks (DQN) and policy gradient methods in enabling FL methods with adaptability to dynamic environments [104][114]. Our work extends these principles by incorporating RL in making unlearning decisions, so as to enable adaptive and efficient client selection for unlearning processes in terms of impact, privacy sacrifice and computational overhead. To the best of our knowledge, we are the first to propose incorporating reinforcement learning into federated unlearning to mitigate the performance and security issues adaptively.

It is still a big challenge to not compromise privacy in the unlearning process. There

exist studies that investigate differential privacy[21] and secure multiparty computation [8] as means of privacy amplification in a federated setting. However, their usage for federated unlearning is still at the exploratory stage, as most of the existing work mainly concentrate on static privacy guarantees rather than dynamic unlearning operations[122]. We complement these efforts by developing an adaptive framework that reduces the risk of privacy leakage while maintaining computational efficiency.

## 5.7 Conclusion

Our contribution is an adaptive and efficient approach to defending poisoning attacks for FL, which is an unlearning scheme powered by DQN. The proposed approach dynamically selects unlearning actions by using reinforcement learning and does so based on client-dependent measure, which balance the trade-off between accuracy, security and computation burden. We run comprehensive experiments to compare the performance of DQN-based unlearning with the baseline algorithms: no unlearning, random unlearning and retraining from scratch. The DQN-based unlearning framework we propose is a milestone to advanced techniques of FL, providing a ready, efficient, and principled approach to dealing with poisoning attacks, which is provably privacy-preserving and secure.

CHAPTER 6

# LEARNING TO DEFEND WHAT MATTERS: DATA IMPORTANCE-AWARE REINFORCEMENT LEARNING FOR MULTI-ATTACK ROBUSTNESS

This chapter proposes a data-importance-aware RL defense for multi-attack robustness. It explains sample profiling, attack signal construction, and adaptive defense execution, and presents theoretical justification and extensive empirical results across backdoor, model stealing, and membership inference attacks.

## 6.1 Introduction

The increasing dependence on machine learning models opens up new and even more critical vulnerabilities [99]. In traditional machine learning systems, they introduce the approach of learning based on data and hence also make them manipulable during not only training but also inference attacks. Many attacks are now increasingly adopted, such as backdoor attacks which inject hidden triggers inside training data to insert targeted misclassification [81], model stealing attacks that reconstruct the functionality of a model through API queries [76], and membership inference attacks that breach training data privacy [36]. These attacks largely exploit the model's inductive biases and reliance on training data, effectively transforming its learning ability into a surface for adversarial manipulation.

One of the most critical but underexploited dimensions of attacks is collecting a

detailed understanding of individual data and its importance for the threats they pose [107]. Not all data are equal weighted in the final output calculation. Therefore, some high-influence samples have a higher propensity toward decision-boundary shaping; some are much more frequently queried, while others are very prone to overfitting. Typically, such asymmetries are leveraged by attackers; they strike with samples of high importance at their disposal to maximize their impact with minimum effort [52]. For example, an adversary needs only a few high-influence samples poisoned to inject statistically sound yet functionally harmful backdoors. With such samples being sensitive, the chances of membership inference attack success go up. However, such high-importance samples can also be detected to uplift the defense and decrease the extra cost that is unneeded.

Despite this, most existing defense mechanisms treat all inputs uniformly and apply static or attack-specific countermeasures. These include noise injection [65] where random perturbations are added to input or model parameters to reduce the success rate of adversarial attacks but usually with a general loss of model performance and lack precision in targeting high-risk samples, adversarial training [120], one of the most widely adopted techniques, involves training models on adversarial examples to improve robustness but often with expensive computation, threshold-based filtering [19] which removes inputs that deviate significantly from expected patterns but often suffers from poor detection accuracy under adaptive or stealthy attacks, or input sanitization [91] including methods such as autoencoder reconstruction or feature squeezing, attempts to sanitize malicious inputs before inference, but these techniques may unintentionally remove informative features from benign samples, reducing accuracy and failing under sophisticated threats.

Several defenses have been proposed for specific attacks. For instance, differential privacy mechanisms mitigate MIAs by injecting calibrated noise[39]; gradient masking defends against model stealing by reducing information leakage.

However, most defenses suffer from same limitations which undermine their robustness and efficiency. Firstly, most defense methods assume that they already have the prior knowledge of a specific attack. This assumption limits their flexibility, as the attackers in real-world often employ adaptive or unforeseen attack strategies that deviate from the modeled threat. As a result, such defense methods may become inapplicable or ineffictive when confronted with unpredictable attack. Besides, these defenses usually obey a static protection strategy with predefined threshold or fixed parameters. This ignores the face that different samples exhibit varying degrees of vulnerability depend-

ing on their semantic features or contribution to model behavior. Accordingly, uniform defenses can lead to two types of failure: excessive protection for low-risk inputs that degrades model utility, or insufficient protection for high-risk inputs that remain exposed to exploitation. Lastly, most of the current defenses fails to reason about input-level vulnerability. Without the ability to assess which data points are more sensitive, defenses can not provide prioritize protection where it is most needed. Consequently, misallocation of defensive resources can lead to both inefficient utilization and degraded protection performance. This limitation becomes particularly evident when facing adversaries that specifically target sensitive or high-value data.

Above limitations show that current deep learning models are vulnerable to all forms of adversarial attacks that exploit the data-dependent nature of the model. This manifests itself as an attack that comes in different guises, such as backdoor injection, model stealing, or membership inference, and all share a similar pattern: attacks operate on well-chosen inputs that exploit a weakness in the model's behavior.

This makes it remarkably challenging to implement heterogeneous risk across individual data samples-that is, not all inputs are equally vulnerable or equally impactful. Some points are just much more likely to experience exploitation by adversaries than to be influential; high frequency in the access path of updates to the model;and high probability of being updated by adversarial examples. Traditional defenses apply static or global protections, which renders it impossible to allow for this variation and hence protect high-risk samples insufficiently and degrade performance with low-risk ones unnecessarily. We aim to design a defense mechanism that:

- Estimates per-sample vulnerability in a principled way;
- Adapts protection strategies based on each sample's risk level;
- Generalizes across different types of attacks, without relying on prior knowledge of the threat model.

To achieve the goal, we present a dynamic defense framework based on reinforcement learning that takes into consideration the differing nature of each sample. Since one sample's vulnerability is influenced by multiple factors, such as how important it is to models, what the quality of it and how well the model learned it, we construct a general attack signal that integrates these dimensions. At the core of our design is the formulation of a general attack signal as a function of three principal metrics, namely: importance score for the sample, noise level at the current time, and model

loss. This signal allows both the attacker and the defender to properly reason about the vulnerability of the sample. Based on this signal, our defense agent learns the choice of optimal protection strategy among noise injection, suppression of gradient contribution, or masking access through continuous interaction and feedback.

We applied and evaluated our framework across backdoor injection, model stealing, and membership inference attack scenarios. Our experimental results show that our method consistently reduces attack success rates while maintaining model performance on benign data. Data importance is integrated in the defense loop, and the ability to make per-sample decisions dynamically opens up a way not just to secure the current ML systems but to do it in a generalizable and effective manner. Our main contributions are as follows.

- 1. We propose a unified, reinforcement learning-based defense framework that dynamically mitigates multiple types of attacks including backdooring, model stealing, and membership inference by making per-sample decisions during both training or inference.
- 2. We introduce a novel attack signal formulation called sample importance score, current noise level, and model loss. This signal allows both attackers and defenders to think about sample vulnerability in a principled manner.
- 3. We perform extensive evaluations across diverse attack scenarios. Our results show that Adversarial training significantly reduces attack success rates while maintaining benchmark accuracy on clean data. Our approach significantly outperforms several stat baselines, forming an ensemble. Single-attack baselines are outperformed as well.

## 6.2 Background

### 6.2.1 Vulnerabilities in Machine Learning Models

Modern machine learning models, particularly deep neural networks, are vulnerable to various adversarial threats. These include backdoor attacks, model stealing, and membership inference attacks (MIAs).

In **backdoor attacks**, an adversary poisons the training set by injecting samples $(\mathbf{x}_{\text{trig}}, y_{\text{target}})$, where $\mathbf{x}_{\text{trig}} = \mathbf{x} + \delta$ contains a trigger $\delta$, and $y_{\text{target}}$ is a target class[86]. The attack objective is:

$$f(\mathbf{x}_{\text{trig}}) = y_{\text{target}} \quad \text{while} \quad f(\mathbf{x}) = y \text{ for clean } \mathbf{x}. \tag{6.1}$$

In **model stealing attacks**, the adversary queries the model to construct a surrogate model $f'(\cdot)$ that approximates the behavior of the victim model $f(\cdot)$ [84]:

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\text{query}}}\left[\|f(\mathbf{x}) - f'(\mathbf{x})\|^2\right] \to \min. \tag{6.2}$$

In **membership inference attacks**, the adversary aims to infer whether a sample $\mathbf{x}$ was used in training. Given a model prediction $f(\mathbf{x})$, the adversary estimates[89]:

$$\Pr[\mathbf{x} \in \mathcal{D}_{\text{train}} \mid f(\mathbf{x}), y] > \tau, \tag{6.3}$$

where $\tau$ is a decision threshold.

These attacks expose the model's generalization behavior and exploit the data-dependent nature of deep learning models.

### 6.2.2 Limitations of Existing Defenses

Let $\mathcal{A}$ denote an attack and $\mathcal{D}$ the dataset. Most defenses minimize the expected risk under a fixed transformation $\mathcal{T}$:

$$\min_f \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[\mathcal{L}(f(\mathcal{T}(\mathbf{x})), y)], \tag{6.4}$$

ignoring the fact that some $\mathbf{x}$ may be more at risk than others.

### 6.2.3 Data Importance in Attack and Defense

Recent work highlights that not all training samples contribute equally to the learned model. Some samples have outsized influence on the model's predictions or parameters [107]. Let $f(\cdot)$ be a model trained on dataset $\mathcal{D}$, and let $\mathcal{L}_{\text{total}}$ be the empirical loss:

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{n} \mathcal{L}(f(\mathbf{x}_i), y_i). \tag{6.5}$$

The **importance score** $S(\mathbf{x}_i)$ of a sample $\mathbf{x}_i$ can be defined as the change in total loss if the sample is removed:

$$S(\mathbf{x}_i) = \mathscr{L}_{\text{total}} - \mathscr{L}_{\text{total}}^{(-i)}. \tag{6.6}$$

This notion of sample importance has implications for both attackers and defenders:

- Attackers can target high-$S(\mathbf{x})$ samples for maximum disruption or extraction.
- Defenders can prioritize high-$S(\mathbf{x})$ samples for protection.

Existing defense strategies rarely leverage this information, treating all samples equally regardless of their influence. In this work, we address this gap by designing a defense mechanism that explicitly models and responds to per-sample importance.

## 6.3 Threat Model

In our work, we consider a general threat model which the learning system is exposed to multiple types of adversarial threats, specifically backdoor attacks, model stealing attacks, and membership inference attacks.

### 6.3.1 Backdoor Attack

- Attacker's goal: To inject some malicious triggers into a subset of training data. When it works, the model misclassifies inputs containing the trigger into a target label while maintaining high accuracy on clean data.
- Attacker's Capabilities: The attacker can control a small portion of the training data. They can insert inputs with specific features or labels but cannot directly modify model weights or observe gradients.
- Attacker's Knowledge: The attacker operates under a black-box setting with no access to model parameters, gradients, or training procedures.

