Cutting Through Privacy: A Hyperplane-Based Data Reconstruction Attack in Federated Learning

Francesco Diana[1,2] **André Nusser**[1,2,3,4] **Chuan Xu**[1,2,3,4] **Giovanni Neglia**[1,2]

[1]Université Côte d'Azur , France
[2]Inria, , France
[3]CNRS, France
[4]I3S, France

Abstract

Federated Learning (FL) enables collaborative training of machine learning models across distributed clients without sharing raw data, ostensibly preserving data privacy. Nevertheless, recent studies have revealed critical vulnerabilities in FL, showing that a malicious central server can manipulate model updates to reconstruct clients' private training data. Existing data reconstruction attacks have important limitations: they often rely on assumptions about the clients' data distribution or their efficiency significantly degrades when batch sizes exceed just a few tens of samples.
In this work, we introduce a novel data reconstruction attack that overcomes these limitations. Our method leverages a new geometric perspective on fully connected layers to craft malicious model parameters, enabling the perfect recovery of arbitrarily large data batches in classification tasks without any prior knowledge of clients' data. Through extensive experiments on both image and tabular datasets, we demonstrate that our attack outperforms existing methods and achieves perfect reconstruction of data batches two orders of magnitude larger than the state of the art.

1 INTRODUCTION

Federated Learning (FL) enables collaborative machine learning model training across distributed clients without directly sharing data with a central server [McMahan et al., 2017]. Although not sharing data ostensibly guarantees privacy, several attacks have revealed existing privacy vulnerabilities in FL. By inspecting the updates received and using publicly available information, the central server can infer training data membership [Melis et al., 2019, Nasr et al., 2019, Choquette-Choo et al., 2021] or infer clients' sensitive attributes [Chen et al., 2024, Diana et al., 2025]. These attacks only expose limited information, whereas more advanced data reconstruction attacks enable the server to fully recover private training data points. These attacks have been proposed in two primary settings: the *honest-but-curious* setting, where the server passively observes updates [Zhu et al., 2019, Zhao et al., 2020, Geiping et al., 2020, Yin et al., 2021, Kariyappa et al., 2023, Dimitrov et al., 2022, 2024], and the *malicious* setting [Fowl et al., 2022, Boenisch et al., 2023, Zhang et al., 2023, Zhao et al., 2024, Wang et al., 2024, Garov et al., 2024], where the server actively manipulates model parameters. In the honest-but-curious setting, existing methods struggle to perfectly recover the training data for batch sizes of a few dozen data points [Dimitrov et al., 2024]) or produce very noisy reconstructions for a large portion of data points in batches exceeding 100 data points [Kariyappa et al., 2023, Fig. 6]. In the malicious setting, existing attacks [Wen et al., 2022, Fowl et al., 2022, Boenisch et al., 2023, Zhang et al., 2023, Garov et al., 2024] perform better but have not succeeded in perfectly recovering a batch of 256 data points.

Furthermore, most previous work has focused on image datasets, even though many real-world applications involve tabular data—such as clinical and financial records—which often contains highly sensitive information.

In this paper we focus on the *malicious* setting, in which the server is able to craft the model parameters to improve the attack's effectiveness. We present a novel attack that reconstructs client data by combining geometric search in the feature space with precise control over each data point's contribution to the client's updates. The core idea behind our attack is to isolate individual data points within strips defined by parallel hyperplanes and then iteratively reconstruct the entire batch by leveraging the knowledge of per-datapoint gradient contributions.

This approach is effective for all classification tasks, including those involving low-dimensional tabular data where existing state-of-the-art attacks perform poorly, demonstrat-

ing its robustness across diverse data modalities. Concretely, our key contributions are the following:

- We derive an analytical upper bound on the accuracy of existing data reconstruction attacks on random data.
- We show how a malicious server can control each data point's contribution to the client's updates.
- We propose a perfect reconstruction algorithm for classification tasks that is agnostic to input dimensionality. Hence, while most of the existing literature primarily addresses image classification, our attack can reconstruct both image and tabular data.
- Our method enables full recovery of all samples within a batch and remains effective regardless of batch size, provided that the number of training rounds is sufficiently high.
- We validate our findings on both image and tabular datasets, achieving perfect reconstruction on large batches of up to 4,096 data points, where our baseline fails to recover batches of 64 inputs.

2 BACKGROUND

In federated learning, a set of users $\mathcal{U}$, orchestrated by a central server, cooperate to optimize a shared model θ, which minimizes the weighted sum of clients' empirical risks:

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) = \sum_{u \in \mathcal{U}} p^u \mathcal{L}(\theta, \mathcal{D}^u), \tag{1}$$

where $\mathcal{D}^u = \{(\mathbf{x}_i^u, y_i^u)\}_{i=1}^{n^u}$ is the local dataset of client u, $\mathcal{L}^u$ measures the loss of model θ on points in $\mathcal{D}^u$, and p^u is a positive weight quantifying the relative importance of client u, s.t. $\sum_{u \in \mathcal{U}} p^u = 1$. Typical choices in FL are $p^u = 1/|\mathcal{U}|$ and $p^u \propto n^u$.

Federated Learning typically involves iterative algorithms that operate in multiple communication rounds between clients and the central server. At each communication round, clients receive the current global model from the server and transmit updates computed on their local datasets. The nature of the updates depends on the specific federated optimization algorithm in use. For instance, in FedSGD [Chen et al., 2016, McMahan et al., 2017], each client computes and transmits a single gradient update $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^u)$. In contrast, algorithms like FedAvg [McMahan et al., 2017] allow clients to perform multiple local stochastic gradient descent steps before sending their updated model parameters to the server.

While FL preserves data locality, the transmitted updates—whether gradients or model parameters—can leak sensitive information about the clients' datasets [Melis et al., 2019, Diana et al., 2025, Fowl et al., 2022]. This vulnerability exposes FL to adversarial attacks aimed at reconstructing private data [Zhu et al., 2019, Fowl et al., 2022, Boenisch et al., 2023, Dimitrov et al., 2024]. The success rate of such attacks depends on the adversary's capabilities. In what follows, we consider the attacker to be the server.

Two adversarial settings are commonly considered in the literature: the *honest-but-curious* setting [Zhu et al., 2019, Dimitrov et al., 2024] and the *malicious* setting [Fowl et al., 2022, Boenisch et al., 2023]. In the honest-but-curious setting, the server passively observes updates without interfering with the training process [Paverd and Martin, 2014]. In the malicious setting, the server actively manipulates the training process by transmitting modified models designed to induce privacy leakage. In both settings, two main categories of attacks have been proposed: optimization-based attacks and analytical attacks.

2.1 OPTIMIZATION-BASED ATTACKS

In optimization-based attacks, the server attempts to reconstruct the client's data by minimizing the difference between the actual update sent by the client to the server (e.g., the gradient or updated model) and the update that would have been computed by the client from the reconstructed data. These attacks often begin with dummy inputs, which are progressively refined through iterative steps.

Early works [Zhu et al., 2019, Zhao et al., 2020, Geiping et al., 2020, Yin et al., 2021, Dimitrov et al., 2022] proposed the recovery of image inputs within the honest-but-curious setting, but were successful only on small (≤ 50) batch sizes and, even then, produced noisy reconstructions. Kariyappa et al. [2023] reformulated the reconstruction task as a blind source separation problem, enabling recovery in presence of secure aggregation [Bonawitz et al., 2017], however, they still only obtained very noisy images on batches larger than 200 data points (see Figure 6 in their work).

Vero et al. [2023] introduced the first optimization-based attack targeting tabular data and reported a 70% reconstruction success rate on a batch of 128 samples.

Overall, these attacks struggle with the challenge posed by gradient aggregation, and do not scale effectively to large batch sizes without requiring auxiliary data. To address these shortcomings, alternative approaches consider a scenario where the server is malicious. In this context, Wen et al. [2022] proposed a *fishing* strategy that alters the weights of the classification layer of a neural network to amplify the gradient contribution of a single input within a batch, increasing the success rate of the optimization step. However, their approach relies on estimating the distribution of one input feature, which the authors assume to follow a normal distribution. While this assumption may hold true for image data, it is often unrealistic for tabular datasets, where features can be binary or categorical. Alternatively, Garov et al. [2024] designed an encoder-decoder architecture to isolate individual input gradients from the aggregated update in a

latent space using an encoder to reduce the detectability, and subsequently reconstruct the original input with a decoder. Nonetheless, their method depends on access to auxiliary data for training the encoder-decoder network and struggles to achieve high-quality recovery for approximately half of the images in a batch of 512 samples.

2.2 ANALYTICAL ATTACKS

Unlike optimization-based attacks, analytical methods aim to exactly recover input data by exploiting the properties of fully connected layers in neural networks. Prior works [Phong et al., 2017, Geiping et al., 2020] have shown that the input to a biased linear layer followed by a ReLU activation function can be exactly reconstructed from its gradients. In what follows, we describe this technique, which forms the core of our attack.

For simplicity, consider a client training a model where the first layer is a fully connected (FC) layer with input vector $\mathbf{x} \in \mathbb{R}^d$ and output vector $\mathbf{z} \in \mathbb{R}^N$. Let $\mathbf{W} \in \mathbb{R}^{N \times d}$ and $\mathbf{b} \in \mathbb{R}^N$ denote the weight matrix and the bias vector, respectively. Let the output of the i-th neuron in the layer be $z_i = \text{ReLU}(\mathbf{W}_i\mathbf{x}+b_i)$. Given a sample $(\mathbf{x}_j, y_j)$, let $\mathcal{L}_j$ and $z_{i,j}$ denote the loss and the output of neuron i for this sample, respectively. The gradient of the loss with respect to $\mathbf{W}_i$ and b_i can be computed as follows:[1] $\frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_i} = \frac{\partial \mathcal{L}_j}{\partial z_i}\frac{\partial z_i}{\partial \mathbf{W}_i} = \frac{\partial \mathcal{L}_j}{\partial z_i}\mathbf{x}_j \mathbb{1}_{z_{i,j}>0}$ and $\frac{\partial \mathcal{L}_j}{\partial b_i} = \frac{\partial \mathcal{L}_j}{\partial z_i}\frac{\partial z_i}{\partial b_i} = \frac{\partial \mathcal{L}_j}{\partial z_i}\mathbb{1}_{z_{i,j}>0}$, from which it follows that $\frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_i} = \frac{\partial \mathcal{L}_j}{\partial b_i}\mathbf{x}_j$. Consequently, as long as $\frac{\partial \mathcal{L}_j}{\partial b_i} \neq 0$ (which requires that $\mathbf{x}_j$ activates neuron i, i.e., $\mathbf{W}_i\mathbf{x}_j + b_i > 0$), knowing the gradient of the model loss enables exact reconstruction of the input:

$$\mathbf{x}_j = \frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_i}\left(\frac{\partial \mathcal{L}_j}{\partial b_i}\right)^{-1}. \tag{2}$$

However, in practice, gradients are computed on a batch $\mathcal{B} = \{(\mathbf{x}_j, y_j)\}_{j=1}^n$ of size n, rather than on a single input. The attacker can then only observe the gradient of the average loss $\mathcal{L} = \frac{1}{n}\sum_{j=1}^n \mathcal{L}_j$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} = \frac{1}{n}\sum_{j=1}^n \frac{\partial \mathcal{L}_j}{\partial b_i}\mathbf{x}_j \tag{3}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \frac{1}{n}\sum_{j=1}^n \frac{\partial \mathcal{L}_j}{\partial b_i}. \tag{4}$$

If $\frac{\partial \mathcal{L}}{\partial b_i} \neq 0$—implying that at least one input activates neuron i—the attacker can reconstruct a combination $\mathbf{g}_i$ of the inputs from this gradient [Zhang et al., 2023, Theorem 1]:

$$\mathbf{g}_i = \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} \cdot \left(\frac{\partial \mathcal{L}}{\partial b_i}\right)^{-1} = \sum_{j=1}^n \alpha_j \mathbf{x}_j, \tag{5}$$

$$\alpha_j = \frac{\frac{\partial \mathcal{L}_j}{\partial b_i}}{\sum_{k=1}^n \frac{\partial \mathcal{L}_k}{\partial b_i}}. \tag{6}$$

Equations 5 and 6 are the starting points for different attacks.

2.2.1 Sparsity-Based Attacks

Several works, commonly referred to as *sparsity-based* attacks, exploit the sparseness induced by the ReLU activation to recover individual data points. In the honest-but-curious setting, Dimitrov et al. [2024] proposed a method that leverages the low-rank structure of gradients and exploits ReLU-induced sparsity to efficiently guide a greedy search that enables the recovery of the inputs. Their attack achieves perfect recovery for batches with ≤ 25 data points but fails to scale to larger batch sizes due to the exponential computational cost w.r.t. the number of samples.

Alternatively, some approaches introduced malicious model modifications to enforce sparsity in the gradients and increase the percentage of single inputs that activate a neuron. Boenisch et al. [2023] presented a *trap weights* method, where the FC layer's weights are randomly initialized, with one half having positive values and the other half larger negative values. The idea behind this approach is to maximize the probability that a single input activates a given neuron and can be reconstructed using (2). Zhang et al. [2023] extended this attack to networks with sigmoid activation, though their method requires additional data to fine-tune the malicious weights effectively. Pasquini et al. [2022] further leveraged ReLU-induced sparsity to bypass secure aggregation [Bonawitz et al., 2017].

As we will demonstrate in Sec. 3, sparsity-based attacks are significantly less effective on low-dimensional data, such as tabular datasets.

2.2.2 Other Analytical Attacks

Some analytical attacks can recover input data without inducing gradient sparsity. Fowl et al. [2022] assume prior knowledge of the distribution of a linear combination of client inputs and configure neuron biases such that the gradients with respect to different neurons' weights differ by only a single input, which can be recovered using (3) and (4). The first round of our parallel search method presented in Sec. 4.4 is similar to their approach but there are two crucial differences: (i) they require prior knowledge of a distribution, while we dynamically discover the appropriate biases, and (ii) they do not propose any iterative procedure that enables a bias search across multiple communication rounds, as we do.

[1] The expression $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^N$, can also be interpreted as the Jacobian of the function $\mathbf{f}$, with the (less common) convention that $\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)_{i,j} = \frac{\partial f_j}{\partial x_i}$.

Zhao et al. [2024] adapted this method to break secure aggregation, designing convolutional filters to isolate individual client's gradient contributions.

2.2.3 Extension to other architectures

Until now, we have discussed reconstruction attacks targeting the first layer in the network. These attacks can be directly extended when the attacked layer is preceded by additional FC layers.

Several works have explored adapting these attacks to convolutional neural networks (CNNs). Boenisch et al. [2023] proposed modifying the weights of convolutional layers to act as identity mappings, allowing the attacker to recover the original input from the reconstructed input of the first FC layer. However, pooling layers and dropout can disrupt the gradient structure, reducing the effectiveness of the recovery process. An alternative approach, introduced by Kariyappa et al. [2023] and Zhang et al. [2023], focuses on reconstructing the feature maps that serve as inputs to the first fully connected layer. Once these feature maps are obtained, the attacker solves an optimization problem to recover the original input. Additionally, Wang et al. [2024] proposed specific parameter modifications to facilitate analytical recovery in CNNs, further broadening the applicability of these attacks.

Other works have specifically targeted transformer-based models. Lu et al. [2022] demonstrated that analytical image recovery is feasible even in attention-based networks; however, their attack is restricted to a batch size of one. Fowl et al. [2023] and Chu et al. [2023] extended their earlier work [Fowl et al., 2022] to extract textual data from clients; however, their methods do not extend to other data modalities, such as images.

3 LIMITATIONS OF EXISTING SPARSITY-BASED ATTACKS

The sparsity-based attacks proposed by Boenisch et al. [2023], Zhang et al. [2023] attempt to modify the parameters of the FC layer so that each neuron in the layer is activated only by a single input. Given the batch $\mathcal{B} = \{(\mathbf{x}_1, y_1) \ldots, (\mathbf{x}_n, y_n)\}$, neuron i is activated only from input $\mathbf{x}_j$ if $\mathbf{W}_i\mathbf{x}_j + b_i > 0$ and $\mathbf{W}_i\mathbf{x}_k + b_i < 0$ for $k \in \{1, \ldots, n\} \setminus \{j\}$. In this section we provide a geometric interpretation of the problem and demonstrate that the accuracy of sparsity-based attacks is fundamentally constrained by the dimensionality of the input they aim to reconstruct.

Consider the set of inputs for which the activation value of neuron i is equal to 0, i.e., $\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{W}_i\mathbf{x} + b_i = 0\}$. This set is a hyperplane in the Euclidean space $\mathbb{R}^d$ orthogonal to the vector $\mathbf{W}_i$. The attacks proposed by Boenisch et al. [2023], Zhang et al. [2023] can be interpreted as identifying hyperplanes orthogonal to random directions $\mathbf{W}_i$ that separate a single input from the rest of the inputs in the batch, which we denote by $\mathcal{B}_\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. Given a gradient computed on the batch $\mathcal{B}$, the maximum number of recoverable points is upper-bounded by the number of points in $\mathcal{B}_\mathbf{x}$ that are linearly separable from all the others, i.e., the number of vertices on the convex hull of $\mathcal{B}_\mathbf{x}$. By bounding the expected number of vertices, we derive the following theorem (proof in App. A):

Theorem 3.1. *Let the input features of the client's local dataset consist of n random points in $\mathbb{R}^d$ that are drawn*

1. *uniformly at random from the unit ball, or*
2. *uniformly at random from the unit hypercube, or*
3. *from a centered normal distribution with covariance matrix $\mathbf{I}_d$.*

Consider training a machine learning model through FedSGD with full-batch updates. The expected number of samples that can be reconstructed by any attack relying on isolating individual inputs from the rest of the batch is:

1. $O(n^{(d-1)/(d+1)})$,
2. $O(\log^{d-1} n)$,
3. $O(\log^{(d-1)/2} n)$,

respectively for the three distributions above.

Theorem 3.1 shows that, for a fixed input dimension d, the expected fraction of reconstructed samples—i.e., the attack success rate—approaches zero asymptotically as the batch size n increases. This result explains the attack performance degradation observed by Boenisch et al. [2023], Zhang et al. [2023] when larger batch sizes are used. Furthermore, the theorem highlights a fundamental limitation in low-dimensional settings: the smaller the input dimension d, the lower the success rate. This indicates that while such attacks can succeed in high-dimensional regimes, their effectiveness diminishes substantially when applied to low-dimensional data. Moreover, high-dimensional data often lies on manifolds of much lower intrinsic dimensionality [Narayanan and Mitter, 2010, Pope et al., 2021, Brown et al., 2023], potentially reducing the success rate of these attacks even further than predicted by Theorem 3.1.

4 OUR ATTACK

In this section, we first present our threat model (Sec. 4.1) and then introduce our attack assuming the attacker initially exploits a single neuron in the linear layer (Sec. 4.2 and 4.3). We subsequently demonstrate how the attacker can leverage all neurons in parallel to significantly accelerate the reconstruction process (Sec. 4.4).

4.1 THREAT MODEL

In this work, we consider a malicious server whose intent is to recover clients' private data by altering model parameters, consistent with the setting described in Wen et al. [2022], Boenisch et al. [2023], Zhang et al. [2023], where clients trust the server and have no control over the model parameters. We note that a third party capable of intercepting communications between the client and server could also act as an attacker.

Additionally, since the central server orchestrates the FL protocol, we assume that it is responsible for selecting the clients participating in each training round. Unlike previous works [Fowl et al., 2022, Wen et al., 2022, Boenisch et al., 2023, Zhang et al., 2023, Zhao et al., 2024], we assume the server has no knowledge of the clients' data distribution and does not possess any auxiliary training data. The information available to the server is limited to some bounds on the range of the data after preprocessing. We suppose that clients follow the FedSGD protocol and at each round compute a full-batch gradient update.

4.2 LOCATE AND RECONSTRUCT THE INPUTS

Consider that the client under attack has dataset $\mathcal{B}$ with corresponding inputs $\mathcal{B}_{\mathbf{x}}$. For simplicity, we assume that the neural network consists solely of a FC input layer with ReLU activations (with parameters $\mathbf{W}^{(1)} \in \mathbb{R}^{N\times d}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^N$) and a FC classification layer (with parameters $\mathbf{W}^{(2)} \in \mathbb{R}^{C\times N}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^C$). The output of the network can be expressed as $\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)} \in \mathbb{R}^C$, where $\mathbf{z}^{(1)} = \text{ReLU}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \in \mathbb{R}^N$. The attack can be extended to other architectures as described in other papers (see discussion in Sec. 2.2.3).

The server selects a random hyperplane by sampling $\mathbf{W}_i^{(1)}$ from an arbitrary random distribution. During different communication rounds, it sends the client under attack a model whose parameters are constant except for the bias value $b_i^{(1)}$. Geometrically, changing $b_i^{(1)}$ corresponds to translating the hyperplane through the input space (see Fig. 1). The goal is to find a sequence of scalars $\hat{b}_{i,0}^{(1)} \leq \cdots \leq \hat{b}_{i,n}^{(1)}$ such that for each consecutive pair of hyperplanes $\mathbf{W}_i^{(1)}\mathbf{x} + \hat{b}_{i,k}^{(1)} = 0$ and $\mathbf{W}_i^{(1)}\mathbf{x} + \hat{b}_{i,k+1}^{(1)} = 0$, exactly one input $\mathbf{x}_j$ in the client's local dataset lies in-between them. Note that as the bias transitions between consecutive values in the sequence, a single additional input activates the neuron i.

We now explain how the server determines this sequence of bias values and how it can reconstruct the isolated samples. To facilitate the exposition, we introduce an assumption here, which we justify in Sec. 4.3.

Assumption 4.1. For each input $(\mathbf{x}_j, y_j) \in \mathcal{B}$ and each value of $b_i^{(1)}$ tested by the attacker, if neuron i is activated by the input $\mathbf{x}_j$ for a given value of $b_i^{(1)}$, the derivative $\partial\mathcal{L}_j/\partial b_i^{(1)}$ remains independent of $b_i^{(1)}$.

According to (5), for a given bias value $b_{i,k}^{(1)}$, the server observes the client's aggregate gradient $\mathbf{g}_k = \sum_{j=1}^n \alpha_{j,k}\mathbf{x}_j$ Given hyperplanes $\mathbf{W}_i^{(1)}\mathbf{x}+b_{i,k}^{(1)} = 0$ and $\mathbf{W}_i^{(1)}\mathbf{x}+b_{i,k+1}^{(1)} = 0$, the attacker can determine whether they enclose at least one point by comparing $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$. