

Nosy Layers, Noisy Fixes: Tackling DRAs in Federated Learning Systems using Explainable AI

Meghali Nandi
University of New South Wales
Sydney, Australia
CSIRO's Data61
m.nandi@unsw.edu.au

Arash Shaghghi
University of New South Wales
Sydney, Australia
a.shaghghi@unsw.edu.au

Nazatul Haque Sultan
CSIRO's Data61
nazatul.sultan@data61.csiro.au

Gustavo Batista
University of New South Wales
Sydney, Australia
g.batista@unsw.edu.au

Raymond K. Zhao
CSIRO's Data61
raymond.zhao@data61.csiro.au

Sanjay Jha
University of New South Wales
Sydney, Australia
sanjay.jha@unsw.edu.au

Abstract

Federated Learning (FL) has emerged as a powerful paradigm for collaborative model training while keeping client data decentralized and private. However, it is vulnerable to Data Reconstruction Attacks (DRA) such as “LoKI” and “Robbing the Fed”, where malicious models sent from the server to the client can reconstruct sensitive user data. To counter this, we introduce DRArmor, a novel defense mechanism that integrates Explainable AI with targeted detection and mitigation strategies for DRA. Unlike existing defenses that focus on the entire model, DRArmor identifies and addresses the root cause (i.e., malicious layers within the model that send gradients with malicious intent) by analyzing their contribution to the output and detecting inconsistencies in gradient values. Once these malicious layers are identified, DRArmor applies defense techniques such as noise injection, pixelation, and pruning to these layers rather than the whole model, minimizing the attack surface and preserving client data privacy. We evaluate DRArmor's performance against the advanced LoKI attack across diverse datasets, including MNIST, CIFAR-10, CIFAR-100, and ImageNet, in a 200-client FL setup. Our results demonstrate DRArmor's effectiveness in mitigating data leakage, achieving high True Positive and True Negative Rates of 0.910 and 0.890, respectively. Additionally, DRArmor maintains an average accuracy of 87%, effectively protecting client privacy without compromising model performance. Compared to existing defense mechanisms, DRArmor reduces the data leakage rate by 62.5% with datasets containing 500 samples per client.

CCS Concepts

• Security and privacy; • Computing methodologies → Machine learning;

Keywords

Federated Learning, Data Reconstruction Attack, Explainable AI

1 Introduction

In Federated Learning (FL) [31], data is not centralized for training and remains distributed between devices in the network. This ensures that raw user data never leaves the local devices, significantly reducing the risk of exposure to external threats. By sharing only

model updates rather than user device data, FL mitigates potential privacy concerns while enabling collaborative model improvement.

The promise of FL to ensure user data privacy is contingent upon the security of the gradients sent from the client nodes to the server. Studies [55] have shown that gradients sent from client nodes can reveal properties of the underlying client data or the data itself. Property inference [30, 32], membership inference [11, 36, 44], and GAN-based attacks [23, 49] have successfully inferred data stored in client nodes. A more severe class of attacks, as described in [55], involves a malicious server that tries to steal private client training data. These types of attacks fall into two categories – Optimization attack and analytical attacks. While optimization attacks [53, 55], which typically target image data, have been mitigated by secure aggregation methods in the FL architecture, analytical reconstruction attacks [8, 21] remain a threat. In optimization attacks, a random dummy sample is initialized and then the difference between the true gradient and the generated one is optimized. However, the quality of reconstructions degrades with the increasing size of batch size [53]. On the other hand, analytical attacks customize model parameters or model architecture to extract training data from the model layers.

One of the most notable Data Reconstruction Attacks (DRA) is the “Robbing the Fed” (RtF) attack [21], which demonstrates how minimal but malicious modifications to the shared model architecture can enable the server to directly obtain a verbatim copy of user data from gradient updates. This attack is particularly concerning because it bypasses the need for solving complex inverse problems, making it highly efficient and scalable. The implications of such an attack can be severe, as it can compromise the privacy of user data even when aggregated over large batches. A more recent attack, LoKI, is proposed by authors in [55]. LoKI overcomes the limitations of RtF by targeting the FedAVG setting and using customized convolutional parameters to maintain the separation of weight gradients between clients, even through aggregation. This attack has shown the ability to leak a substantial portion of training data, with success rates of 76–86% in a single training round, even when secure aggregation is employed. The effectiveness of LoKI further highlights the vulnerability of FL systems to sophisticated model manipulation techniques.

An increasing number of solutions have emerged to protect the privacy of data in FL. Secure aggregation methods [10, 40] have been

proposed to mitigate such threats. However, they remain vulnerable to advanced attacks such as LoKI. Byzantine-resilient aggregation techniques [45, 56] can identify and exclude outlier gradients that deviate significantly from the majority, thereby reducing the risk of data leakage. However, in these methods, the server can manipulate the weight gradients to a normalized value, so that the gradients are not detected. Anomaly detection algorithms [35] can monitor the training process for unusual patterns indicative of an attack. However, skilled attackers can design their malicious updates to evade detection by anomaly detection algorithms, reducing their effectiveness. Additionally, differential privacy (DP) [24, 26, 50] involves adding noise to the model updates before they are sent to the server, which can prevent a server from fully reconstructing private data. However, this method significantly decreases model performance, especially with large vision models [54]. As mentioned in [55], because the attack could occur at any point in the training process, differential privacy would need to be applied at every training step, making it extremely costly in terms of accuracy.

The current solutions fall short against sophisticated attacks without compromising the utility of FL models. Existing approaches predominantly treat models as black-box systems, limiting their ability to address the underlying vulnerabilities. The authors in [55] emphasize that identifying modified layers in a model is particularly challenging due to their potential to be embedded in various locations within a larger architecture. This challenge arises because current solutions do not analyze the inner workings of the model’s layers – the very components that might facilitate DRA. To bridge this gap, we propose a novel methodology that treats the model as *a white-box system by focusing on the layers involved in gradient updates*. By leveraging Explainable AI (XAI), our approach seeks to understand how these layers contribute to the model’s output and identify those whose gradients deviate from normal behavior, indicating an intent to reconstruct data rather than improve the model’s performance. Further, most related research [42] focuses on server-side mechanisms or aggregation methods to safeguard the data of client nodes. In this work, our goal is to *identify the root cause of the attack on the client side*, irrespective of the aggregation algorithms on the server side. To the best of our knowledge, this is the first work to take advantage of explainable AI to design a robust algorithm to detect malicious layers in a modified model architecture aimed at reconstructing client data in FL.

Our work. We introduce DRArmor, a novel system designed to detect and mitigate malicious layers in DRA-modified models, such as those introduced in Rtf [21] and LoKI [55]. By leveraging XAI techniques, including Layer-wise Relevance Propagation (LRP) [4] and Deep Taylor Decomposition (DTD) [34] on these replicated attacks, DRArmor explains how each layer of a model contributes to its output. Layers exhibiting low relevance but high learning—a characteristic often linked to malicious intent—are flagged by the system. These layers are then addressed and mitigated, ensuring only the truly relevant layers remain in the model. This process significantly enhances the security of client devices in FL environments. Unlike traditional approaches that rely heavily on centralized server-side defenses, DRArmor enables clients to take proactive control by detecting and responding to DRA attacks independently. This shift to client-side protection addresses the root cause of such attacks more effectively. Our evaluation demonstrates the system’s

effectiveness, achieving True Positive Rates (TPR) of 0.910 and True Negative Rates (TNR) of 0.890 across multiple datasets. When malicious layers are identified, DRArmor employs targeted techniques such as adding DP noise or pixelated noise solely to these layers. This approach is more effective than applying standard DP methods (e.g., [26]) across the model. Additionally, DRArmor supports pruning irrelevant layers along with preserving model performance. The system’s robustness is further highlighted by its ability to maintain high detection accuracy and significantly reduce data leakage, even as the number of client nodes and dataset sizes increase. Moreover, DRArmor is effective against both continuous and periodic poisoning attacks, ensuring that model accuracy remains intact with minimal degradation.

Our main contributions are:

- (1) We introduce DRArmor, a novel defense solution that uses XAI to detect and mitigate malicious layers in DRA-modified models. Even with extended replications of attacks such as LoKI with 200 clients and dataset sizes of 500 per round, DRArmor achieved 87% detection accuracy, including cases where malicious layers were deeply embedded in the model architecture.
- (2) Unlike traditional server-side methods or aggregation defenses on the entire model, DRArmor empowers client devices to identify and defend against malicious layers independently. DRArmor applies targeted defenses, such as DP noise and pixelation, exclusively to malicious layers, leaving the rest of the model unaffected.
- (3) DRArmor consistently detects and mitigates DRA across iterations, demonstrating robustness against both continuous and periodic poisoning attacks while maintaining model accuracy.
- (4) Compared to existing defense mechanisms, DRArmor reduces the data leakage rate by 62.5% on networks with 200 clients and datasets of 500 samples per client.

2 Background and Related Work

2.1 DRA on FL

While numerous attacks on FL have been proposed, our focus is on defenses against DRA, in which the model architecture is altered to directly obtain client data through their updates. These attacks exploit vulnerabilities in the gradient calculation and sharing processes in FL.

Optimisation-based attacks [17, 53] operate by iteratively refining a dummy data sample to match the gradient of the actual training data.

Although these attacks are effective with small batch sizes, they suffer from decreased quality and require more iterations as batch sizes increase. They often fail with larger image resolutions and aggregated gradients, and they are largely limited to FedSGD [31]. Some of these methods assume knowledge of the user labels, which are obtained outside of the optimisation process. Additionally, optimisation - based attacks can produce reconstructed images that display characteristics of the image class but are not actually present in the training image, as shown in [53].

Different methods have been proposed targeting FL with secure aggregation, often with limited success. One approach [51] is to

magnify gradients to target a single data point. This involves manipulating weights so that the gradients of a targeted class are magnified by reducing model confidence in that class’s prediction. Another approach [39] aims to make the aggregated gradient the same as an individual client’s gradient. This can be achieved by sending model parameters that result in zero gradients for all but the targeted client. There are also methods [25] that treat the inputs of an FC layer as a blind source separation problem. However, these methods are often limited in scale, typically only managing to extract data from a single user per training round or having restrictions on the number of inputs.

Analytic attacks, particularly those involving linear layer leakage [8, 21], directly extract training data from the gradients of FC layers. These attacks modify model parameters or architecture to retrieve inputs from the weight and bias gradients of an FC layer. When only one data sample activates a neuron in an FC layer, the input can be directly computed from the gradients. These methods can reconstruct inputs with high accuracy but typically fail when multiple data samples activate the same neuron. As shown in [55], they also face scalability issues when dealing with larger batch sizes or secure aggregation, which conceals individual client updates. While increasing the size of the FC layer can maintain a high leakage rate, it often results in very large models. LoKI improves DRAs by introducing techniques to overcome scalability challenges and secure aggregation limitations. Unlike the previous methods, it enables simultaneous data reconstruction from multiple clients in a single training round, addressing the inefficiencies of optimisation-based and analytic attacks. Key features of LoKI include the use of customized convolutional kernels for each client, allowing the server to separate and recover client-specific data even after aggregation. This attack highlights critical vulnerabilities in FL systems and motivates the development of robust defenses.

2.2 Existing Defense Mechanisms

Differential Privacy (DP). DP [18] is a privacy-preserving technique that introduces random noise to the data or model updates to protect individual data points from being inferred. In the context of FL, DP can be applied by adding noise to the gradients before they are shared with the server as surveyed in [19]. This noise addition ensures that the contribution of any single data point is obscured, making it difficult for an attacker to reconstruct the original data.

Mathematically, differential privacy guarantees for any two datasets D and D' differing by a single data point, the probability of obtaining a particular output O is nearly the same. This can be expressed as: $\Pr[\mathcal{M}(D) = O] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') = O]$, where \mathcal{M} is the randomized mechanism (e.g., the gradient update process), and ϵ is the privacy budget that controls the level of privacy. A smaller ϵ indicates stronger privacy but may introduce more noise.

In the context of DRA, the added noise affects the reconstructed data by making the gradients less precise. Suppose the true gradient is g , and the noise added is drawn from a Laplace distribution with scale parameter $\frac{\Delta g}{\epsilon}$, where Δg is the sensitivity of the gradient. The noisy gradient g' can be represented as: $g' = g + \text{Lap}\left(\frac{\Delta g}{\epsilon}\right)$.

The effectiveness of DP in preventing DRA is due to the noise, which makes it challenging for the attacker to accurately infer the true data - especially when the privacy ϵ is small. Mironov [33] used

Rényi divergence for more accurate privacy loss calculations, and Abadi et al. [1] created DP-SGD, an empirical algorithm that applies DP to machine learning training. Other research [2, 3, 9, 52] builds upon DP-SGD by improving its performance through methods such as adaptive gradient clipping and suitable perturbations. Authors in [28] also apply DP to FL to protect user-level privacy and ensuring data privacy when collecting local parameters. However, this comes at the cost of potentially degrading the model’s performance, highlighting the trade-off between privacy and utility as shown in [12, 16].

Gradient compression is also commonly adopted by studies like [53] as a baseline defense to examine the robustness of attacks. Recently, Soteria [46] leveraged a similar idea to gradient compression but with a smarter pruning strategy to decrease privacy leakage risks with similar performance. Recent works such as [5] evaluated these defense methods and broke them by introducing a Bayesian optimal adversary. The Bayesian framework provides a way to analyse the problem of gradient leakage by considering the joint probability distribution of inputs and their gradients. This framework allows for the formulation of a Bayes optimal adversary as an optimisation problem, which minimises the risk of reconstructing the input given the observed gradients. The paper demonstrates that existing attacks can be interpreted as approximations of this optimal adversary, each making different assumptions about the underlying probability distributions. This theoretical grounding enables the development of stronger attacks by leveraging the knowledge of these distributions, proving more effective than previous methods.

Limiting Mutual Information (MI). Several studies [6, 13] explore the connection between MI and DP by deriving upper bounds of MI for different DP mechanisms. Other studies [29, 37] propose training feature extractors that reduce the MI between the output features and the assigned labels while maximising the MI between the output features and the original data. Fisher information has also been used to measure information leakage, aligning with the theories in [22]. Authors in [48] model FL using information theory to measure MI between variables in FL. A more recent study [47] proves that the reconstruction error of DRA is directly related to the information an attacker gains through shared parameters, as measured by MI. To limit this information leakage, the method establishes a channel model where the shared parameters act as a communication channel and then limits the channel capacity by adding Gaussian noise to the parameters. This effectively reduces the information available to an attacker, making data reconstruction challenging.

The defense methods reviewed may effectively limit the information shared by client devices with the server. However, they do not address one fundamental issue: *client devices cannot detect how or where gradients are being leaked*. Moreover, these methods lack transparency in explaining their effectiveness. We postulate that if the nature of the attack evolves, these approaches may fail in detection. As mentioned by the authors of LoKI, since the malicious layers can be placed at any part of the model architecture, it would be challenging to detect such attacks if the attacker modifies the parameters. Additionally, our approach offers computational savings as the defense mechanism is applied only to the malicious layers rather than the entire model.

2.3 Preliminary - Explainable AI

Explainable AI (XAI) refers to techniques used to provide insights into how models arrive at their predictions, enabling users to understand, trust, and effectively manage AI systems. It helps identify biases and improve model performance. There are multiple varieties of XAI methods. In this work, we are using the following:

Layer-wise Relevance Propagation (LRP). LRP is a technique used to explain the predictions of deep neural networks by propagating the prediction backward through the network. LRP assigns relevance scores to each input feature, indicating their contribution to the final prediction. Mathematically, the relevance score R_j for a neuron j in layer l is computed by distributing the relevance from the neurons in the subsequent layer $l + 1$:

$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon \cdot \text{sign}(\sum_{i'} z_{i'j})} R_j^{(l+1)} \quad (1)$$

where z_{ij} is the contribution of neuron i in layer l to neuron j in layer $l + 1$, ϵ is a small stabilization term to prevent division by zero, and $R_j^{(l+1)}$ is the relevance of neuron j in the next layer $l + 1$. LRP helps in visualizing which parts of the input data are most relevant to the model's output, making it easier to interpret complex neural networks and identify potential issues.

Deep Taylor Decomposition (DTD). DTD is an advanced method for explaining the predictions of deep neural networks by decomposing the prediction into contributions from each input feature. DTD is based on Taylor series expansion, which approximates a function using its derivatives. Mathematically, the relevance R_i of an input feature x_i is computed by decomposing the relevance of the output R into contributions from the input features: $R_i = x_i \frac{\partial R}{\partial x_i}$. By iteratively applying this decomposition from the output layer to the input layer, DTD provides a detailed explanation of the model's decision-making process, highlighting the importance of each input feature.

3 System and Threat Model

System Model. The components of the FL-based system used in this work include Server, Client Device, and Attacker. The Server is responsible for registering a model, iteratively sending it to the clients, aggregating the client updates, and training the global model. The Client Device is responsible for storing the data, training the model received from the server, and sending updates back to the server. The Attacker is responsible for altering model parameters and architecture to obtain updates from the client, with the goal of reconstructing the client's data. The system consists of one server, multiple client devices, and multiple attackers. The client devices communicate directly with the server and train the model received from the server. We consider a real-life FL scenario in which any device can receive any model, and the client cannot assume that a model received in the iteration after a malicious one will also be malicious. Unlike the studies in [55] and [21], which used 100 clients, our experiments scale up to 200 clients to evaluate the impact of these attacks on model accuracy in a larger setting.

Threat Model.

We consider a standard FL setting where a central server coordinates model training across multiple clients, each holding private local data. The server initializes and distributes a global model,

while clients perform local training and return gradient updates for aggregation. In our threat model—adopted from LoKI [55] and originally introduced in [21]—the server follows the FL protocol but may attempt to reconstruct private client data by modifying the model architecture before the distribution. Specifically, the server can inject malicious layers into the model to exploit gradient leakage for DRA. However, the server cannot deviate from the standard federated learning protocol in ways beyond changes to model architecture or parameters, and such modifications must remain within the operational limits of standard machine learning frameworks. Clients are assumed to be benign, unaware of these manipulations, and follow the protocol by training the received model on local data and returning gradients. The adversarial model includes a single attacker—the server—with no client-side collusion.

4 DRArmor

DRArmor is designed to address the critical challenge of detecting and mitigating malicious layers introduced by DRA in FL systems. By leveraging Explainable AI (XAI), DRArmor analyzes the model architecture as a white-box, examining the model architecture and the weights of each layer to detect any modifications indicative of malicious intent.

As described in [21, 55], the input x_i is calculated by dividing the weight gradient by the bias gradient, as shown in the equation: $x_i = \frac{\delta L / \delta W_i}{\delta L / \delta B_i}$, where i is the activated neuron, and $\frac{\delta L}{\delta W_i}$ and $\frac{\delta L}{\delta B_i}$ are the weight and bias gradients of the neuron, respectively. In the context of detecting malicious layers, the variables $\frac{\delta L}{\delta W_i}$ and $\frac{\delta L}{\delta B_i}$ play a critical role. Malicious layers, designed to reconstruct private data rather than contribute meaningfully to the overall model performance, exhibit gradients that are not aligned with the model's intended learning objectives. Specifically, the weight gradients $\frac{\delta L}{\delta W_i}$ in these layers reflect manipulation aimed at data leakage, leading to anomalous behavior such as higher magnitude or misalignment with gradients of other layers. These manipulated gradients fail to contribute to improving the model's predictive performance, making them indicative of malicious intent. This often indicates that the model is trying to extract information that does not significantly contribute to the final output. This is where Explainable AI (XAI) becomes crucial. XAI techniques help us understand which neural network layers are actually contributing to the model's output. For instance, if a model is designed to classify an image as a cat, certain layers focusing on reconstructing the data might not directly contribute to the final classification of "cat". These layers might produce higher gradients than those more relevant to the classification task. By using XAI, we can identify and understand the roles of different layers in the model, ensuring that the model's decision-making process is transparent and interpretable.

Two of the most popular methods for understanding the relevance of a layer to the output are Layer-wise Relevance Propagation (LRP) and Deep Taylor Decomposition (DTD). Our method leverages the explainability provided by LRP and DTD to identify layers that contribute disproportionately to the model's output, which indicates potential malicious behavior.

Suppose we have a neural network model designed to classify images along with the additional malicious layers described in [55], and we input an image of a cat. These layers might perform a DRA,

making them irrelevant to the final classification task. The standard propagation rule for relevance in LRP is given by Equation 1.

To tailor LRP to our needs, we modify the propagation rule to emphasize the discrepancy between the intended contributions of layers and their actual gradient behaviors. Specifically, for a layer l , we adapt the rule as: $R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon} \cdot \gamma \cdot R_j^{(l+1)}$, where $\gamma \in [0, 1]$ is a scaling factor that adjusts the contribution of relevance attribution of the previous layer. It is computed adaptively to reflect the model's certainty about the importance of a given layer. Layers that exhibit high gradients and are deemed highly relevant contribute more strongly to the propagation through a higher γ , while less relevant or low-confidence layers result in a smaller γ . This dynamic adjustment ensures that the propagation process emphasizes layers that are both significant and exhibit confident behavior in terms of relevance attribution.

4.1 Detecting Malicious Layers

In a benign model, the layers collaborate to generate the output by focusing on relevant input features.

Malicious layers are modified to reconstruct input data rather than contributing to output generation. This discrepancy is manifested in their gradients and relevance scores:

- **Relevance Scores:** Malicious layers contribute minimally to the output relevance. When propagating relevance using the modified LRP rule, these layers will exhibit low relevance scores ($R^{(l)} \approx 0$) since their primary purpose is not aligned with the model's intended task.

- **Gradient Analysis:** Malicious layers are designed to reconstruct input data, leading to unusually high gradients with respect to input features. Mathematically, the gradient $\nabla_{\mathbf{x}} f^{(l)}$ of the malicious layers will exhibit large magnitudes, indicating their strong dependence on the input data for reconstruction. Gradient analysis is incorporated by evaluating the norm of gradients $\nabla_{\mathbf{x}} f^{(l)}$ for each layer: $G^{(l)} = \|\nabla_{\mathbf{x}} f^{(l)}\|$, where $G^{(l)}$ captures the gradient of the layer with respect to input features for each layer l . Layers with anomalously high gradients with low relevance are flagged as potentially malicious, as they disproportionately influence data reconstruction.

Gradient-Relevance Discrepancy. To formally identify malicious layers, we define a metric based on the gradient relevance discrepancy. For each layer l , we compute: $D^{(l)} = \frac{\|\nabla_{\mathbf{x}} f^{(l)}\|}{\|R^{(l)}\|}$, where $\|\cdot\|$ denotes the norm (e.g., L_2 norm).

A high value of $D^{(l)}$ indicates a layer with large gradients but low relevance, characteristic of malicious behavior. Using a threshold τ , we classify a layer l as malicious if: $D^{(l)} > \tau$.

Justification for LRP Modifications. The modifications to the standard LRP formulation are critical for adapting it to the detection of malicious layers. By introducing the scaling factor γ , we ensure that the propagation process captures subtle but critical deviations in layer behavior. For example, benign layers that contribute to the output will maintain proportional relevance scores under this adjustment, while malicious layers will show diminished relevance.

Furthermore, by computing the discrepancy $D^{(l)}$, we leverage the complementary information provided by gradients and relevance. Gradients highlight sensitivity to input changes, while relevance scores reflect actual contribution to the output. Malicious layers, which prioritize data reconstruction over task alignment, are naturally distinguished by this divergence (see §6.1).

While LRP is effective for small and moderately sized models, its performance degrades with deeper architectures and models with large numbers of parameters. These limitations arise due to:

- **Vanishing Relevance:** As relevance is propagated backward through multiple layers, it may diminish or concentrate disproportionately in certain neurons, making it challenging to identify malicious layers accurately.

- **Sensitivity to Stabilization Terms:** The stabilization term ϵ can significantly impact the relevance propagation process, leading to inconsistent results for different architectures.

- **Computational Overhead:** For large-scale models, the backward relevance propagation requires substantial computational resources, making it computationally expensive for large-scale analysis (see Appendix A).

To address these limitations, we propose using DTD as an alternative for large-scale models. DTD provides a more robust mechanism for layer-wise relevance analysis by decomposing the model's output into contributions from each layer using a Taylor series expansion (see §6.1). The incorporation of Wasserstein distance further refines this process by quantifying the similarity between reconstructed relevance distributions and expected distributions.

For a neural network $f(\mathbf{x})$ with output y , DTD approximates the relevance of layer l by considering the Taylor expansion of f around a root point \mathbf{x}_0 (typically chosen to minimize f):

$$R^{(l)} = f(\mathbf{x}) - f(\mathbf{x}_0) \approx \nabla_{\mathbf{x}} f^{(l)} \cdot (\mathbf{x} - \mathbf{x}_0).$$

Unlike LRP, DTD derives relevance scores directly from the gradient information relative to an input perturbation, making it more robust against the vanishing relevance issue that affects LRP. The term $(\mathbf{x} - \mathbf{x}_0)$ ensures that the decomposition aligns with localized input changes, capturing the contributions of malicious layers more effectively. A key advantage of DTD is that it avoids the dependency on stabilization terms (ϵ) that can introduce inconsistencies in LRP. Mathematically, DTD provides a more accurate decomposition by considering higher-order derivatives implicitly, ensuring that the relevance scores remain consistent across deep architectures.

Wasserstein Distance for Consecutive Layers. The Wasserstein distance is a metric for comparing probability distributions. In this context, we use it to measure the similarity between relevance distributions of consecutive layers $R^{(l)}$ and $R^{(l+1)}$. Mathematically, the Wasserstein distance is defined as:

$$W(R^{(l)}, R^{(l+1)}) = \inf_{\gamma \in \Gamma(R^{(l)}, R^{(l+1)})} \int \|r - r'\| d\gamma(r, r'),$$

where $\Gamma(R^{(l)}, R^{(l+1)})$ is the set of all joint distributions with marginals $R^{(l)}$ and $R^{(l+1)}$. A high Wasserstein distance between consecutive layers indicates a significant deviation in their relevance behavior, flagging the layer as potentially malicious.

By applying Wasserstein distance to consecutive layers, we capture the transition irregularities introduced by malicious layers.

This approach avoids reliance on predefined reference distributions.

The defense methodology using DTD and Wasserstein distance is implemented as follows:

- (1) Perform DTD to compute relevance scores $R^{(l)}$ for all layers based on their Taylor expansion contributions.
- (2) Calculate the Wasserstein distance $W(R^{(l)}, R^{(l+1)})$ for each pair of consecutive layers.
- (3) Compute the gradient-relevance discrepancy $D^{(l)}$ for additional validation: $D^{(l)} = \frac{\|\nabla_x f^{(l)}\|}{\|R^{(l)}\|}$.
- (4) Flag layers with $W(R^{(l)}, R^{(l+1)}) > \tau_W$ or $D^{(l)} > \tau_D$ as malicious.
- (5) Retrain or replace flagged layers to mitigate their impact on the model.

DTD, as used in DRArmor, is gradient-based and avoids recursive backpasses, making it significantly more efficient and well-suited for typical federated clients with limited resources. Similarly, the Wasserstein distance computation is applied once per round for layer-wise distributions and can be computed in linear time with entropic regularization. These components are executed once per local round (not per batch or epoch), thus incur minimal additional overhead relative to standard local training.

4.2 Defense against Malicious Layers

Once malicious layers are identified within the model, the next critical step involves implementing appropriate defensive actions to prevent DRA. The following strategies can be employed as mitigation techniques:

Differential Privacy-Based Noise Addition. Differential Privacy (DP) is a widely recognized strategy to mitigate information leakage in FL. While the authors of LoKI suggest that applying DP universally by adding noise to all gradient updates can significantly reduce model accuracy, a more targeted approach can strike a balance between privacy and performance. Instead of adding noise to all gradient updates, noise is injected specifically into the updates originating from the malicious layers. This targeted application minimises the impact on overall model accuracy while effectively mitigating the DRA. Since only the malicious layers are affected, the integrity of legitimate layers remains intact, preserving the utility of the model. Moreover, noise levels can be dynamically adjusted based on the confidence in the detection of malicious layers. For instance, layers identified with higher certainty can have more noise added, while those with marginal detection can have less aggressive noise injection.

Let $\mathbf{g} = [g_1, g_2, \dots, g_L]$ represent the gradients of a model with L layers, where g_l is the gradient vector of the l -th layer. Assume $\mathcal{M} \subset \{1, 2, \dots, L\}$ denotes the set of malicious layers identified by the detection mechanism. For each $l \in \mathcal{M}$, noise is added as follows: $\tilde{g}_l = g_l + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma^2)$ is Gaussian noise with mean 0 and variance σ^2 . Layers not identified as malicious remain unchanged, i.e., $\tilde{g}_l = g_l$ for $l \notin \mathcal{M}$.

The noise level σ^2 can be dynamically adjusted based on the confidence score c_l for identifying layer l as malicious: $\sigma_l^2 = \sigma_{\text{base}}^2 \cdot (1 + \alpha \cdot c_l)$, where σ_{base}^2 is base noise level, $c_l \in [0, 1]$ is confidence

score for layer l being malicious, and α is scaling factor that controls the impact of confidence on noise level.

Higher confidence in detecting a malicious layer results in larger noise being added, ensuring better obfuscation of sensitive information.

After noise injection, the updated gradients $\tilde{\mathbf{g}}$ are aggregated and sent to the server: $\tilde{\mathbf{g}} = [\tilde{g}_1, \dots, \tilde{g}_L]$. This ensures that gradients from malicious layers are obfuscated while preserving the integrity of legitimate layer gradients.

Targeted DP minimizes the degradation of model performance. The change in accuracy due to noise application, denoted as Δ_{acc} , is given by: $\Delta_{\text{acc}} = \text{Accuracy}_{\text{no DP}} - \text{Accuracy}_{\text{DP}}$. Since only malicious layers are affected, Δ_{acc} remains significantly smaller compared to universal DP application. (see §6.3)

Pixelating gradients. Pixelation involves reducing the gradient by grouping adjacent values into blocks and replacing them with a single representative value, such as the average or median intensity of the block. By applying pixelation to the gradients sent from malicious layers, the precision of the gradient information is reduced, thereby limiting the ability of the server to reconstruct the original client data.

Given a gradient matrix G of dimensions $m \times n$, the pixelation process operates as follows:

The gradient matrix G is divided into non-overlapping blocks of size $b \times b$. The total number of blocks is given by: $N = \frac{m}{b} \times \frac{n}{b}$, where b is the block size, chosen such that m and n are divisible by b . Each block B_{ij} corresponds to a sub-matrix: $B_{ij} = G[i \cdot b : (i+1) \cdot b, j \cdot b : (j+1) \cdot b]$, where $i \in \{0, \dots, \frac{m}{b} - 1\}$ and $j \in \{0, \dots, \frac{n}{b} - 1\}$.

Each block B_{ij} is replaced by the representative mean: $\hat{B}_{ij} = \text{mean}(B_{ij}) = \frac{1}{b^2} \sum_{p=1}^b \sum_{q=1}^b G_{p,q}$. The pixelated gradient matrix \hat{G} is constructed by replacing each block B_{ij} in G with its corresponding representative value \hat{B}_{ij} : $\hat{G}[i \cdot b : (i+1) \cdot b, j \cdot b : (j+1) \cdot b] = \hat{B}_{ij}$.

By pixelating gradients G from malicious layers, the client can obfuscate sensitive information, reducing the precision of data reconstruction attempts. The server's ability to reconstruct client data is typically dependent on G , which with pixelated gradients, becomes \hat{G} . The quantization effect introduced by pixelation disrupts the reconstruction process, protecting client data while retaining the model's learning capacity (see §6.3).

Pruning Malicious Layers. Another effective strategy involves pruning the identified malicious layers entirely from the model architecture. This approach relies on bypassing the malicious layers while maintaining the flow of relevant information through the network. After pruning a malicious layer, a skip connection can be introduced to route the input image (or feature map) directly to the next relevant layers. Ensuring that the dimensions match is crucial for the network to function correctly when implementing skip connections. If they do not match, we can use techniques like padding, cropping, or applying a linear transformation to adjust the dimensions. In our experiment, we use a linear layer along with padding to transform the input to the required dimensions. Consider an input image of size $(64 \times 64 \times 3)$ and a target convolutional layer (non-malicious) with a filter size of (3×3) and 32 filters as an example. First, we apply an initial convolutional layer with the same filter size and number of filters to the input image, using a stride of 1 and padding to maintain the spatial dimensions. This results

in an output size of $(64 \times 64 \times 32)$. Next, we bypass the potentially malicious layer by directly connecting the adjusted input to the third convolutional layer. To ensure dimensional compatibility, we use a convolutional layer with the same filter size and number of filters as the target layer, resulting in an adjusted output size of $(64 \times 64 \times 32)$. This preprocessing step ensures that the input image can be effectively integrated into the network at the desired point, preserving the data’s integrity and enhancing the network’s robustness.

5 Experimental Setup

The experiments were carried out in a simulated FL environment, where we implemented the defense system components using the Keras and Flower [7] frameworks. The system was executed on a server equipped with an NVIDIA H1000NVL GPU, an Intel Xeon 2.10 GHz CPU with 8 cores, and 96 GB of RAM. The system’s scalability was validated by simulating 200 clients, a scenario representative of real-world FL environments. We selected widely used datasets for our experiments, including MNIST, CIFAR-10, CIFAR-100, ImageNet, and Cats vs Dogs. These datasets were chosen because they are commonly used in related research for evaluating FL systems and defense mechanisms, offering a benchmark for comparison. Additionally, their diverse characteristics – ranging from simple grayscale digits to complex natural images – enable us to thoroughly assess the effectiveness of our approach across various data types and scenarios. The associated model configurations for each dataset are detailed below.

MNIST [15]. This is a set of 60,000 handwritten numbers from 0–9. Each data consists of 28×28 pixel grayscale images. We used a CNN model with three convolutional layers, where the first layer consists of 32 output channels with kernel size 3 and stride 1. The second convolutional layer outputs 64 channels with the same kernel size and stride of 3 and 1, respectively. The third convolutional layer outputs 256 channels with the same kernel size and stride of 3. It is followed by two dropout layers and two fully connected layers. The final output is given by a log softmax layer that predicts the number denoted in the input image.

CIFAR-100 [27]. This dataset consists of 60,000 32×32 color images in 100 classes, with 600 images per class. There are 50,000 training images and 10,000 test images. The model begins with an initial convolutional layer with 64 output channels, a kernel size of 3, and a stride of 1, followed by batch normalization and a ReLU activation. Subsequent layers consist of repeated residual blocks, each comprising two convolutional layers with 64, 128, and 256 output channels as the depth increases. Each convolution is followed by batch normalization and ReLU activation. To handle overfitting, dropout layers are interspersed between some convolutional blocks. After the convolutional and residual layers, global average pooling is applied to reduce the spatial dimensions. The fully connected layers include one hidden layer of 512 units followed by a final output layer with 10 units corresponding to the CIFAR-10 classes. The final predictions are produced using a log softmax layer.

CIFAR-10 [27]. This dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. We used the same model as used for CIFAR-10.

ImageNet [14]. This dataset contains over 14 million images across 1,000 classes. For our experiments, we used a subset of 1.2 million training images and 50,000 validation images. The model begins with an initial convolutional layer that outputs 64 channels, using a kernel size of 7 and a stride of 2, followed by batch normalization, ReLU activation, and max pooling to reduce spatial dimensions. The subsequent layers are structured as residual blocks. Each block contains two convolutional layers with increasing output channels – 64, 128, 256, and 512 – as the network depth increases. All convolutions are followed by batch normalization and ReLU activation, with downsampling implemented through strided convolutions. To prevent overfitting and improve generalization, dropout layers are interspersed between some residual blocks. The feature maps are then reduced using global average pooling before being passed through two fully connected layers. The final output layer consists of 1,000 units, corresponding to the ImageNet classes, and predictions are generated using a log softmax function.

Cats v Dogs [38]. For this dataset, we employed the widely-used ResNet-18 architecture, which balances computational efficiency and performance.

Each dataset was divided into training, testing, and validation sets, with a sampling ratio of 60%, 30%, and 10%, respectively. The primary objective of our experiments was to detect and defend against malicious layers attempting to execute DRA within a system of 200 clients – twice the number used in the LoKI attack. All models were modified by adding a convolutional layer and two FC layers, initiating with the start of the model and then proceeding further into the model architecture. Additionally, we evaluated the behavior of DRArmor under varying numbers of layers, with the results presented in §6. In all experimental settings, we used a learning rate of $\alpha = 1 \times 10^{-4}$. Local models were trained for 15 to 200 rounds, depending on the complexity of the model architecture and the size of the dataset. The training process aimed to achieve stable accuracy while avoiding overfitting, ensuring the reliability of the evaluation.

6 Evaluation

We begin by visualizing the detection of malicious layers to offer an intuitive understanding of the mechanism (§6.1). Next, we analyze its accuracy in identifying malicious layers (§6.2) and evaluate the system’s performance by comparing accuracy, leakage rate, and the number of leaked images with those of other systems. This is followed by an assessment of DRArmor’s defense capabilities compared to other defense mechanisms (§6.3). Additionally, we evaluate the system’s performance by comparing metrics such as accuracy, leakage rate, and the number of leaked images with those of other systems. Furthermore, we compute key metrics, including True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR), to assess the mechanism under various experimental settings.

6.1 Detection of Malicious Layer

The impact of the placement of malicious layers on the effectiveness of DRArmor is analyzed below.

Malicious Layers at the Start of the Model. The original model architecture was modified to include convolutional layers

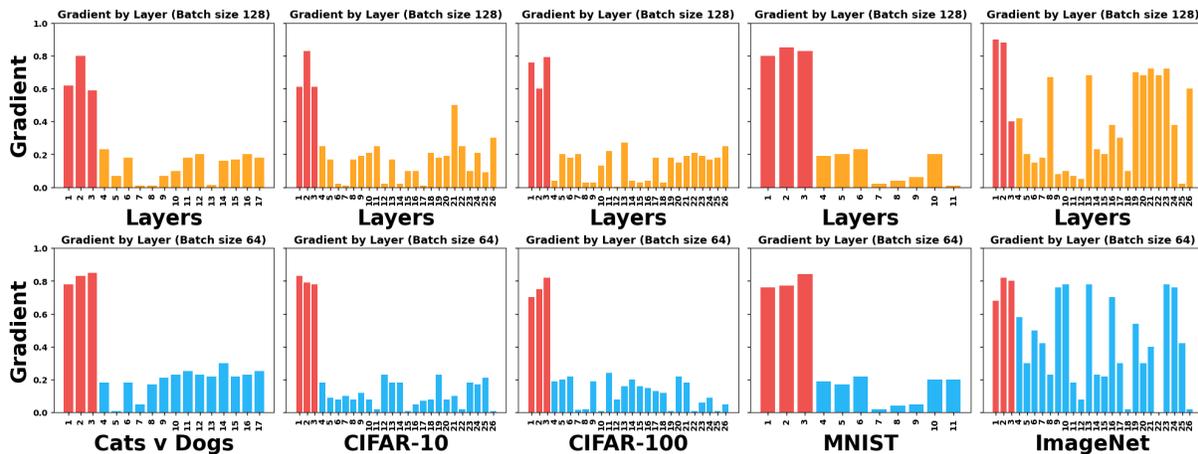


Figure 1: Gradient Analysis Across Datasets: Identifying Malicious Layers at the Start of the Model using LRP (Batch Sizes: 64, 128) with threshold value of 0.5.

and FC layers at the beginning. To analyze the behavior of the model, LRP was applied after training the model architecture in the client devices. The training process was carried out using batch sizes of 64 and 128. Experiments were carried out on five datasets – Cats v Dogs, CIFAR-10, CIFAR-100, ImageNet, and MNIST – using models described in Section 5. Each architecture contained three malicious layers, as introduced in the LoKI paper. These malicious layers were deliberately designed to learn irrelevant features from the data, thereby disrupting the model’s ability to produce meaningful outputs. The goal of the experiments was to evaluate the effectiveness of LRP in detecting these malicious layers. Upon applying LRP, the method demonstrated its ability to identify malicious layers effectively in most datasets. Figure 1 illustrates this result, where the gradient values for the malicious layers were significantly higher compared to those of non-malicious layers. This is because, while the majority of layers contributed meaningfully to the model’s output, the malicious layers displayed anomalous behavior by learning features that were irrelevant to the task. The higher scores for these layers indicate their deviation from expected behavior, making them stand out in the relevance analysis.

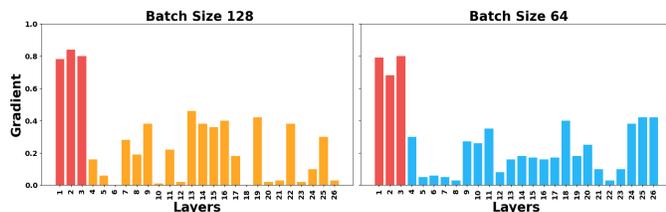


Figure 2: Gradient Analysis in ImageNet: Identifying Malicious Layers at the Start of the Model using DTD with threshold value of 0.5.

The results for the ImageNet dataset were less promising. LRP struggled to effectively differentiate between malicious and non-malicious layers for this dataset. This reduced effectiveness can

be attributed to the complexity of the ImageNet dataset, which contains a large number of classes and significantly more diverse and intricate data compared to the other datasets. Models trained on ImageNet require highly detailed feature representations, which can dilute the relevance scores, leading to difficulty in isolating malicious layers. Unlike LRP, DTD provides a more refined decomposition of relevance scores, enabling it to handle complex scenarios more effectively. As shown in Figure 2, DTD improved the detection of malicious layers in the ImageNet model. Although the difference between the relevance values of malicious and non-malicious layers was less pronounced compared to simpler datasets, DTD offered a more robust solution for this complex task.

The comparison between LRP and DTD highlights the need for adaptive analysis techniques depending on the dataset’s complexity. For simpler datasets like Cats v Dogs, CIFAR-10, CIFAR-100, and MNIST, LRP was sufficient to identify malicious layers with high accuracy. In contrast, for the highly complex ImageNet dataset, the limitations of LRP necessitated the use of DTD to achieve better results.

Malicious Layers Deeper in the Model. The placement of malicious layers was randomized and positioned at different depths within the model architecture across various datasets. This design choice was made to emulate real-world scenarios where malicious behavior may not be localized to specific regions of the model but instead distributed unpredictably. The placement strategy ensured that the model’s overall accuracy was maintained while introducing the desired anomalies. Figure 3 illustrates the results of this setup, showing how DRArmor performed in detecting the malicious layers. Unlike the earlier experiments, where malicious layers were consistently positioned at the beginning of the architecture, this randomized placement posed additional challenges for interpretability techniques. Despite these challenges, the methods were able to detect the malicious layers effectively across all datasets, as indicated by the elevated gradients for the malicious layers in Figure 3. However, compared to the previous setup, the difference in gradient values between malicious and non-malicious layers was

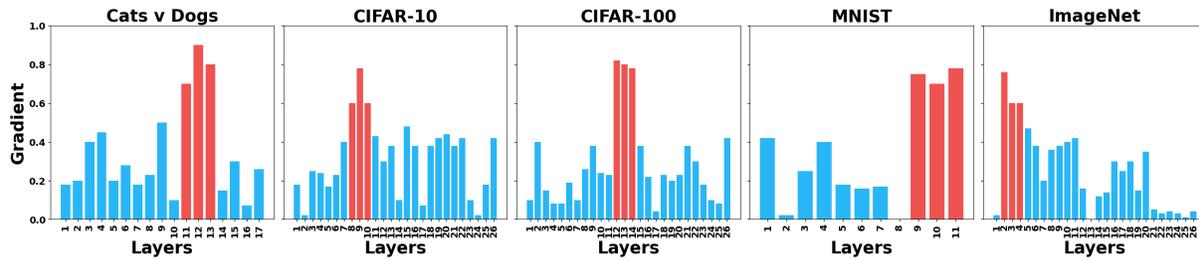


Figure 3: Gradient Analysis: Identifying Malicious Layers Further in the Model by DRArmor (Batch Size 128) with threshold value of 0.5.

notably smaller. This reduced distinction can be attributed to the following factors:

- **Increased Depth and Dilution of Relevance:** When malicious layers are positioned deeper in the model, the relevance propagation process becomes less distinct due to the accumulation and interaction of gradients from preceding layers. This results in a partial dilution of the relevance scores.
- **Randomized Placement Complexity:** The random placement of malicious layers creates additional variability in the feature maps processed by the model. This variability can mask the anomalous behavior of the malicious layers, reducing the sharpness of their detection.
- **Dependency on Layer Contributions:** As layers further in the model are typically more specialized to work on the learnt features of the previous layers, the ability to isolate gradients depends heavily on how well the interpretability method accounts for these contributions. This can result in smaller differences in gradient values between layers.

Despite these reduced distinctions, DRArmor still performs well in detecting malicious layers. As seen in Figure 3, the relevance scores for these layers remain consistently higher than those for non-malicious layers. This indicates that, even with increased separation, the techniques retain their capacity to highlight anomalous behaviour within the model.

6.2 Analyzing Accuracy of Detecting Malicious Layers

Tables 3 and 4 present the evaluation metrics for the accuracy of the defense mechanism in detecting malicious layers under two scenarios depending on their placement: *the start of the model* and *deeper in the architecture*. The evaluation metrics offer insights into the effectiveness of DRArmor in correctly identifying malicious and benign layers.

Performance Across Different Datasets. For all datasets, the TPR and TNR values are relatively high, indicating that the defense mechanism effectively detects most malicious layers and correctly classifies benign ones. MNIST consistently demonstrates the highest TPR and TNR values in both scenarios, with TPR values of 0.980 (start) and 0.906 (middle), and TNR values of 0.989 (start) and 0.941 (middle). This suggests that the simpler nature of MNIST data facilitates more accurate detection of malicious layers.

ImageNet, by contrast, shows lower TPR and TNR values in both cases. For layers at the start, the TPR is 0.746, and the TNR is 0.802. For layers deeper in the model, these drop further to 0.746 and 0.684, respectively. This reduction can be attributed to the increased complexity and diversity of ImageNet, making it more challenging to effectively distinguish malicious from benign layers.

Effect of Layer Positioning. When malicious layers are positioned at the start of the architecture, the detection mechanism achieves higher TPR and TNR values across most datasets compared to when the layers are deeper in the model. For instance, CIFAR-100 has a TPR of 0.926 and TNR of 0.910 at the start, while these decrease to 0.889 and 0.927, respectively, when the layers are further into the architecture.

This trend suggests that malicious layers located at the start of the architecture are easier to detect. Layers closer to the input are often more general in their feature extraction, making anomalies introduced by malicious layers more apparent. In contrast, layers deeper in the architecture learn more specialized presentations, making it harder to identify malicious activity.

False Positive and False Negative Rates. FPR and FNR remain reassuringly small even with three back-door layers. As shown in Table 3, at the start of the model it misclassifies ≈ 0.11 benign and misses 0.06 malicious layers on MNIST, and 4.6 and 0.76 layers, respectively, on ImageNet’s 26-layer network. When the attacker buries the payload deep in the architecture, as shown in Table 4, the worst-case rates on ImageNet rise to FPR = 0.316 and FNR = 0.311, i.e. ≈ 7 false alarms among 23 benign layers and < 1 undetected back-door (0.93 of 3). All other datasets keep both FPR and FNR below 0.20, so at least 80% of benign layers stay untouched while $\geq 74\%$ of malicious insertions are caught. Thus, DRArmor still misses *fewer than one* malicious layer per run on every benchmark, perturbing only a small fraction of the model and maintaining protection. Table 5 confirms that the residual errors have limited consequences: the largest accuracy drop is 4.8 pp (ImageNet, deep), and the highest SSIM obtained by an attacker is 0.39, well below the recognisability threshold of 0.45–0.50 as reported in prior inversion work.

To summarise, DRArmor demonstrates robust performance across all datasets, particularly when malicious layers are located at the start of the model architecture. DRArmor balances high TPR and TNR values and low FPR and FNR values, ensuring reliable detection with minimal impact on benign layers.



Figure 4: Data Reconstructed at the Server Using DP-Gaussian noise with $\sigma^2 = 0.2$ after Identification of the Malicious Layers.

6.3 Analysis of DRArmor Defense

Two strategies may be adopted to mitigate the impact of malicious layers in a model: *adding noise to the gradients* or *completely pruning the malicious layers*. Both approaches aim to protect client data from reconstruction attacks while maintaining the performance of the FL system to the greatest possible extent.

Impact of Adding Gaussian Noise. Figure 12 illustrates the impact of applying low Gaussian noise on the gradients. This method perturbs the gradient values, introducing uncertainty that disrupts the ability of malicious layers to perform data reconstruction. Although the added noise significantly reduces the utility of the gradients for machine learning models attempting to interpret them, it does not fully obscure the information from human perception. As shown in the figure, objects in the reconstructed image remain visually identifiable to human observers despite the noise. This highlights a limitation of noise-based methods, as attackers with advanced techniques might still exploit residual patterns to reconstruct sensitive data.

Impact of Adding Pixelated Noise. Figure 13 demonstrates the effect of pixelation on gradients. Pixelation transforms the gradient values into coarse, block-like representations by averaging groups of values within fixed-size blocks. This process effectively removes fine-grained details crucial for reconstruction, replacing them with large, uniform regions. Unlike Gaussian noise, pixelation makes the reconstructed images appear as unrecognizable mosaics of color blocks. While the overall color distribution may be preserved, the structural and spatial details necessary for accurate reconstruction are completely lost. This makes pixelation a stronger defense mechanism against gradient-based DRA, as attackers are unable to retrieve meaningful representations of the original data.

Impact of Pruning. The impact of pruning malicious layers is summarized in Tables 1 and 2. Pruning removes entirely the

Table 1: FL Task Accuracy Analysis: Comparison of Baseline, DRArmor with Noise & Pixelation (NP), and DRArmor with Pruning at the Start of the Model.

Dataset	FL Acc.	DRArmor (NP)	DRArmor (Prune)
MNIST	0.924	0.874	0.857
CIFAR-10	0.902	0.868	0.823
CIFAR-100	0.894	0.843	0.796
ImageNet	0.873	0.791	0.714
Cats v Dogs	0.981	0.923	0.872

identified malicious layers, eliminating their ability to contribute to DRAs. This approach balances accuracy and privacy, achieving 86% accuracy for MNIST and 80% for CIFAR-100, even with enhanced privacy measures. While there is a slight performance trade-off compared to noise addition, it leaves no space for data leakage as discussed in §6.6.

Impact of False Positives and False Negatives on Defense. Tables 3–5 show that DRArmor’s false alarms (FP) and misses (FN) have tightly bounded consequences. A missed back-door reveals at most the gradients of that single layer. At the same time, a spurious alarm adds controlled noise or prunes a benign layer, producing a recoverable dip in accuracy. The server handles both cases gracefully: updates with the correct shape are accepted, and any malformed update is quietly discarded, preventing knock-on effects. As the tables indicate, occasional misclassifications introduce some information exposure and accuracy loss, but these effects stay within the modest bounds detailed in the tables and do not materially compromise overall privacy or utility.

6.4 Impact of DRArmor on FL Accuracy

Eliminating malicious layers (§6.3) is designed to have minimal impact on the model’s overall accuracy, as these layers contribute

Table 2: FL Task Accuracy Analysis: Comparison of Baseline, DRArmor with Noise & Pixelation (NP), and DRArmor with Pruning Further in the Model.

Dataset	FL Acc.	DRArmor (NP)	DRArmor (Prune)
MNIST	0.924	0.824	0.813
CIFAR-10	0.902	0.829	0.789
CIFAR-100	0.894	0.794	0.755
ImageNet	0.873	0.767	0.708
Cats v Dogs	0.981	0.870	0.853

Table 3: Detection reliability when three malicious layers are inserted at the start of the model.

Dataset	B	M	TPR	TNR	FPR	FNR	#FP	#FN
MNIST	8	3	0.980	0.989	0.011	0.020	0.09	0.06
CIFAR-10	23	3	0.913	0.924	0.076	0.087	1.75	0.26
CIFAR-100	23	3	0.926	0.910	0.090	0.074	2.07	0.22
ImageNet	23	3	0.746	0.802	0.198	0.254	4.55	0.76
Cats v Dogs	14	3	0.842	0.859	0.141	0.158	1.97	0.47



Figure 5: Illustration of Reconstruction Results using DRArmor with Pixelation.

little to the overall output of the model. The aim is to evaluate how defense mechanisms, such as adding noise or pruning, impact the model’s accuracy. The result for accuracy is shown in Tables 1 and 2 for the scenarios, respectively. The first column displays the baseline accuracy of the model without any malicious layers. This represents the optimal accuracy achieved when the model is fully functional and unaffected by adversarial layers. The second column shows the model’s accuracy after applying the defense mechanism by adding noise to the gradients sent by the client. The third column reflects the accuracy after pruning the malicious layers. It is observed that the model can predict the output with an accuracy of 79 to 92% for the data sets used in the experiment.

Moreover, Figures 6 and 7 illustrate the progression of accuracy and loss during training under two conditions: (1) when the model contains only benign layers, and (2) when malicious layers are detected, and defense mechanisms are applied. The initial trajectories of accuracy and loss differ between the two cases of layer positioning, reflecting the disruption caused by malicious layers. However, as training progresses and the defense mechanisms are applied, the metrics gradually converge. For the MNIST dataset, this common convergence point reaches approximately 87%, demonstrating that the defense mechanisms successfully restore the model’s performance to a level comparable to that achieved without malicious layers.

Table 4: Detection reliability when three malicious layers are inserted deep in the model.

Dataset	B	M	TPR	TNR	FPR	FNR	#FP	#FN
MNIST	8	3	0.906	0.941	0.059	0.094	0.47	0.28
CIFAR-10	23	3	0.894	0.910	0.090	0.106	2.07	0.32
CIFAR-100	23	3	0.889	0.927	0.073	0.111	1.68	0.33
ImageNet	23	3	0.746	0.684	0.316	0.311	7.27	0.93
Cats v Dogs	14	3	0.800	0.814	0.186	0.200	2.60	0.60

Table 5: Effect of misclassifications on utility and privacy when three malicious layers are injected per run. Utility Δ = drop in validation accuracy caused by false-positive (FP) perturbations. Privacy Δ = SSIM of reconstructions enabled by a false negative (FN); higher means more leakage.

Dataset	Start		Deep	
	Utility Δ (%)	Privacy Δ (SSIM)	Utility Δ (%)	Privacy Δ (SSIM)
MNIST	-0.3	0.09	-0.6	0.13
CIFAR-10	-0.8	0.14	-1.4	0.20
CIFAR-100	-1.1	0.16	-2.0	0.24
ImageNet	-2.6	0.27	-4.8	0.39
Cats v Dogs	-1.4	0.20	-2.6	0.26

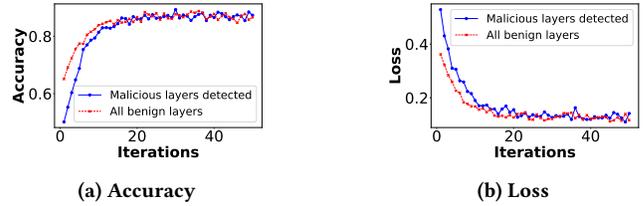


Figure 6: FL Task Accuracy and Loss Comparison of the MNIST Aggregated Model: Without Malicious Layers vs. With Detected and Mitigated Malicious Layers.

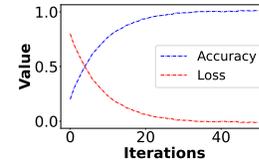


Figure 7: FL Task Accuracy and Loss Comparison of the Aggregated Model on MNIST.

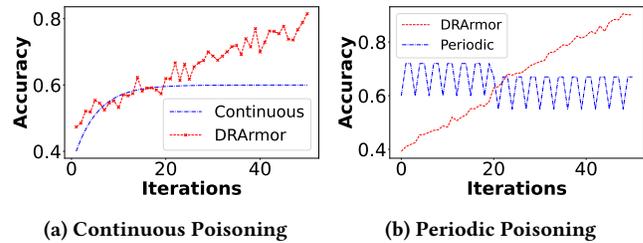


Figure 8: Comparison of Poisoning Impact on Accuracy for Continuous and Periodic Poisoning vs. Malicious Layers Mitigated.

While the defense mechanisms improve accuracy, they are not perfect due to the non-zero chance of false positives in the detection mechanism. Occasionally, the system may mistakenly identify a non-malicious layer as malicious. When this occurs, either noise is added to, or pruning is applied to a legitimate layer, which can impact the model’s performance. As seen in Tables 3 and 4, these cases are infrequent and contribute only marginally to accuracy reduction.

6.5 Impact of Poisoning on Accuracy

As an extension of the experimentation, we examined the impact of continuous and periodic poisoning on the accuracy of DRArmor. This metric is not available in related solutions such as [46] and [20].

In an FL setting, an adversary can inject malicious layers into the model either continuously throughout the training process or during specific, random rounds of training. These scenarios mimic real-world attack strategies where malicious behavior may be persistent or sporadic to evade detection. To assess the robustness of our approach, the client nodes were evaluated for their ability to detect malicious layers, regardless of whether the injection occurred continuously or periodically. The goal was to determine whether the type and timing of poisoning affected the model’s ability to identify and mitigate malicious activity. Figure 8 illustrates the relationship between the poisoning strategy and the accuracy of DRArmor. The graph contains two lines: the poisoning line, representing the frequency and pattern of malicious layer injections (continuous or periodic); and the accuracy line, showing the model’s accuracy over time during training. The figure shows that the accuracy of DRArmor remains unaffected by the type of poisoning strategy. With the proposed detection and mitigation mechanism, the accuracy begins to increase until it stabilises steadily at a consistent level.

6.6 Impact of DRArmor on Leakage Rate

The leakage rate is primarily influenced by two factors: *the size of the local dataset* and *the size of the FC layers*. Our experiment aimed to identify malicious layers and mitigate leakage from these layers using two approaches: noise addition and pruning. Figure 9 shows the effect of the dataset size on the leakage rate.

When the malicious layers are pruned, the system ensures that no gradients from these layers contribute to the updates. This effectively reduces the leakage rate to 0%, as no information from the malicious layers is propagated. When noise is added to the gradients of malicious layers, the leakage rate can vary depending on the magnitude of the noise applied. A higher noise level generally results in a lower leakage rate, but some information may still be leaked if the noise is insufficient. The system’s effectiveness in detecting malicious layers directly influences the leakage rate. The leakage rate is significantly reduced if the malicious layers are correctly identified. If detection is inaccurate (false negatives), leakage may occur, as malicious layers remain unmitigated. With an accuracy range of 72%–87%, some leakage is observed, particularly as the local dataset size increases. However, this leakage is minimal compared to the baseline leakage rate observed in the LoKI framework. The experiment confirms that DRArmor reduces the leakage rate to a minimal level.

7 Discussion

Comparison with Other Defense Algorithms Previous research (§2) focuses on mitigating the effects of gradients sent by malicious models to the server, which the DRA exploits. Although these methods attempt to obscure or alter the gradients to prevent reconstruction, they fail to address the root cause of these attacks. DRArmor takes a fundamentally different direction by identifying the malicious layers and model parameters responsible for leaking

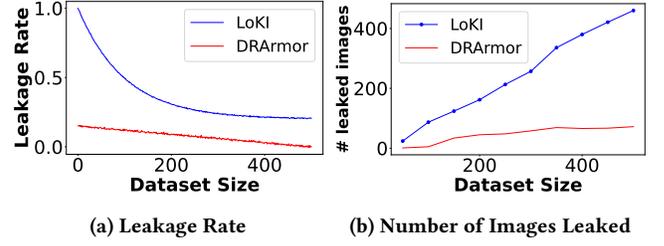


Figure 9: Leakage Rate and Number of Leaked Images as a Function of the Local Dataset Size Averaged over 200 Clients.

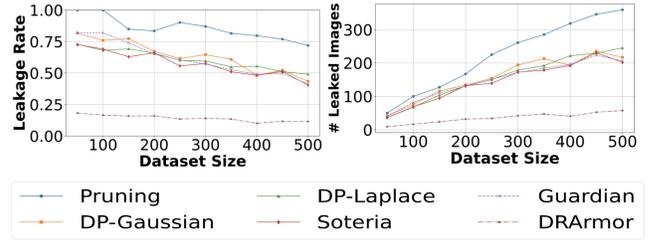


Figure 10: Leakage Rate and Number of Leaked Images as a Function of the Local Dataset Size Averaged over 200 Clients against Other Defenses.

Table 6: Comparison of DRArmor with existing defenses on CIFAR-100

Defense Method	Privacy	Utility
No Defense	74.8% / 312 ± 7.4	92.5 ± 0.4
DP-Gaussian	19.5% / 68 ± 3.4	82.1 ± 0.6
DP-Laplace	26.4% / 92 ± 4.1	80.3 ± 0.7
Soteria	41.8% / 154 ± 6.0	82.4 ± 0.5
DRArmor	12.2% / 44 ± 2.3	86.7 ± 0.5

gradients to reconstruct the client device data. DRArmor prevents gradient leakage and ensures that malicious layers causing leakage are effectively detected and eliminated (as shown in §6.3).

The new approach in DRArmor makes directly comparing with related work challenging. Inspired by [55], to evaluate its effectiveness, we compare its performance against existing defences based on leakage rate and the number of images leaked per dataset size. As shown in Figure 10, DRArmor achieves a significantly lower leakage rate compared to other methods replicated in our evaluations (e.g. [20, 46]).

Moreover, table 6 compares DRArmor with baseline defenses in terms of privacy and utility. Privacy is quantified using the leakage rate and number of reconstructed images, while utility is measured by test accuracy while defending against the DRA. DRArmor achieves the lowest privacy leakage with minimal accuracy degradation, outperforming methods like DP-SGD and Soteria, which apply global obfuscation without identifying the malicious layers responsible for leakage.

Another notable strength of our approach is its robustness to changes in attack architecture. Traditional defense mechanisms are often compromised when the attack model changes, but our method remains effective due to its use of Explainable AI. As shown in Figure 3, DRArmor can identify malicious layers even when the model architecture changes, providing flexibility and resilience not feasible in related work.

The threat model used in DRArmor is based on a real-world scenario where an attacker can inject malicious models at any iteration, where the client must detect the attack as soon as it occurs. We evaluated DRArmor with continuous and periodic poisoning attacks to test this, achieving consistent accuracy in both cases §6.5. We also assessed the scalability of DRArmor by testing it across multiple clients. Unlike the studies in the attacks[21, 55], which used 100 clients, our experiments scale up to 200 clients to evaluate the impact of these attacks on model accuracy in a larger setting.

Limitations of DRArmor: While our method is the first to detect DRAs in FL, it currently relies on a limited set of XAI techniques for identifying malicious layers. Although we demonstrate the effectiveness of DRArmor across both small and large models, distinguishing between malicious and non-malicious layers can become less precise as model complexity increases. To improve robustness and interpretability, more advanced or domain-adapted attribution methods (e.g.,[41, 43]) may offer better explanatory power. DRArmor is designed to be modular, so such techniques can be substituted into our framework depending on deployment needs. A discussion on the computational practicality of DRArmor’s components, including their applicability to typical federated clients, is provided in §4 and Appendix A. We leave the exploration of alternative XAI methods and extended scalability analysis as directions for future work.

Lastly, while DRArmor’s detection mechanism reduces data leakage, it also leads to a slight decrease in the model’s accuracy compared to the original model. However, considering our primary objective is preserving privacy in FL systems, we argue this trade-off is acceptable given significantly improved privacy.

8 Conclusion

We present DRArmor, a robust defense mechanism designed to detect and mitigate Data Reconstruction Attacks in FL. To the best of our knowledge, DRArmor is the first solution that leverages Explainable AI to effectively analyze how individual layers of a model contribute to its output. By identifying layers that are not learning relevant features but still send disproportionately high gradients, DRArmor can accurately classify these layers as malicious. Once detected, DRArmor employs techniques such as noise injection, pixelation, and pruning to neutralize the impact of these malicious layers, safeguarding client privacy without compromising model integrity. Unlike the studies in [55] and [21], which used 100 clients, we replicated these attacks with 200 clients to evaluate the effectiveness of DRArmor in larger settings. Our results demonstrate that DRArmor significantly reduces data leakage, even as the number of client nodes and the size of the dataset increase. Compared to existing defense algorithms, DRArmor achieves an average accuracy rate of 87%, confirming the efficacy of our proposed solution.

References

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
- [2] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. *Advances in Neural Information Processing Systems* 31 (2018).
- [3] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. 2021. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems* 34 (2021), 17455–17466.
- [4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS one* 10, 7 (2015), e0130140.
- [5] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems* 35 (2022), 7641–7654.
- [6] Gilles Barthe and Boris Köpf. 2011. Information-theoretic bounds for differentially private mechanisms. In *2011 IEEE 24th Computer Security Foundations Symposium*. IEEE, 191–204.
- [7] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchu Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).
- [8] Franziska Boenisch, Adam Dziedzic, Roi Schuster, Ali Shahin Shamsabadi, Iliia Shumailov, and Nicolas Papernot. 2023. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 175–199.
- [9] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2024. Automatic clipping: Differentially private deep learning made easier and stronger. *Advances in Neural Information Processing Systems* 36 (2024).
- [10] Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. 2024. When federated learning meets privacy-preserving computation. *Comput. Surveys* 56, 12 (2024), 1–36.
- [11] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International conference on machine learning*. PMLR, 1964–1974.
- [12] Graham Cormode. 2011. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1253–1261.
- [13] Paul Cuff and Lanqing Yu. 2016. Differential privacy as a mutual information constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 43–54.
- [14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [15] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
- [16] Travis Dick, Cynthia Dwork, Michael Kearns, Terrance Liu, Aaron Roth, Giuseppe Vietri, and Zhiwei Steven Wu. 2023. Confidence-ranked reconstruction of census microdata from published statistics. *Proceedings of the National Academy of Sciences* 120, 8 (2023), e2218605120.
- [17] Dimitar Iliev Dimitrov, Mislav Balunovic, Nikola Konstantinov, and Martin Vechev. 2022. Data leakage in federated averaging. *Transactions on Machine Learning Research* (2022).
- [18] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.
- [19] Ahmed El Ouaqrhi and Ahmed Abdelhadi. 2022. Differential privacy for deep and federated learning: A survey. *IEEE access* 10 (2022), 22359–22380.
- [20] Mingyuan Fan, Yang Liu, Cen Chen, Chengyu Wang, Minghui Qiu, and Wenmeng Zhou. 2024. Guardian: Guarding against Gradient Leakage with Provable Defense for Federated Learning. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 190–198.
- [21] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. 2021. Robbing the fed: Directly obtaining private data in federated learning with modified models. *ICLR 2022* (2021).
- [22] Awni Hannun, Chuan Guo, and Laurens van der Maaten. 2021. Measuring data leakage in machine-learning models with fisher information. In *Uncertainty in Artificial Intelligence*. PMLR, 760–770.
- [23] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 603–618.
- [24] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security)*

- 19). 1895–1912.
- [25] Sanjay Kariyappa, Chuan Guo, Kiwan Maeng, Wenjie Xiong, G Edward Suh, Moinuddin K Qureshi, and Hsien-Hsin S Lee. 2023. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In *International Conference on Machine Learning*. PMLR, 15884–15899.
- [26] Jong Wook Kim, Kennedy Edemacu, Jong Seon Kim, Yon Dohn Chung, and Beakcheol Jang. 2021. A survey of differential privacy-based techniques and their applicability to location-based services. *Computers & Security* 111 (2021), 102464.
- [27] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [28] Daniel Levy, Ziteng Sun, Kareem Amin, Satyen Kale, Alex Kulesza, Mehryar Mohri, and Ananda Theertha Suresh. 2021. Learning with user-level privacy. *Advances in Neural Information Processing Systems* 34 (2021), 12466–12479.
- [29] Ang Li, Yixiao Duan, Huanrui Yang, Yiran Chen, and Jianlei Yang. 2020. TIPRDC: task-independent privacy-respecting data crowdsourcing framework for deep learning with anonymized intermediate representations. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 824–832.
- [30] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 181–192.
- [31] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
- [32] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.
- [33] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.
- [34] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. 2017. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern recognition* 65 (2017), 211–222.
- [35] Virraji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. 2021. Federated-learning-based anomaly detection for IoT security attacks. *IEEE Internet of Things Journal* 9, 4 (2021), 2545–2554.
- [36] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 739–753.
- [37] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R Rabiee. 2018. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering* 32, 1 (2018), 54–66.
- [38] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3498–3505.
- [39] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. 2022. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2429–2443.
- [40] Zhongang Qi, Saeed Khorram, and Fuxin Li. 2019. Visualizing Deep Networks by Optimizing with Integrated Gradients. In *CVPR workshops*, Vol. 2. 1–4.
- [41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “Why should I trust you?” Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.
- [42] Nuria Rodríguez-Barroso, Daniel Jiménez-López, M Victoria Luzón, Francisco Herrera, and Eugenio Martínez-Cámara. 2023. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. *Information Fusion* 90 (2023), 148–173.
- [43] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 618–626. <https://doi.org/10.1109/ICCV.2017.74>
- [44] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.
- [45] Jinhyun So, Başak Güler, and A Salman Avestimehr. 2020. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications* 39, 7 (2020), 2168–2181.
- [46] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. 2021. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9311–9319.
- [47] Qi Tan, Qi Li, Yi Zhao, Zhuotao Liu, Xiaobing Guo, and Ke Xu. 2024. Defending Against Data Reconstruction Attacks in Federated Learning: An Information Theory Approach. *Proceedings of the 33rd USENIX Conference on Security Symposium, 24* (2024).
- [48] Md Palash Uddin, Yong Xiang, Xuequan Lu, John Yearwood, and Longxiang Gao. 2020. Mutual information driven federated learning. *IEEE Transactions on Parallel and Distributed Systems* 32, 7 (2020), 1526–1538.
- [49] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2512–2520.
- [50] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security* 15 (2020), 3454–3469.
- [51] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. 2022. Fishing for user data in large-batch federated learning via gradient magnification. *International Conference on Machine Learning* (2022).
- [52] Tianyu Xia, Shuheng Shen, Su Yao, Xinyi Fu, Ke Xu, Xiaolong Xu, and Xing Fu. 2023. Differentially private learning with per-sample adaptive clipping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 10444–10452.
- [53] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradient inversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 16337–16346.
- [54] Xinwei Zhang, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Jinfeng Yi. 2022. Understanding clipping for federated learning: Convergence and client-level differential privacy. In *International Conference on Machine Learning, ICML 2022*.
- [55] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. 2024. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1287–1305.
- [56] Lingchen Zhao, Jianlin Jiang, Bo Feng, Qian Wang, Chao Shen, and Qi Li. 2021. Sear: Secure and efficient aggregation for byzantine-robust federated learning. *IEEE Transactions on Dependable and Secure Computing* 19, 5 (2021), 3329–3342.

A Impact of DRArmor on Scalability and Computation Overhead

While evaluating DRArmor’s practicality in federated settings, it is essential to acknowledge that the computational overhead and scalability of the defense are inherently influenced by multiple factors—including the model’s depth and architecture, the type of dataset used, batch size, and the nature of layers (e.g., convolutional vs. dense). To provide a realistic assessment, we measured per-iteration wall-clock time across 200 client nodes using the five datasets described in §5, encompassing models of different sizes and complexities. Figure 11 illustrates the iteration runtimes under four configurations: baseline training (no defense), and DRArmor’s attribution routines (DTD or LRP) with batch sizes of 64 and 128. Notably, these explainability routines are invoked only once per local round rather than per mini-batch, thereby constraining the overhead to a single backward pass (for DTD) or a single relevance propagation (for LRP).

Introducing DTD incurs an overhead of approximately 15–25 % relative to baseline, which can be accommodated within the typical idle windows of federated clients. In contrast, LRP induces a higher latency on deeper architectures—for example, on ImageNet it approaches 1.3 s per iteration. The small fluctuations in each curve capture realistic runtime variability (I/O jitter, network latency, etc.). The expected slowdown when halving the batch size from 128 to 64, on the order of 10–20 %—is also evident across all methods.

Notably, the near-linear increase in runtime from MNIST’s lightweight model to ImageNet’s deep network demonstrates that DRArmor’s one-time, post-training analysis scales proportionally with model complexity of varying batch sizes. These results confirm

Estimated Runtime Overhead per Iteration: Baseline vs. DTD vs. LRP

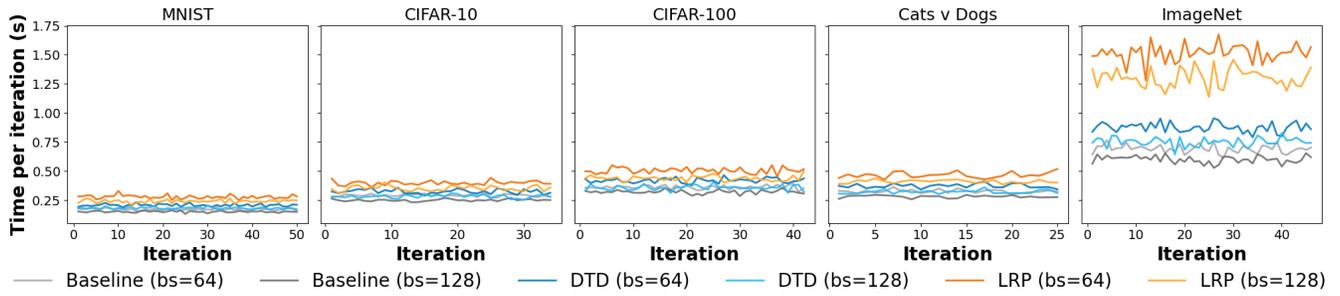


Figure 11: Estimated per-iteration runtime overhead: baseline training vs. DRArmor with DTD and LRP, at batch sizes 64 and 128, across models of varying complexity.

that the default DTD-based detection is efficient and scalable, requiring no specialized hardware, while allowing practitioners to

substitute alternative XAI techniques when tighter latency budgets are needed.



Figure 12: Data reconstructed at the server using DP-Gaussian Noise with $\sigma^2=0.2$ after identification of the malicious layers

B Sample reconstructed images for other datasets

In §6, we presented a sample reconstruction example demonstrating how DRArmor mitigates DRA. Here, we extend that analysis to a broader set of examples across the dataset, providing a more comprehensive evaluation of our defense mechanism.

Figure 12 showcases a reconstruction example where the server attempts to recover input data after applying DP Gaussian noise with $\sigma^2=0.2$, following the identification of malicious layers. This image demonstrates how noise-injected gradients can partially obscure sensitive information while retaining some recognizable structure.

Figure 13 provides a broader set of reconstruction results. Each group of images consists of three components: the original image from the dataset (first), the reconstruction generated by the DRA attack using the LOKI method (second), and the result produced when pixelated gradients from malicious layers are intercepted and modified by our defense mechanism (third). These comparisons highlight the degradation in reconstruction quality achieved through DRArmor, illustrating its effectiveness in disrupting high-fidelity data recovery by adversaries.

These examples underscore the robustness of DRArmor in protecting user data in FL settings. By detecting compromised layers and modifying their outputs, our method significantly impairs adversarial reconstructions while maintaining the integrity of the learning process.



Figure 13: Illustration of reconstruction results across multiple examples. Each group contains three images: the original image (first), the reconstruction produced by the DRA attack using LoKI (second), and the reconstruction generated when pixelated gradients are sent from malicious layers (third) after our detection mechanism.