### 6.3.2 Model Stealing Attack

- Attacker's Goal: To extract a surrogate model that approximates the behavior of the target model by issuing a sequence of carefully crafted input queries and collecting the corresponding outputs.

- Attacker's Capabilities: The attacker can query the target model multiple times and collect the associated prediction outputs (soft labels or logits), which they use to train a local copy that replicates the target model's decision boundary.
- Attacker's knowledge: The attacker does not have access to model internals (architecture, parameters, training data). They rely purely on input-output observations in a black-box manner.

### 6.3.3 Membership Inference Attack

- Attacker's Goal: To determine whether a specific data sample was used during the training phase of the target model.
- Attacker's Capabilities: The attacker can submit data points to the target model and observe its confidence scores or output probabilities. Additionally, the attacker may train shadow models to simulate the behavior of the target model under known membership conditions.
- Attacker's knowledge: They assume access to a shadow dataset drawn from a similar distribution and can query the target model freely.

## 6.4 Problem Statement

Formally, let $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ be the training dataset, and let $f_\theta$ be a deep model parameterized by $\theta$. An adversary $\mathscr{A}$ aims to compromise $f_\theta$ by maximizing an attack objective $\mathscr{L}_{\text{attack}}$ through manipulation or observation of a subset of inputs $\mathscr{S}_{\mathscr{A}} \subseteq \mathscr{D}$:

$$\mathscr{A}^* = \arg\max_{\mathscr{A}} \mathbb{E}_{\mathbf{x} \in \mathscr{S}_{\mathscr{A}}}[\mathscr{L}_{\text{attack}}(f_\theta(\mathbf{x}))] \tag{6.7}$$

Our goal is to learn a defense policy $\pi(s)$ that, given a sample state $s$, selects a defensive action $a \in \mathscr{A}_{\text{defense}}$ to minimize both attack success and model performance degradation:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\mathbf{x} \in \mathscr{D}}[-\mathscr{L}_{\text{attack}}(f_\theta(\mathbf{x})) - \lambda \cdot \mathscr{L}_{\text{clean}}(f_\theta(\mathbf{x}), y)] \tag{6.8}$$

Here, $\mathscr{L}_{\text{clean}}$ is the standard classification loss, and $\lambda$ controls the trade-off between robustness and accuracy.

To guide both the attacker and the defender, we introduce a sample-level *attack signal* $T(\mathbf{x})$, defined as a function of three metrics.

$$T(\mathbf{x}) = \lambda \cdot S(\mathbf{x}) \cdot (1 - N(\mathbf{x})) \cdot L(\mathbf{x}) \tag{6.9}$$

- $S(\mathbf{x})$: importance score of the sample;
- $N(\mathbf{x})$: current noise level;
- $L(\mathbf{x})$: current model loss on the sample.

This signal quantifies the vulnerability or possible attack on a given input, leaving the defense policy to be able to reason which samples to protect and just how. The core issue is to implement an adaptive defense policy that leverages this attack signal for mitigation of multiple attack types at a per-sample granularity without the need for any tuning based on attacks and without making an assumption about it.

## 6.5 Methodology

We introduce a dynamic defense approach, where the reinforcement learning methodology automatically adapts the requirement of protection that is offered against different kinds of adversarial attacks on a sample-wise basis. At its core, our belief is that different samples in any dataset have varied levels of vulnerability to different types of attacks and hence need differentiated treatment defensively. On our part, we built a pipeline (1) sample profiling, where each input is analyzed for potential vulnerability; (2) construction of an attack signal, which quantifies the exposure risk using principled metrics; (3) execution of a defense policy via a trained RL agent that selects appropriate actions; (4) deployment of the selected defense action to mitigate risk, and (5) policy update through feedback rewards, incorporating both attack outcomes and model utility. The entire workflow is shown in Fig.6.1.

The basic components and the sequential flow of our dynamic defense framework are shown in Fig.6.1. Although being simple in nature the various stages incorporate several mechanisms which are explained in following subsections. In the Sample Profiling phase, we analyse per-input properties, including the sensitivity of the gradient and the contribution to the loss, to determine exploitability. The Attack Signal Construction phase quantifies these intuitions into a single vulnerability score computed based on the

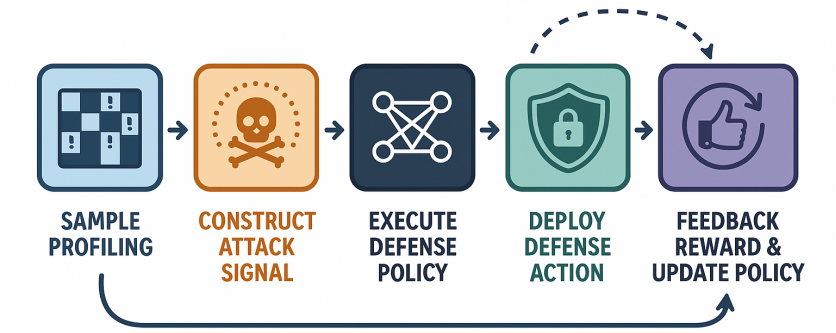


Figure 6.1: Workflow of Proposed Method

weighted sum of sample importance, observed noise level and model loss (this becomes the attack signal) - which serves as a measure of the per-sample likelihood of being attacked. In the policy execution stage, the reinforcement learning agent interprets the attack signal, and the defender acts an appropriate defense action from a set of known defense actions (e.g., injecting perturbations, zeroing out gradient contributions, and conducting masking). The Defense Deployment phase can deploy the chosen intervention onto the input or its update. Aggregating all together, the Feedback and Policy Update stage utilizes both attack success metrics and the downstream model utility to update the RL policy based on reward signals. This integration of actional, learning-based adaptation throughout the pipeline is the source of that the picture is not just illustrative but captures the operational depth of the system.

### 6.5.1 Sample Profiling

Sample profiling aims to build a summary representation for a single data point that is compact and yet very informative. A representation that captures influence, risk, and defense history. Each sample $\mathbf{x}$ is encoded into a normalized state vector.

$$\mathbf{s}(\mathbf{x}) = [\mathscr{I}(\mathbf{x}),\ \mathscr{N}(\mathbf{x}), \tag{6.10}$$

where:

- $\mathscr{I}(\mathbf{x})$ is the sample **importance score**,
- $\mathscr{N}(\mathbf{x})$ is the current **noise level**,
- $\mathscr{L}(\mathbf{x})$ is the **normalized model loss**.

#### 6.5.1.1 Importance Score $\mathscr{I}(\mathbf{x})$

We adopt a Shapley value approximation inspired by KNN-Shapley [107] to quantify the contribution of $\mathbf{x}$ to model performance. The importance is estimated as:

$$\mathscr{I}(\mathbf{x}) \approx \frac{1}{k} \sum_{i=1}^{k} [U(\mathscr{D}_i \cup \{\mathbf{x}\}) - U(\mathscr{D}_i)] \tag{6.11}$$

where $U(\cdot)$ denotes the model utility (e.g., validation accuracy), and $\mathscr{D}_i$ are randomly sampled local neighborhoods. This formulation captures both representativeness and influence on generalization.

#### 6.5.1.2 Noise Level $\mathscr{N}(\mathbf{x})$

The noise level reflects the cumulative perturbation applied to the sample. We define it as:

$$\mathscr{N}(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{x}^{\text{def}}\|_2}{\|\mathbf{x}\|_2 + \epsilon} \tag{6.12}$$

where $\mathbf{x}^{\text{def}}$ is the defended (perturbed) version of the sample, and $\epsilon$ is a small constant for numerical stability. This ratio quantifies relative distortion due to prior defense actions.

#### 6.5.1.3 Model Loss $\mathscr{L}(\mathbf{x})$

We compute the cross-entropy loss on the sample and normalize it across each mini-batch:

$$\mathscr{L}(\mathbf{x}) = -\log p_\theta(y \mid \mathbf{x}) \tag{6.13}$$

$$\mathscr{L}_{\text{norm}}(\mathbf{x}) = \frac{\mathscr{L}(\mathbf{x}) - \min \mathscr{L}}{\max \mathscr{L} - \min \mathscr{L}} \tag{6.14}$$

This normalized loss captures model uncertainty and can highlight under-trained or overfitted points.

### 6.5.2 Attack Signal Construction

For heterogeneous attacks, we use a unified, interpretable scalar metric termed attack signal that which quantifies adversarial vulnerability per input and at the same time sets a standard reference state input for the RL-based defense agent.

As shown in [107], data samples that are highly influential to model learning or poorly predicted are disproportionately more vulnerable to a range of attacks, including backdoor, model stealing, and membership inference. Furthermore, if a sample has already been perturbed or obfuscated, its effective risk decreases. These observations motivate the use of a signal that jointly reflects:

- The importance of a sample to model performance;
- Its current defensive exposure (noise level);
- Its predictive uncertainty or confidence.

Given a sample $\mathbf{x}$ and its profiling vector $\mathbf{s}(\mathbf{x}) = [\mathscr{I}(\mathbf{x}),\ \mathscr{N}(\mathbf{x}),\ \mathscr{L}(\mathbf{x})]$, we define the attack signal as:

$$T(\mathbf{x}) = \lambda \cdot \mathscr{I}(\mathbf{x}) \cdot (1 - \mathscr{N}(\mathbf{x})) \cdot \mathscr{L}(\mathbf{x}) \tag{6.15}$$

where:

- $\mathscr{I}(\mathbf{x}) \in [0,1]$ is the sample importance score;
- $\mathscr{N}(\mathbf{x}) \in [0,1]$ is the current noise level;
- $\mathscr{L}(\mathbf{x}) \in [0,1]$ is the normalized model loss;
- $\lambda > 0$ is a global scaling factor.

This formulation satisfies several desirable properties:

- **Boundedness:** $T(\mathbf{x}) \in [0, \lambda]$, ensuring numerical stability.
- **Zero-sensitivity:** If a sample is well-learned or strongly defended, its signal value approaches zero.
- **Compositionality:** Each dimension of the signal is individually interpretable and empirically observable.

The sensitive signal $T(\mathbf{x})$ is intended to assess the vulnerability of a sample by considering the sample's importance to the model, the available protection against the model's adversarially attacked loss, and the uncertainty with which the model makes its predictions. This formulation arises from empirical findings that relatively more important but under-defended and weakly learned samples lie at higher risks of attacks compared to any specific attack risk dimension-high importance, low defense, and low learning level all intersect the vulnerability of very diverse attacks. We use importance-normalized loss-inverse noise-level multiplicative structure to have the signal increase visibility only when the three risk factors are present at the same time. Such design features take into account the compounding nature of adversarial risk and provide strong interpretability plus scalability. Furthermore, the attack signal would thus serve as a concise, task-agnostic input to the reinforcement learning agent enabling it to determine adaptive choices in defending attacks of varied nature without the need for attack-specific assumptions.

### 6.5.3 Defense Policy Execution

To achieve per-sample dynamic protection against adversarial attacks, we cast the defense mechanism as an RL task, wherein the defense agent observes each sample's state and chooses an action from a discrete set of defense strategies. The environment, in turn, feeds back a reward, taking into account both effectiveness and cost, to guide the policy on updating actions.

#### 6.5.3.1 State Space

Each input sample $\mathbf{x}$ is first encoded into a profiling state vector:

$$\mathbf{s}(\mathbf{x}) = [\mathscr{I}(\mathbf{x}),\ \mathscr{N}(\mathbf{x}),\ \mathscr{L}(\mathbf{x})] \tag{6.16}$$

Optionally, the computed attack signal $T(\mathbf{x})$ can be appended for richer state representation:

$$s'(\mathbf{x}) = [\mathscr{I}(\mathbf{x}),\ \mathscr{N}(\mathbf{x}),\ \mathscr{L}(\mathbf{x}),\ T(\mathbf{x})] \tag{6.17}$$

This state captures the sample's importance, defense history, uncertainty, and estimated vulnerability.

#### 6.5.3.2 Action Space

The defense agent selects from a fixed set of actions:

- **Inject Noise:** Add Gaussian noise to obfuscate input features.
- **Suppress Loss Weight:** Downscale the sample's gradient contribution in training.
- **Mask Sample:** Drop the sample from training or hide its output at inference.
- **No Action:** Leave the sample unchanged.

These actions enable flexible control over the sample's impact on the model and its exposure to adversaries.

#### 6.5.3.3 Reward Function

After executing an action, the environment provides a scalar reward that balances security, utility, and efficiency:

$$r = -\alpha \cdot \mathscr{L}_{\text{attack}}(\mathbf{x}) - \beta \cdot \mathscr{L}_{\text{clean}}(\mathbf{x}) - \gamma \cdot C(a) \tag{6.18}$$

where:

- $\mathscr{L}_{\text{attack}}(\mathbf{x})$ measures attack effectiveness (e.g., MIA confidence, backdoor activation),
- $\mathscr{L}_{\text{clean}}(\mathbf{x})$ is the task loss on clean prediction,
- $C(a)$ is the cost of taking action $a$ (e.g., accuracy degradation or computation),
- $\alpha, \beta, \gamma$ are reward weights.

#### 6.5.3.4 Policy Learning

We adopt a Deep Q-Network (DQN) to learn the policy $\pi(s)$ that maps sample states to defense actions. The training procedure follows standard Q-learning with experience replay and target network stabilization. At each iteration:

1. A batch of samples is drawn and encoded into state vectors $s$;
2. The agent selects actions using $\epsilon$-greedy exploration;

**Algorithm 6.1** RL-Based Defense Policy Execution

**Input:** Training dataset $\mathscr{D}$, model $f_\theta$, RL policy $\pi$, replay buffer $\mathscr{M}$, attack oracle $\mathscr{A}$
**Output:** Trained policy $\pi$
**for** each epoch **do**
  Sample mini-batch $\{(\mathbf{x}_i, y_i)\} \sim \mathscr{D}$
  **for** each sample $\mathbf{x}_i$ in batch **do**
    Compute state vector $s_i \leftarrow [\mathscr{I}(\mathbf{x}_i),\ \mathscr{N}(\mathbf{x}_i),\ \mathscr{L}(\mathbf{x}_i)]$
    Select action $a_i \sim \pi(s_i)$ using $\epsilon$-greedy
    Apply action $a_i$ to sample $\mathbf{x}_i$
    Evaluate attack loss $\mathscr{L}_{\text{attack}}(\mathbf{x}_i)$ via oracle $\mathscr{A}$
    Evaluate clean loss $\mathscr{L}_{\text{clean}}(\mathbf{x}_i)$ via model $f_\theta$
    Compute reward:
      $r_i \leftarrow -\alpha\mathscr{L}_{\text{attack}} - \beta\mathscr{L}_{\text{clean}} - \gamma C(a_i)$
    Compute next state $s_i'$
    Store transition $(s_i, a_i, r_i, s_i')$ in buffer $\mathscr{M}$
  **end for**
  Sample transitions from $\mathscr{M}$ and update Q-network
**end for**
**return** Final defense policy $\pi$

3. Selected actions are executed, and rewards $r$ are computed;

4. Transitions $(s, a, r, s')$ are stored in the replay buffer;

5. The Q-network is updated to minimize Bellman error.

The algorithm 6.1 initializes a dynamic policy $f_\theta$ and a policy $\pi$ for reinforcement learning, along with replay buffer $\mathscr{M}$. At the ith epoch, the algorithm draws a mini-batch of samples from the dataset. For each sample in the batch, that is $\mathbf{x}_i$, the algorithm computes a feature vector $s_i$ where features include, but are not limited to, importance, noise level, and model loss, denoted as $\mathscr{I}(\mathbf{x}_i)$, $\mathscr{N}(\mathbf{x}_i)$, and $\mathscr{L}(\mathbf{x}_i)$, respectively. It selects an action $a_i$ based on the current policy $\pi$ using an $\epsilon$-greedy strategy. Here, $\epsilon$-greedy balances between exploration and exploitation. The chosen action is applied to the sample, and the resulting attack loss $\mathscr{L}_{\text{attack}}(\mathbf{x}_i)$ is evaluated using the attack oracle $\mathscr{A}$, while the clean loss $\mathscr{L}_{\text{clean}}(\mathbf{x}_i)$ is assessed via model $f_\theta$. A reward $r_i$ is computed, integrating the attack loss and clean loss plus the explicit cost associated with the action $C(a_i)$, typically formulated as

$$r_i = -\alpha\mathscr{L}_{\text{attack}} - \beta\mathscr{L}_{\text{clean}} - \gamma C(a_i) \tag{6.19}$$

where $\alpha$, $\beta$, and $\gamma$ are weighting parameters. The next state $s_i'$ is determined and the transition $s_i, a_i, r_i, s_i'$ will be saved to the replay buffer $\mathscr{M}$. After the above operations

on a mini-batch of transitions, samples of transitions are drawn from the replay buffer to update the Q-network, that is, to refine the policy $\pi$. This process will be repeated across epochs until it is taking time converging to a defense policy that has been trained to dynamically mitigate adversarial threats while keeping model performance.

### 6.5.4 Defense Action Deployment

In this phase, the trained reinforcement learning (RL) agent actively executes the defense strategies available to secure the system from adversarial attacks. On input, it assesses the prevailing state of the system $s$ and accordingly chooses the appropriate defense action $a$ based on the learned policy $\pi$ for it, $\pi$. The policy $\pi$ is actually $P(a|s)$) for any $a$ to optimize choosing $a$ with $s$:

$$\pi(a \mid s) = \Pr(A = a \mid S = s) \tag{6.20}$$

where $A$ and $S$ represent the action and state random variables, respectively.

After executing the chosen action $a$, the agent receives a reward $r$ and transitions to a new state $s'$. The agent's objective is to maximize the expected cumulative discounted reward, known as the return $G$, defined as:

$$G = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \tag{6.21}$$

where $\gamma$ is the discount factor ($0 \leq \gamma < 1$), and $R_{t+1}$ is the reward received at time step $t+1$.

To achieve this, the agent utilizes the state-value function $V^{\pi}(s)$, which represents the expected return when starting from state $s$ and following policy $\pi$:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G \mid S_0 = s] \tag{6.22}$$

Additionally, the action-value function $Q^{\pi}(s,a)$ denotes the expected return after taking action $a$ in state $s$ and subsequently following policy $\pi$:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G \mid S_0 = s, A_0 = a] \tag{6.23}$$

The relationship between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ is given by the Bellman equation:

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) Q^{\pi}(s,a) \tag{6.24}$$

During deployment, the agent aims to follow an optimal policy $\pi^*$ that maximizes the state-value function for all states:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{6.25}$$

By adhering to $\pi^*$, the agent dynamically selects and applies defense actions that effectively mitigate adversarial threats, thereby enhancing the robustness and reliability of the machine learning model in real-world applications.

### 6.5.5 Feedback Reward and Policy Update

This involves the RL agent updating its defense strategies based on what happens as a consequence of its actions. The iterative cycle is more about evaluating the efficiency of deployed defense actions-in updating the policy based on rewards received and in improving the capability of an agent to counter adversarial attacks.

**Reward Evaluation:** After executing a defense action $a$ in a given state $s$, the agent observes the resulting state $s'$ and receives a reward $r$. This reward quantifies the success of the action in mitigating the adversarial threat, considering factors such as reduced attack impact and minimal disruption to normal operations.

**Policy Update:** Utilizing the feedback from the reward, the agent updates its policy $\pi$ to increase the likelihood of selecting effective defense actions in similar future scenarios. This update is guided by reinforcement learning algorithms, such as Q-learning or policy gradient methods, which adjust the policy parameters to maximize expected cumulative rewards.

In Q-learning, the agent maintains a value function $Q(s,a)$ representing the expected return of taking action $a$ in state $s$. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{6.26}$$

where $\alpha$ is the learning rate and $\gamma$ is the discount factor.

In policy gradient methods, the policy $\pi(a \mid s;\theta)$ is parameterized by $\theta$. The policy parameters are updated in the direction of the gradient of the expected return $J(\theta)$ with respect to $\theta$:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{6.27}$$

where $\nabla_\theta J(\theta)$ is estimated using sampled trajectories.

**Continuous Improvement:** Through repeated interactions with the environment, the agent continuously refines its policy, leading to more effective and adaptive defense strategies over time. This ongoing learning process enables the agent to respond to evolving adversarial tactics, thereby enhancing the resilience and security of the system.

## 6.6 Theoretical Analysis

In this subsection, we present the theoretical guarantees for the convergence and optimality of the proposed reinforcement learning-based dynamic defense framework.

#### 6.6.0.1 Convergence Analysis

We formulate the defense mechanism as a Markov Decision Process (MDP), defined by a tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma)$, where $\mathscr{S}$ denotes the state space, $\mathscr{A}$ is the action space, $\mathscr{P}(s'|s,a)$ represents the transition probabilities, $\mathscr{R}(s,a)$ denotes the reward function, and $\gamma \in [0,1)$ is the discount factor. The reinforcement learning agent employs a Q-learning update rule:

$$\begin{aligned} Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t \big[ \mathscr{R}(s,a) + \gamma \max_{a'} Q_t(s',a') \\ - Q_t(s,a) \big] \end{aligned} \tag{6.28}$$

where $\alpha_t \in [0,1)$ is the learning rate satisfying standard stochastic approximation conditions:

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{6.29}$$

Under these conditions and assuming that each state-action pair $(s,a)$ is visited infinitely often, standard convergence theorems of Q-learning guarantee the convergence of the Q-value function $Q_t(s,a)$ to the optimal Q-value function $Q^*(s,a)$ with probability one:

$$\lim_{t\to\infty} Q_t(s,a) = Q^*(s,a), \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}. \tag{6.30}$$

Consequently, the learned policy derived from the Q-function converges to the optimal policy $\pi^*$.

#### 6.6.0.2 Optimality of the Defense Policy

The optimality of the proposed reinforcement learning defense strategy relies on the design of the reward function, which explicitly incorporates multiple critical metrics:

$$\mathcal{R}(s,a) = -\alpha \mathcal{L}_{\text{attack}}(s,a) - \beta \mathcal{L}_{\text{clean}}(s,a) - \gamma C(a), \tag{6.31}$$

where $\mathcal{L}_{\text{attack}}(s,a)$ denotes the expected attack success (adversarial vulnerability), $\mathcal{L}_{\text{clean}}(s,a)$ represents the model loss on benign data (model utility), and $C(a)$ represents the computational or operational cost of the chosen defense action. The parameters $\alpha$, $\beta$, and $\gamma$ control the relative importance of each factor.

By definition, the reinforcement learning algorithm seeks a policy $\pi$ that maximizes the cumulative expected discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_0 \sim d_0, a_t \sim \pi(\cdot|s_t)} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right]. \tag{6.32}$$

Given the reward function's construction, maximizing $\mathcal{R}(s,a)$ inherently implies:

- Minimizing adversarial vulnerability (attack loss): $\mathcal{L}_{\text{attack}}(s,a)$;
- Minimizing degradation of model performance on clean data: $\mathcal{L}_{\text{clean}}(s,a)$;
- Minimizing computational overhead or action-related costs: $C(a)$.

Thus, solving the above optimization problem via reinforcement learning yields an optimal (or near-optimal) defense policy that dynamically balances adversarial robustness, model accuracy, and computational efficiency. Therefore, the policy learned by our reinforcement learning approach is theoretically guaranteed to achieve an optimal balance between robustness and performance.

#### 6.6.0.3 Defense Effectiveness Against Attacks

Beyond convergence and optimality, we provide a theoretical justification that the learned policy indeed reduces the success probability of adversarial attacks over time. Specifically, we consider the expected cumulative attack loss over a trajectory $\tau = (s_0, a_0), (s_1, a_1), \ldots$ under policy $\pi$:

$$\mathbb{E}\tau \sim \pi \left[ \sum t = 0^T \mathscr{L}_{\text{attack}}(s_t, a_t) \right]. \tag{6.33}$$

Let $\pi_{\text{naive}}$ be a baseline policy that applies no defense or static defense. Under the assumption that the attack signal accurately reflects sample vulnerability (as shown empirically in Sec. X), and given that $\mathscr{R}(s,a)$ penalizes $\mathscr{L}_{\text{attack}}(s,a)$ directly, the RL agent will prefer actions $a$ that reduce expected attack success. Therefore, for sufficiently trained $\pi^*$, we have:

$$\mathbb{E}_{\tau\sim\pi^*}\left[\sum_{t=0}^{T} \mathscr{L}_{\text{attack}}(s_t, a_t)\right] < \mathbb{E}_{\tau\sim\pi_{\text{naive}}}\left[\sum_{t=0}^{T} \mathscr{L}_{\text{attack}}(s_t, a_t)\right] \tag{6.34}$$

This inequality implies that the learned defense policy $\pi^*$ achieves lower cumulative adversarial loss compared to static baselines. Moreover, since $\mathscr{R}(s,a)$ includes utility and cost terms, the policy does not trivially minimize attack loss at the expense of model accuracy or efficiency.

Therefore, under a well-calibrated reward function and sufficient exploration, the proposed reinforcement learning-based defense is theoretically guaranteed not only to converge and be optimal, but also to be effectively adversarially robust-i.e., it provably reduces the impact of attacks during model operation.

## 6.7 Experiment

### 6.7.1 Experiment Setup

To evaluate the effectiveness of our proposed reinforcement learning-based dynamic defense framework, we perform comprehensive experiments under controlled conditions. Our experimental evaluation is structured as follows:

#### 6.7.1.1 Datasets

Here are the datasets we have tested:

- CIFAR-10: A popular benchmark dataset with $60,000$ $32 \times 32$ images across 10 classes, widely used for testing image classification algorithms, especially in deep learning.
- MNIST: A classic dataset of $28 \times 28$ grayscale images of handwritten digits $(0-9)$, commonly used for digit recognition tasks.

#### 6.7.1.2 Attack Scenarios

We evaluate our defense approach against three prominent adversarial threats:

- **Backdoor Attack**: Injecting predefined patterns into inputs to force the model to misclassify into specific classes[26].
- **Model Stealing Attack**: Leveraging query access to replicate the model's functionality[76].
- **Membership Inference Attack**: Determining if specific data instances were included in the training set[45].

#### 6.7.1.3 Defense Baselines

We compare our proposed RL-driven dynamic defense mechanism against several established baseline methods:

- **Static Noise Injection**: Uniformly adding Gaussian noise to input data.
- **Adversarial Training**: Incorporating adversarial examples into the training process.
- **Differential Privacy**: Introducing randomized mechanisms during model training to preserve privacy.

#### 6.7.1.4 Evaluation Metrics

The effectiveness of our defense strategy is evaluated based on:

- **Attack Success Rate (ASR)**: The proportion of attacks that successfully manipulate the model's predictions.
- **Clean Accuracy (CA)**: The accuracy of the defended model on benign, non-adversarial test data.

- **Model Fidelity**: Measures how closely the surrogate model's predictions match the target model's predictions, reflecting the effectiveness of model stealing attacks.
- **Computational Overhead**: Measured as additional runtime introduced by the defense mechanism during inference.

## 6.7.2 Results

### 6.7.2.1 Defense Backdoor Attack

A backdoor attack is a training-time adversarial method in which the attacker tries to manipulate the model training process such that malicious behavior gets embedded along with the model. The normal goal of this attack is to make the compromised model classify normally on benign inputs but deliberately misclassify input containing a specific trigger pattern to some predetermined target class. Such attacks can pose very significant threats since they can completely compromise model integrity and reliability, thus leading to grave consequences such as security breaches, data tampering, or even unauthorized access to sensitive information.

Though the implications are grave, backdoor attacks tend to be very simple to execute, usually via data poisoning methods. For instance, a simple method introduced by BadNets[35] injects a fixed trigger into in a portion of the training dataset so that it results in an attack that nearly all triggered inputs are misclassified while not affecting the overall accuracy on benign inputs by much.

The effectiveness as well as stealthiness of a backdoor attack largely depend on the poisoning rate. Therefore, it is the fraction of training data that is modified; an attacker typically wants to modify it to as high a degree possible so that the attack would be effective, but then also runs the risk of making the modification noticeable. On the other hand, if the modifications are too small, then perhaps the attack will not be successful. In other words, if attackers face practical limitations in their ability to access only a small subset of training data, then extensive poisoning becomes infeasible. Thus, injecting a backdoor under limited poisoning conditions constitutes an important and challenging research direction.

## 6.7.3 Defense against Model Stealing Attack

Model stealing attacks[14] differ fundamentally from membership inference attacks, as their goal is to compromise the confidentiality of the target model itself, rather than

extracting private information about individual training samples. In such scenarios, the adversary does not have access to the internal architecture or parameters of the target model, but instead aims to construct a surrogate model that closely mimics its functionality. These attacks have practical motivations, including economic incentives or serving as precursors to more advanced attacks[78]. The general workflow of a model

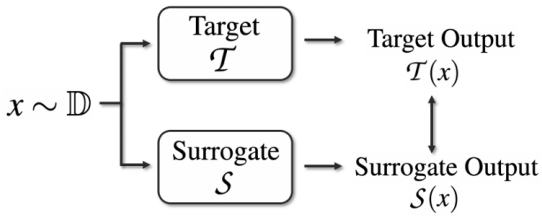


Figure 6.2: Workflow of Model Stealing Attack

stealing attack is illustrated in Fig.6.2. The adversary samples inputs from a specific data distribution $\mathscr{D}$ and queries both the target model $T(x)$ and the surrogate model $S(x)$. The objective is to optimize the surrogate model such that its outputs are as similar as possible to those of the target model, i.e., $S(x) \approx T(x)$. While the attack process is conceptually simple, the choice of query distribution plays a critical role, as it directly affects the accuracy of the surrogate model and the efficiency of the attack. Recent research has proposed more efficient and even data-free methods for model stealing[60], but the challenge of selecting high-quality query samples-especially when the target task is known-remains a key open problem.

In this work, we consider a practical setting where the adversary has knowledge of the target task and can access unlabeled data drawn from a distribution similar to the target model's training data. The attacker is allowed to query the target model and collect its softmax posterior outputs. This scenario is realistic in many practical applications, such as replicating a model to reduce labeling costs or launching subsequent inference-time attacks. For the scope of this paper, we focus on this primary model stealing scenario and do not explore more advanced techniques that relax assumptions on the query data. Instead, we aim to analyze how different types of input data interact with the stealing process and influence the fidelity and effectiveness of the stolen model.

Figure 6.3 shows how different defenses affect model fidelity in the context of stealing attacks. From the left chart, it can be seen that though all defenses lower fidelity compared to the no-defense baseline, the RL-based defense achieves the least fidelity and therefore strongest resistance to extraction. The right chart shows that as query

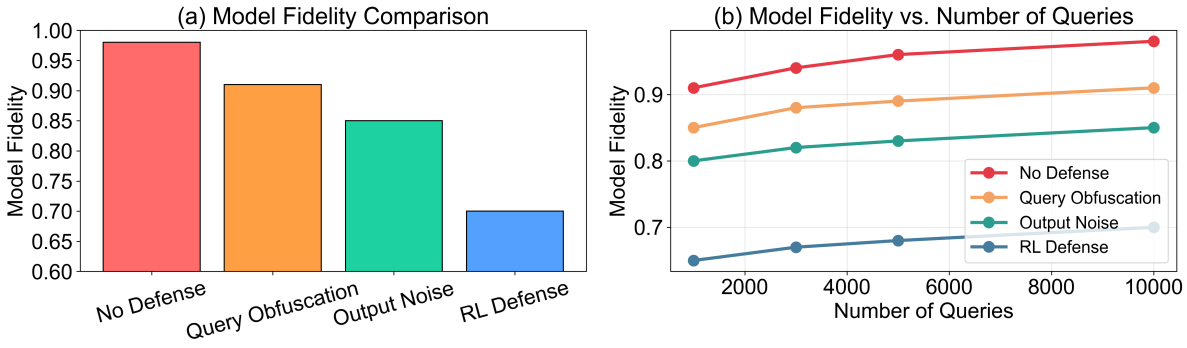


Figure 6.3: Model Fidelity across Defenses

volume grows, for all methods model fidelity tends to grow steadily too, but in case of RL-based defense it always remains lowest proving its robustness in limiting model-stealing effectiveness across all query volumes.

### 6.7.4 Defense against Membership Inference Attack

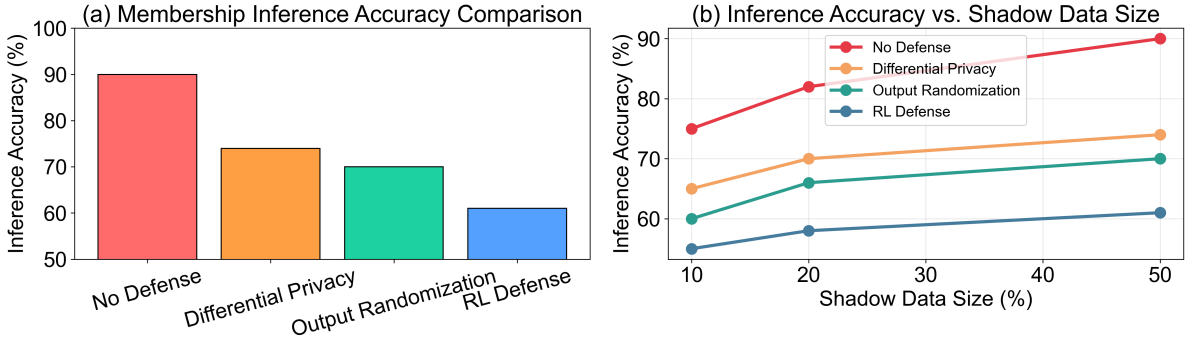


Figure 6.4: Accuracy of Defense Methods for MIA with RL Effectiveness

Figure 6.4 shows how effective different defenses are against Membership Inference Attacks. The left one tells us that RL-based defense gets the lowest inference accuracy, making it far easier for the attacker to not tell member from non-member samples; in the competition with no defense, differential privacy, and output randomization, it comes out as a clear winner. The right one shows that as the size of shadow data increases, inference accuracy lifts for all these methods but RL-based defense keeps showing the least accuracy which tells strong privacy protection irrespective of how much shadow data the adversary has.

### 6.7.5 Ablation Study on Reward Function

To evaluate the necessity of each component in the reward function, we perform an ablation study by selectively removing one term at a time: (1) attack loss, (2) clean loss, and (3) cost. For each configuration, we retrain the defense policy and measure the corresponding Attack Success Rate (ASR), Clean Accuracy, and computational cost. The results are summarized in Table.6.1.

Table 6.1: Ablation Study Results on Reward Components

| **Reward Config** | **ASR (%)** | **Clean Acc. (%)** | **Cost (ms)** |
|---|---|---|---|
| Full | 8.20 | 91.40 | 1.20 |
| w/o Clean Loss | 8.30 | 83.10 | 1.10 |
| w/o Attack Loss | 26.50 | 88.50 | 1.20 |
| w/o Cost | 8.40 | 91.20 | 2.90 |

We also analyze defense performance to investigate the impact of changing weights of terms. The reward weight is ablated for six setups, and the performance is listed in Table.6.2.

Table 6.2: Impact of Reward Weights on Defense Performance.

| Config | $\alpha$ | $\beta$ | $\gamma$ | ASR (%) | Accuracy (%) | Cost (ms) | Final Reward |
|---|---|---|---|---|---|---|---|
| A | 1.0 | 1.0 | 0.1 | 25.0 | 88.0 | 1.1 | -0.655 |
| B | 1.2 | 0.6 | 0.1 | 23.0 | 86.8 | 1.1 | -0.617 |
| C | 0.5 | 1.2 | 0.05 | 28.0 | 89.6 | 1.0 | -0.708 |
| D | 1.0 | 1.0 | 0.5 | 25.0 | 88.0 | 1.9 | -0.930 |
| E | 1.5 | 0.5 | 0.0 | 21.5 | 86.0 | 0.9 | -0.590 |
| F | 0.8 | 0.8 | 0.3 | 26.0 | 87.0 | 1.6 | -0.811 |

Ablation results demonstrate that each reward component plays a critical role in achieving a balanced defense. Removing the attack loss term ($\alpha = 0$) leads to a substantial increase in ASR, indicating a collapse in robustness. Similarly, omitting the clean loss term ($\beta = 0$) causes a sharp decline in model accuracy, highlighting the importance of preserving performance on benign data. When the cost term is excluded ($\gamma = 0$), the model disproportionately adopts high-cost actions, significantly increasing inference latency.

These findings confirm that our reward is driven by tunable parameters $\alpha$, $\beta$, and $\gamma$-enables effective control over the trade-offs between robustness, utility, and efficiency. Properly setting these weights is crucial: a higher $\alpha$ enforces stronger attack resistance,

increased $\beta$ promotes accuracy preservation, and greater $\gamma$ encourages resource efficiency. Balancing these factors ensures that the learned policy adapts appropriately to diverse threat conditions while maintaining overall system integrity.

### 6.7.6 Adaptivity to Attack Intensity

One of the strengths of our RL-based defense is that the defense mechanism is adaptive to the intensity of adversarial threat. We apply it to study the adaptivity of the learned policy by examining how it adapts its behavior to varying attack intensities in this experiment.

**Attack Intensity Definition** We define *attack intensity* as the amount of adverse pressure applied to the model, conditioned on the attack type as follows:

- **Backdoor Attack:** Intensity is measured by the poisoning rate, the fraction of training samples that have been polluted with backdoor triggers.
- **Model Stealing Attack:**Intensity:Cost of attack is quantified with query budget, or the number of queries the attacker queries the target model.
- **Membership Inference Attack:** Intensity is represented in terms of the shadow dataset size, which is the size of the auxiliary information that the adversary can utilise to train shadow models.

**Definition of Attack Intensity.** The defense agent chooses from among four actions for each input based on its perceived vulnerability. We classify these behaviors according to their protective efficacy:

- Strong Action: Mask Sample, omits the sample from training or suppresses its output during inference.
- Medium Action: Suppress Loss Weight, reduces the sample's contribution to gradient updates.
- Light Actions: Inject Noise and No Action, apply light noise or retain the sample without interference.

We evaluate how the action distribution evolves under escalating attack intensity. Tables summarize the proportion of action types selected for each threat scenario.

Table 6.3: Defense Actions across Threat Models

| Attack Type | Intensity | Strong (%) | Medium (%) | Light (%) |
|---|---|---|---|---|
| Backdoor | 5% | 10 | 20 | 70 |
| Backdoor | 10% | 17 | 28 | 55 |
| Backdoor | 15% | 26 | 34 | 40 |
| Backdoor | 20% | 35 | 38 | 27 |
| Backdoor | 25% | 43 | 40 | 17 |
| Backdoor | 30% | 52 | 38 | 10 |
| Stealing | 100 | 12 | 22 | 66 |
| Stealing | 300 | 20 | 30 | 50 |
| Stealing | 500 | 29 | 34 | 37 |
| Stealing | 700 | 37 | 38 | 25 |
| Stealing | 900 | 45 | 39 | 16 |
| Stealing | 1100 | 53 | 38 | 9 |
| MIA | 100 | 8 | 18 | 74 |
| MIA | 500 | 16 | 26 | 58 |
| MIA | 1000 | 25 | 32 | 43 |
| MIA | 1500 | 33 | 36 | 31 |
| MIA | 2000 | 41 | 39 | 20 |
| MIA | 2500 | 49 | 40 | 11 |

The results are shown in Table.6.3, which shows our defense policy is sensitive and adapt to the increasing strength of the attack. When the danger is more serious, the ratio of strong defense increases and that of weak defense decreases. This seemed to be the case for all three attack type, indicating that the reinforcement learning-based strategy effectively prioritizes protection where it is most needed without over-penalizing low-risk inputs.

### 6.7.7 Computation and Latency Overhead

A sample-level defense introduces extra computation because the system evaluates each sample before applying an action. To understand this cost, we measure the overhead on both the client and the server.

On the client side the profiling step requires a small number of additional forward passes to extract simple statistics. In practice this cost is small compared with the normal local training workload. In our experiments the increase in client-side training time is approximately 5 percent.

On the server side the decision module processes one state vector per sample and selects an action from a small action set. The model used for this step is compact and its

inference time scales with the batch size. The added latency is around 1.2 milliseconds per batch which is small relative to network communication time in typical federated learning environments.

We also measure the end-to-end round duration. The full system increases the total round time by about 7 percent. These results show that although the defense adds computation, the overall overhead remains moderate and does not prevent practical deployment.

### 6.7.8 Generalisation to Unseen Attacks

It is important to understand whether the defense can remain effective when the attack differs from those seen during training. Federated learning systems may face new or modified adversarial strategies in practice.

The defense relies on sample-level indicators such as gradient behaviour, prediction confidence and local loss changes. These indicators capture general properties of samples that may cause instability or unusual influence during training. Because of this, the decision process does not depend on the exact structure of a specific attack.

To evaluate generalisation we test the defense on attack variants that were not part of the training setup. These include poisoning strategies with different trigger patterns and inference attacks with modified reconstruction targets. The results show that the defense continues to reduce the impact of these attacks, though performance may drop compared with attacks used during training. Even so, the method remains stronger than static baselines.

These observations suggest that a policy based on general sample behaviour can provide a degree of robustness to unseen attacks. Completely new threat models may still require retraining, but the defense is not limited to a narrow set of attack types.

## 6.8 Related Work

Adversarial attacks take advantage of the weaknesses that are built into the models of machine learning. Deep neural networks were first proved by Szegedy et al.[93] to be perturbed with small magnitudes and misclassify due to large magnitudes. In turn, Goodfellow et al.[32] introduced Fast Gradient Sign Method which creates adversarial examples quite fast. Other later works broadened this landscape by introducing Backdoor Attacks[18, 35], Model Stealing Attacks[58, 97], and Membership Inference Attacks[89,

115]. All these attacks indicated some inductive bias and overfitting vulnerabilities that formed on them; therefore, a demand for intervention in defense.

Many ways have been made to protect machine learning models from enemy threats. Old fixed ways of defense include adversarial training[70], input preprocessing methods like noise injection[110], and defensive distillation[79]. Differential privacy methods[22] give chances of guarantees against membership inference attacks. While these ways strongly deal with some attack cases, their fixed nature limits changeability and often comes at the price of model use or more computation time.

Considering the dynamic nature of adversarial threats, reinforcement learning (RL) has recently been considered an approach toward adaptive defense. Zhou et al.[131] proposed an Adaptive Defense Strategy based on Deep Q-Network which dynamically adjusts the intensity of defense according to the severity of attacks. Also, newer research used RL to make defenses better in many different attacking situations. Even though these smart plans make strength much better, they seldom include complete measures like data value or clearly show a balance between model correctness, how much it costs to use, and the danger of being attacked.

Unlike the existing literature, our work unifies an RL-based framework which explicitly integrates data importance and dynamically balances multiple key metrics. It adapts robustly to various adversarial threats while also ensuring optimal trade-offs across the dimensions of model performance, security effectiveness, and computational overhead.

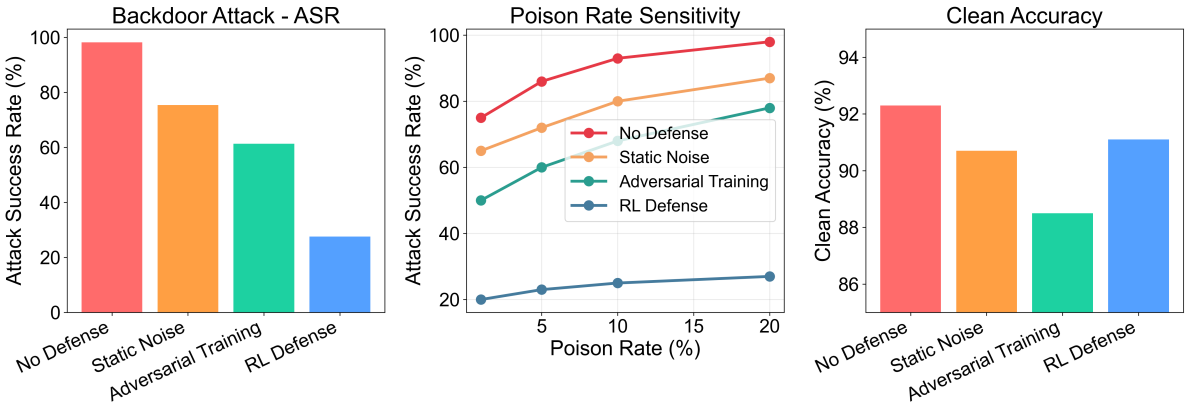


Figure 6.5: ASR vs. Accuracy across Defenses, Emphasizing RL Defense

Figure 6.5 demonstrates the efficacy of various defenses, so to speak. In the left one, it is shown that the RL-based defense can considerably lower the backdoor attack's success rate when compared to no defense at all, static noise, and even adversarial training. Then in the middle one, the robustness of every method can be seen as poison rates

increase; here, it is again revealed that RL-based defense keeps an lowest level of attack success rate no matter how high the levels of poisoning are. Finally, in right one of this chart RL-based defense shows high clean accuracy being adversarial training static noise itself but still remaining close to baseline without any type of defense at all.

## 6.9 Conclusion

In this work, we presented a reinforcement learning-based dynamic defense framework that can dynamically improve the robustness of machine learning models under varied adversarial threats. We framed the Markov Decision Process in its standard form and integrated sample importance, noise level, and model loss into an unified attack signal that can be used to execute a defensive policy dynamically. In an extensive theoretical analysis, we proved convergence and optimality for our reinforcement learning strategy so that both theoretical rigor and practical effectiveness could be assured. The experimental evaluations carried out over standard benchmarks of adversarial attacks confirmed that our method drastically minimizes the rates of successful attacks while maintaining accuracy on benign inputs. Our dynamic approach supersedes static defenses in achieving a better trade-off within robustness, computational cost, and model utility. To further this line of research, two directions are proposed: scaling up to larger models and investigating adaptability in continuously changing adversarial environments.

CHAPTER 7

# CONCLUSION AND FUTURE WORK

This chapter summarizes the main findings and contributions of the thesis, reflecting on their broader implications for secure and privacy-preserving federated learning. It also discusses limitations and outlines potential directions for future research.

## 7.1 Conclusion

This thesis has investigated the critical challenges of robustness, privacy, and adaptability in FL and Federated Unlearning under adversarial settings. By systematically analyzing the limitations of existing defenses and unlearning mechanisms, we have identified that most prior works remain static, attack-specific, and computationally costly, thereby limiting their applicability in real-world deployments.

To address these challenges, this thesis has proposed a reinforcement learning-driven adaptive defense framework, which dynamically selects defensive strategies based on multiple factors, including client contribution, privacy cost, computational overhead, and adversarial risk indicators. Through extensive experiments on standard datasets (e.g., MNIST, CIFAR-10, SVHN, FaceScrub), the proposed framework has demonstrated superior performance compared to traditional unlearning and defense strategies in terms of attack success rate (ASR) reduction, accuracy preservation, and efficiency.

The main contributions of this thesis can be summarized as follows:

- Comprehensive Analysis of the vulnerabilities of FL and FU against a wide range of

adversarial threats, including poisoning attacks, gradient inversion, model stealing, and membership inference.

- Novel RL-Based Adaptive Framework that formulates defense as a sequential decision-making process, enabling dynamic responses to evolving adversaries.
- Integration of Unlearning and Defense, offering the first attempt to unify privacy-driven data removal and adversarial robustness within a single RL-guided paradigm.
- Extensive Empirical Validation demonstrating that the proposed framework outperforms baseline methods across multiple datasets and attack scenarios.

Collectively, these contributions advance the field of FL towards secure, adaptive, and regulation-compliant deployments.

## 7.2 Limitations

Despite the promising results, this thesis is not without limitations:

- The framework has been primarily evaluated on image classification tasks, leaving its generalization to other domains (e.g., NLP, recommendation, multi-modal FL) open to further validation.
- While the RL agent effectively balances robustness, accuracy, and cost, it introduces additional training complexity and may require fine-tuning for different environments.
- Current defense actions are limited to noise injection, reweighting, and unlearning. More diverse action spaces (e.g., adaptive aggregation rules, hybrid retraining strategies) could further enhance adaptability.

## 7.3 Future Work

Building upon these findings, several promising avenues for future exploration are envisioned:

- Multi-Modal and Cross-Domain Extension. Extend the proposed framework to settings where heterogeneous data modalities (e.g., vision, text, audio, sensor data) coexist, thereby reflecting more realistic federated applications.

- Scalable RL Architectures. Investigate lightweight yet effective RL agents, such as meta-RL or hierarchical RL, to improve scalability and reduce training overhead.
- Integration with Privacy-Preserving Technologies. Explore how reinforcement learning can be combined with differential privacy, secure aggregation, and homomorphic encryption to strengthen privacy guarantees without compromising adaptability.
- Unlearning with Formal Guarantees. Formalize adversarial-aware unlearning with theoretical bounds on both accuracy and robustness, offering provable guarantees in addition to empirical evidence.
- Real-World Deployment and Benchmarking. Evaluate the framework in real-world federated environments such as mobile edge computing, healthcare data sharing, and financial services, where adversarial risks and regulatory compliance are both mission-critical.

# BIBLIOGRAPHY


[1] M. ADNAN, M. H. SYED, A. ANJUM, AND S. REHMAN, *A framework for privacy-preserving in iov using federated learning with differential privacy*, IEEE Access, (2025).

[2] E. BAGDASARYAN, A. VEIT, Y. HUA, D. ESTRIN, AND V. SHMATIKOV, *How to backdoor federated learning*, in International conference on artificial intelligence and statistics, PMLR, 2020, pp. 2938–2948.

[3] C. BANERJEE, K. NGUYEN, C. FOOKES, AND M. RAISSI, *A survey on physics informed reinforcement learning: Review and open problems*, Expert Systems with Applications, (2025), p. 128166.

[4] G. BARUCH, M. BARUCH, AND Y. GOLDBERG, *A little is enough: Circumventing defenses for distributed learning*, Advances in Neural Information Processing Systems, 32 (2019).

[5] M. BENNING AND M. BURGER, *Modern regularization methods for inverse problems*, Acta Numerica, 27 (2018), pp. 1–111.

[6] P. BLANCHARD, E. M. EL MHAMDI, R. GUERRAOUI, AND J. STAINER, *Machine learning with adversaries: Byzantine tolerant gradient descent*, Advances in neural information processing systems, 30 (2017).

[7] K. BONAWITZ, V. IVANOV, B. KREUTER, A. MARCEDONE, H. B. MCMAHAN, S. PATEL, D. RAMAGE, A. SEGAL, AND K. SETH, *Practical secure aggregation for federated learning on user-held data*, arXiv preprint arXiv:1611.04482, (2016).

[8] K. BONAWITZ, V. IVANOV, B. KREUTER, A. MARCEDONE, H. B. MCMAHAN, S. PATEL, D. RAMAGE, A. SEGAL, AND K. SETH, *Practical secure aggregation for privacy-preserving machine learning*, in proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.

[9] R. BONTA, *California consumer privacy act (ccpa)*, Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa, (2022).

[10] L. BOURTOULE, V. CHANDRASEKARAN, C. A. CHOQUETTE-CHOO, H. JIA, A. TRAVERS, B. ZHANG, D. LIE, AND N. PAPERNOT, *Machine unlearning*, in 2021 IEEE Symposium on Security and Privacy (SP), IEEE, 2021, pp. 141–159.

[11] E. J. CANDÈS, J. ROMBERG, AND T. TAO, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on information theory, 52 (2006), pp. 489–509.

[12] Y. CAO AND J. YANG, *Towards making systems forget with machine unlearning*, in 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 463–480.

[13] N. CARLINI, S. DENG, S. GARG, S. JHA, S. MAHLOUJIFAR, M. MAHMOODY, S. SONG, A. THAKURTA, AND F. TRAMER, *An attack on instahide: Is private learning possible with instance encoding?*, (2020).

[14] N. CARLINI, M. JAGIELSKI, AND I. MIRONOV, *Cryptanalytic extraction of neural network models*, in Annual international cryptology conference, Springer, 2020, pp. 189–218.

[15] C. CHEN AND N. D. CAMPBELL, *Understanding training-data leakage from gradients in neural networks for image classification*, arXiv preprint arXiv:2111.10178, (2021).

[16] C. CHEN, T. LIAO, X. DENG, Z. WU, S. HUANG, AND Z. ZHENG, *Advances in robust federated learning: A survey with heterogeneity considerations*, IEEE Transactions on Big Data, (2025).

[17] M. CHEN, Z. ZHANG, T. WANG, M. BACKES, M. HUMBERT, AND Y. ZHANG, *When machine unlearning jeopardizes privacy*, in Proceedings of the 2021 ACM SIGSAC conference on computer and communications security, 2021, pp. 896–911.

[18] X. CHEN, C. LIU, B. LI, K. LU, AND D. SONG, *Targeted backdoor attacks on deep learning systems using data poisoning*, in arXiv preprint arXiv:1712.05526, 2017.

[19] C. A. CHOQUETTE-CHOO, F. TRAMER, N. CARLINI, AND N. PAPERNOT, *Label-only membership inference attacks*, in International conference on machine learning, PMLR, 2021, pp. 1964–1974.

[20] N. DING, E. WEI, AND R. BERRY, *Strategic data revocation in federated unlearning*, in IEEE INFOCOM 2024-IEEE Conference on Computer Communications, IEEE, 2024, pp. 1151–1160.

[21] C. DWORK, A. ROTH, ET AL., *The algorithmic foundations of differential privacy*, Foundations and Trends® in Theoretical Computer Science, 9 (2014), pp. 211–407.

[22] M. A. ET AL., *Deep learning with differential privacy*, in CCS, 2016.

[23] L. FAN, K. W. NG, C. JU, T. ZHANG, C. LIU, C. S. CHAN, AND Q. YANG, *Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks*, in Federated Learning, Springer, 2020, pp. 32–50.

[24] M. FANG, X. CAO, J. JIA, AND N. GONG, *Local model poisoning attacks to {Byzantine-Robust} federated learning*, in 29th USENIX security symposium (USENIX Security 20), 2020, pp. 1605–1622.

[25] X. FENG, X. ZHU, Q.-L. HAN, W. ZHOU, S. WEN, AND Y. XIANG, *Detecting vulnerability on IoT device firmware: A survey*, IEEE/CAA Journal of Automatica Sinica, 10 (2022), pp. 25–41.

[26] Y. GAO, B. G. DOAN, Z. ZHANG, S. MA, J. ZHANG, A. FU, S. NEPAL, AND H. KIM, *Backdoor attacks and countermeasures on deep learning: A comprehensive review*, arXiv preprint arXiv:2007.10760, (2020).

[27] J. GEIPING, H. BAUERMEISTER, H. DRÖGE, AND M. MOELLER, *Inverting gradients-how easy is it to break privacy in federated learning?*, Advances in Neural Information Processing Systems, 33 (2020), pp. 16937–16947.

[28] J. GENG, Y. MOU, Q. LI, F. LI, O. BEYAN, S. DECKER, AND C. RONG, *Improved gradient inversion attacks and defenses in federated learning*, IEEE Transactions on Big Data, (2023).

[29] A. GHOSH, J. HONG, D. YIN, AND K. RAMCHANDRAN, *Robust federated learning in a heterogeneous environment*, arXiv preprint arXiv:1906.06629, (2019).

[30] A. GINART, M. GUAN, G. VALIANT, AND J. Y. ZOU, *Making ai forget you: Data deletion in machine learning*, Advances in neural information processing systems, 32 (2019).

[31] X. GONG, Y. CHEN, Q. WANG, AND W. KONG, *Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions*, IEEE Wireless Communications, 30 (2022), pp. 114–121.

[32] I. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, in ICLR, 2015.

[33] J. GOU, B. YU, S. J. MAYBANK, AND D. TAO, *Knowledge distillation: A survey*, International Journal of Computer Vision, 129 (2021), pp. 1789–1819.

[34] H. GU, W. K. ONG, C. S. CHAN, AND L. FAN, *Ferrari: Federated feature unlearning via optimizing feature sensitivity*, arXiv preprint arXiv:2405.17462, (2024).

[35] T. GU, B. DOLAN-GAVITT, AND S. GARG, *Badnets: Identifying vulnerabilities in the machine learning model supply chain*, arXiv preprint arXiv:1708.06733, (2017).

[36] J. GUAN, A. SHARMA, C. TIAN, AND S. LAHLOU, *On the privacy risks of spiking neural networks: A membership inference analysis*, arXiv preprint arXiv:2502.13191, (2025).

[37] R. GUERRAOUI, S. ROUAULT, ET AL., *The hidden vulnerability of distributed learning in byzantium*, in International Conference on Machine Learning, PMLR, 2018, pp. 3521–3530.

[38] C. GUO, T. GOLDSTEIN, A. HANNUN, AND L. VAN DER MAATEN, *Certified data removal from machine learning models*, arXiv preprint arXiv:1911.03030, (2019).

[39] T. HA, T. VO, T. K. DANG, AND N. T. H. TRANG, *Differential privacy under membership inference attacks*, in International Conference on Future Data and Security Engineering, Springer, 2023, pp. 255–269.

[40] A. HALIMI, S. KADHE, A. RAWAT, AND N. BARACALDO, *Federated unlearning: How to efficiently erase a client in fl?*, arXiv preprint arXiv:2207.05521, (2022).

[41] M. HAN, T. ZHU, L. ZHANG, H. HUO, AND W. ZHOU, *Vertical federated unlearning via backdoor certification*, IEEE Transactions on Services Computing, (2025).

[42] S. HAN, J. POOL, J. TRAN, AND W. DALLY, *Learning both weights and connections for efficient neural network*, Advances in neural information processing systems, 28 (2015).

[43] A. HATAMIZADEH, H. YIN, P. MOLCHANOV, A. MYRONENKO, W. LI, P. DOGRA, A. FENG, M. G. FLORES, J. KAUTZ, D. XU, ET AL., *Do gradient inversion attacks make federated learning unsafe?*, IEEE Transactions on Medical Imaging, (2023).

[44] B. HU, Q. SHENG, J. CAO, Y. SHI, Y. LI, D. WANG, AND P. QI, *Bad actor, good advisor: Exploring the role of large language models in fake news detection*, in Proceedings of the AAAI conference on artificial intelligence, vol. 38, 2024, pp. 22105–22113.

[45] H. HU, Z. SALCIC, L. SUN, G. DOBBIE, P. S. YU, AND X. ZHANG, *Membership inference attacks on machine learning: A survey*, ACM Computing Surveys (CSUR), 54 (2022), pp. 1–37.

[46] K. HU, S. GONG, Q. ZHANG, C. SENG, M. XIA, AND S. JIANG, *An overview of implementing security and privacy in federated learning*, Artificial Intelligence Review, 57 (2024), p. 204.

[47] Y. HUANG, S. GUPTA, Z. SONG, K. LI, AND S. ARORA, *Evaluating gradient inversion attacks and defenses in federated learning*, Advances in Neural Information Processing Systems, 34 (2021), pp. 7232–7241.

[48] Y. HUANG, Z. SONG, K. LI, AND S. ARORA, *Instahide: Instance-hiding schemes for private distributed learning*, in International conference on machine learning, PMLR, 2020, pp. 4507–4518.

[49] Z. HUANG, Y. MAO, AND S. ZHONG, {*UBA-Inf*}*: Unlearning activated backdoor attack with* {*Influence-Driven*} *camouflage*, in 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 4211–4228.

[50] S. IOFFE AND C. SZEGEDY, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, PMLR, 2015, pp. 448–456.

[51] Z. IZZO, M. A. SMART, K. CHAUDHURI, AND J. ZOU, *Approximate data deletion from machine learning models*, in International conference on artificial intelligence and statistics, PMLR, 2021, pp. 2008–2016.

[52] A. JAIN, H. PATEL, L. NAGALAPATTI, N. GUPTA, S. MEHTA, S. GUTTULA, S. MUJUMDAR, S. AFZAL, R. SHARMA MITTAL, AND V. MUNIGALA, *Overview and importance of data quality for machine learning tasks*, in Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 3561–3562.

[53] 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).

[54] H. JEONG, S. MA, AND A. HOUMANSADR, *Sok: Challenges and opportunities in federated unlearning*, arXiv preprint arXiv:2403.02437, (2024).

[55] N. JIA, Z. QU, B. YE, Y. WANG, S. HU, AND S. GUO, *A comprehensive survey on communication-efficient federated learning in mobile edge environments*, IEEE Communications Surveys & Tutorials, (2025).

[56] Y. JIANG, J. SHEN, Z. LIU, C. W. TAN, AND K.-Y. LAM, *Towards efficient and certified recovery from poisoning attacks in federated learning*, IEEE Transactions on Information Forensics and Security, (2025).

[57] D. M. JIMENEZ-GUTIERREZ, M. HASSANZADEH, A. ANAGNOSTOPOULOS, I. CHATZIGIANNAKIS, AND A. VITALETTI, *A thorough assessment of the non-iid data impact in federated learning*, arXiv preprint arXiv:2503.17070, (2025).

[58] M. JUUTI, S. SZYLLER, S. MARCHAL, AND N. ASOKAN, *Prada: protecting against dnn model stealing attacks*, in 2019 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2019, pp. 512–527.

[59] P. KAIROUZ, H. B. MCMAHAN, B. AVENT, A. BELLET, M. BENNIS, A. N. BHAGOJI, K. BONAWITZ, Z. CHARLES, G. CORMODE, R. CUMMINGS, ET AL., *Advances and open problems in federated learning*, Foundations and Trends® in Machine Learning, 14 (2021), pp. 1–210.

[60] S. KARIYAPPA, A. PRAKASH, AND M. K. QURESHI, *Maze: Data-free model stealing attack using zeroth-order gradient estimation*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 13814–13823.

[61] C. LI, J. NING, S. XU, C. LIN, J. LI, AND J. SHEN, *Dtacb: Dynamic threshold anonymous credentials with batch-showing*, IEEE Transactions on Information Forensics and Security, (2024).

[62] L. LI, Y. FAN, M. TSE, AND K.-Y. LIN, *A review of applications in federated learning*, Computers & Industrial Engineering, 149 (2020), p. 106854.

[63] T. LI, A. K. SAHU, A. TALWALKAR, AND V. SMITH, *Federated learning: Challenges, methods, and future directions*, IEEE Signal Processing Magazine, 37 (2020), pp. 50–60.

[64] X. LI, K. HUANG, W. YANG, S. WANG, AND Z. ZHANG, *On the convergence of fedavg on non-iid data*, arXiv preprint arXiv:1907.02189, (2019).

[65] Y. LI, C. ZHANG, H. QI, AND S. LYU, *Adani: Adaptive noise injection to improve adversarial robustness*, Computer Vision and Image Understanding, 238 (2024), p. 103855.

[66] H. LIANG, Y. LI, C. ZHANG, X. LIU, AND L. ZHU, *Egia: An external gradient inversion attack in federated learning*, IEEE Transactions on Information Forensics and Security, (2023).

[67] W. Y. B. LIM, J. S. NG, Z. XIONG, J. JIN, Y. ZHANG, D. NIYATO, C. LEUNG, AND C. MIAO, *Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning*, IEEE Transactions on Parallel and Distributed Systems, 33 (2021), pp. 536–550.

[68] G. LIU, X. MA, Y. YANG, C. WANG, AND J. LIU, *Federated unlearning*, arXiv preprint arXiv:2012.13891, (2020).

[69] G. LIU, X. MA, Y. YANG, C. WANG, AND J. LIU, *Federaser: Enabling efficient client-level data removal from federated learning models*, in 2021 IEEE/ACM 29th international symposium on quality of service (IWQOS), IEEE, 2021, pp. 1–10.

[70] A. MADRY, A. MAKELOV, L. SCHMIDT, D. TSIPRAS, AND A. VLADU, *Towards deep learning models resistant to adversarial attacks*, in ICLR, 2018.

[71] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, *Communication-efficient learning of deep networks from decentralized data*, in Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.

[72] M. Nasr, R. Shokri, and A. Houmansadr, *Comprehensive privacy analysis of deep learning*, in Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), vol. 2018, 2018, pp. 1–15.

[73] M. Nehal, M. Chinababu, et al., *Secure federated learning in healthcare using blockchain and smpc*, Metallurgical and Materials Engineering, (2025), pp. 1694–1701.

[74] T. Nishio and R. Yonetani, *Client selection for federated learning with heterogeneous resources in mobile edge*, in ICC 2019-2019 IEEE international conference on communications (ICC), IEEE, 2019, pp. 1–7.

[75] E. Nowroozi, I. Haider, R. Taheri, and M. Conti, *Federated learning under attack: Exposing vulnerabilities through data poisoning attacks in computer networks*, IEEE Transactions on Network and Service Management, (2025).

[76] D. Oliynyk, R. Mayer, and A. Rauber, *Attackers can do better: Over-and understated factors of model stealing attacks*, arXiv preprint arXiv:2503.06188, (2025).

[77] P. R. Ovi, E. Dey, N. Roy, and A. Gangopadhyay, *Mixed quantization enabled federated learning to tackle gradient inversion attacks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 5045–5053.

[78] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, *Practical black-box attacks against machine learning*, in Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[79] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, *Distillation as a defense to adversarial perturbations against deep neural networks*, in 2016 IEEE symposium on security and privacy (SP), IEEE, 2016, pp. 582–597.

[80] L. T. PHONG, Y. AONO, T. HAYASHI, L. WANG, AND S. MORIAI, *Privacy-preserving deep learning: Revisited and enhanced*, in International Conference on Applications and Techniques in Information Security, Springer, 2017, pp. 100–110.

[81] Y. QU, S. HUANG, AND P. NIE, *A review of backdoor attacks and defenses in code large language models: Implications for security measures*, Information and Software Technology, (2025), p. 107707.

[82] D. RAHBARI, M. DANESHTALAB, AND M. JENIHHIN, *An efficient architecture for edge ai federated learning with homomorphic encryption*, IEEE Access, (2025).

[83] H. RATNAYAKE, L. CHEN, AND X. DING, *Privacy-preserving hierarchical federated learning with front-loaded differential privacy mechanism*, Concurrency and Computation: Practice and Experience, 37 (2025), p. e70411.

[84] R. N. REITH, T. SCHNEIDER, AND O. TKACHENKO, *Efficiently stealing your machine learning models*, in Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, 2019, pp. 198–210.

[85] F. SABAH, Y. CHEN, Z. YANG, M. AZAM, N. AHMAD, AND R. SARWAR, *Model optimization techniques in personalized federated learning: A survey*, Expert Systems with Applications, 243 (2024), p. 122874.

[86] A. SALEM, R. WEN, M. BACKES, S. MA, AND Y. ZHANG, *Dynamic backdoor attacks against machine learning models*, in 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), IEEE, 2022, pp. 703–718.

[87] G. K. SANTHANAM AND P. GRNAROVA, *Defending against adversarial attacks by leveraging an entire gan*, preprint arXiv: 1805.10652, (2018).

[88] J. SHEN, H. YANG, P. VIJAYAKUMAR, AND N. KUMAR, *A privacy-preserving and untraceable group data sharing scheme in cloud computing*, IEEE Transactions on Dependable and Secure Computing, 19 (2021), pp. 2198–2210.

[89] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[90] J. SO, B. GÜLER, AND A. S. AVESTIMEHR, *Byzantine-resilient secure federated learning*, IEEE Journal on Selected Areas in Communications, 39 (2020), pp. 2168–2181.

[91] D. SONG, J. LETTNER, P. RAJASEKARAN, Y. NA, S. VOLCKAERT, P. LARSEN, AND M. FRANZ, *Sok: Sanitizing for security*, in 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 1275–1295.

[92] H. SUN, Y. ZHANG, H. ZHUANG, J. LI, Z. XU, AND L. WU, *Pear: privacy-preserving and effective aggregation for byzantine-robust federated learning in real-world scenarios*, The Computer Journal, (2025), p. bxae086.

[93] C. SZEGEDY, W. ZAREMBA, I. SUTSKEVER, J. BRUNA, D. ERHAN, I. GOODFELLOW, AND R. FERGUS, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199, (2013).

[94] Y. K. TELILA, D. SENEVIRATHNE, D. TISSERA, A. NARAYAN, M. A. CAPRETZ, AND K. GROLINGER, *Federated learning for anomaly detection in energy consumption data: Assessing the vulnerability to adversarial attacks*, in 2025 IEEE Conference on Technologies for Sustainability (SusTech), IEEE, 2025, pp. 1–8.

[95] V. TOLPEGIN, S. TRUEX, M. E. GURSOY, AND L. LIU, *Data poisoning attacks against federated learning systems*, in Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25, Springer, 2020, pp. 480–501.

[96] F. TRAMER AND D. BONEH, *Differentially private learning needs better features (or much more data)*, preprint arXiv: 2011.11660, (2020).

[97] F. TRAMÈR, F. ZHANG, A. JUELS, M. K. REITER, AND T. RISTENPART, *Stealing machine learning models via prediction apis.*, in USENIX security symposium, vol. 16, 2016, pp. 601–618.

[98] D. USYNIN, D. RUECKERT, AND G. KAISSIS, *Beyond gradients: Exploiting adversarial priors in model inversion attacks*, ACM Transactions on Privacy and Security, 26 (2023), pp. 1–30.

[99] J. VADILLO, R. SANTANA, AND J. A. LOZANO, *Adversarial attacks in explainable machine learning: A survey of threats against models and humans*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 15 (2025), p. e1567.

[100] P. VOIGT AND A. VON DEM BUSSCHE, *The eu general data protection regulation (gdpr)*, A practical guide, 1st ed., Cham: Springer International Publishing, 10 (2017), pp. 10–5555.

[101] Y. WAN, Y. QU, W. NI, Y. XIANG, L. GAO, AND E. HOSSAIN, *Data and model poisoning backdoor attacks on wireless federated learning, and the defense mechanisms: A comprehensive survey*, IEEE Communications Surveys & Tutorials, 26 (2024), pp. 1861–1897.

[102] Y. WAN, Y. QU, W. NI, Y. XIANG, L. GAO, AND E. HOSSAIN, *Data and model poisoning backdoor attacks on wireless federated learning, and the defense mechanisms: A comprehensive survey*, IEEE Communications Surveys & Tutorials, (2024).

[103] D. WANG, W. CHEN, X. TANG, AND J. REN, *A server-side model intellectual property protection method for federated learning against model theft*, Authorea Preprints, (2025).

[104] H. WANG, Z. KAPLAN, D. NIU, AND B. LI, *Optimizing federated learning on non-iid data with reinforcement learning*, in IEEE INFOCOM 2020-IEEE conference on computer communications, IEEE, 2020, pp. 1698–1707.

[105] Z. WANG, X. GAO, C. WANG, P. CHENG, AND J. CHEN, *Efficient vertical federated unlearning via fast retraining*, ACM Transactions on Internet Technology, 24 (2024), pp. 1–22.

[106] W. WEI, L. LIU, M. LOPER, K.-H. CHOW, M. E. GURSOY, S. TRUEX, AND Y. WU, *A framework for evaluating gradient leakage attacks in federated learning*, arXiv preprint arXiv:2004.10397, (2020).

[107] R. WEN, M. BACKES, AND Y. ZHANG, *Understanding data importance in machine learning attacks: Does valuable data pose greater harm?*, arXiv preprint arXiv:2409.03741, (2024).

[108] C. WU, S. ZHU, P. MITRA, AND W. WANG, *Unlearning backdoor attacks in federated learning*, in 2024 IEEE Conference on Communications and Network Security (CNS), IEEE, 2024, pp. 1–9.

[109] X. WU, F. HUANG, Z. HU, AND H. HUANG, *Faster adaptive federated learning*, in Proceedings of the AAAI conference on artificial intelligence, vol. 37, 2023, pp. 10379–10387.

[110] C. XIE, J. WANG, Z. ZHANG, Y. ZHOU, L. XIE, AND A. YUILLE, *Mitigating adversarial effects through randomization*, in ICLR, 2018.

[111] C. XU, Y. QU, Y. XIANG, AND L. GAO, *Asynchronous federated learning on heterogeneous devices: A survey*, Computer Science Review, 50 (2023), p. 100595.

[112] L. YANG, Y. MIAO, Z. LIU, Z. LIU, X. LI, D. KUANG, H. LI, AND R. H. DENG, *Enhanced model poisoning attack and multi-strategy defense in federated learning*, IEEE Transactions on Information Forensics and Security, (2025).

[113] D. YE, S. SHEN, T. ZHU, B. LIU, AND W. ZHOU, *One parameter defense,Äîdefending against data inference attacks via differential privacy*, IEEE Transactions on Information Forensics and Security, 17 (2022), pp. 1466–1480.

[114] D. YE, T. ZHU, K. GAO, C. ZHU, AND W. ZHOU, *Cooperating or kicking out: Defending against poisoning attacks in federated learning via the evolution of cooperation*, IEEE Transactions on Dependable and Secure Computing, (2025).

[115] S. YEOM, I. GIACOMELLI, M. FREDRIKSON, AND S. JHA, *Privacy risk in machine learning: Analyzing the connection to overfitting*, in 2018 IEEE 31st computer security foundations symposium (CSF), IEEE, 2018, pp. 268–282.

[116] D. YIN, Y. CHEN, R. KANNAN, AND P. BARTLETT, *Byzantine-robust distributed learning: Towards optimal statistical rates*, in International Conference on Machine Learning, Pmlr, 2018, pp. 5650–5659.

[117] H. YIN, A. MALLYA, A. VAHDAT, J. M. ALVAREZ, J. KAUTZ, AND P. MOLCHANOV, *See through gradients: Image batch recovery via gradinversion*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16337–16346.

[118] X. YIN, Y. ZHU, AND J. HU, *A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions*, ACM Computing Surveys (CSUR), 54 (2021), pp. 1–36.

[119] W. YUAN, H. YIN, F. WU, S. ZHANG, T. HE, AND H. WANG, *Federated unlearning for on-device recommendation*, in Proceedings of the sixteenth ACM international conference on web search and data mining, 2023, pp. 393–401.

[120] H. ZHANG AND J. WANG, *Defense against adversarial attacks using feature scattering-based adversarial training*, Advances in neural information processing systems, 32 (2019).

[121] J. ZHANG, L. PAN, Q.-L. HAN, C. CHEN, S. WEN, AND Y. XIANG, *Deep learning based attack detection for cyber-physical system cybersecurity: A survey*, IEEE/CAA Journal of Automatica Sinica, 9 (2021), pp. 377–391.

[122] L. ZHANG, T. ZHU, H. ZHANG, P. XIONG, AND W. ZHOU, *Fedrecovery: Differentially private machine unlearning for federated learning frameworks*, IEEE Transactions on Information Forensics and Security, (2023).

[123] M. ZHANG, S. CHEN, J. SHEN, AND W. SUSILO, *Privacyeafl: Privacy-enhanced aggregation for federated learning in mobile crowdsensing*, IEEE Transactions on Information Forensics and Security, (2023).

[124] Q. ZHANG, C. YANG, J. LOU, L. XIONG, ET AL., *Contrastive unlearning: A contrastive approach to machine unlearning*, arXiv preprint arXiv:2401.10458, (2024).

[125] Q.-S. ZHANG AND S.-C. ZHU, *Visual interpretability for deep learning: a survey*, Frontiers of Information Technology & Electronic Engineering, 19 (2018), pp. 27–39.

[126] B. ZHAO, K. R. MOPURI, AND H. BILEN, *idlg: Improved deep leakage from gradients*, arXiv preprint arXiv:2001.02610, (2020).

[127] Y. ZHAO, J. YANG, Y. TAO, L. WANG, X. LI, D. NIYATO, AND H. V. POOR, *Exploring federated unlearning: Review, comparison, and insights*, IEEE Network, (2025).

[128] S. ZHOU, C. LIU, D. YE, T. ZHU, W. ZHOU, AND P. S. YU, *Adversarial attacks and defenses in deep learning: from a perspective of cybersecurity*, ACM Computing Surveys (CSUR), (2021).

[129] S. ZHOU, T. ZHU, D. YE, X. YU, AND W. ZHOU, *Boosting model inversion attacks with adversarial examples*, IEEE Transactions on Dependable and Secure Computing, (2023).

[130] X. ZHOU, M. XU, Y. WU, AND N. ZHENG, *Deep model poisoning attack on federated learning*, Future Internet, 13 (2021), p. 73.

[131] X. ZHOU, Y. YANG, AND H. ZHANG, *Dast: Deep adaptive defense strategy towards adversarial attacks*, in IEEE International Conference on Communications, 2020.

[132] G. ZHU, D. LI, H. GU, Y. YAO, L. FAN, AND Y. HAN, *Fedmia: An effective membership inference attack exploiting" all for one" principle in federated learning*, in Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 20643–20653.

[133] H. ZHU, J. XU, S. LIU, AND Y. JIN, *Federated learning on non-iid data: A survey*, Neurocomputing, 465 (2021), pp. 371–390.

[134] J. ZHU AND M. BLASCHKO, *R-gap: Recursive gradient attack on privacy*, arXiv preprint arXiv:2010.07733, (2020).

[135] J. ZHU AND M. BLASCHKO, *Differentially private sgd with sparse gradients*, arXiv preprint arXiv:2112.00845, (2021).

[136] L. ZHU, Z. LIU, AND S. HAN, *Deep leakage from gradients*, Advances in neural information processing systems, 32 (2019).

[137] T. ZHU, D. YE, S. ZHOU, B. LIU, AND W. ZHOU, *Label-only model inversion attacks: Attack with the least information*, IEEE Transactions on Information Forensics and Security, 18 (2022), pp. 991–1005.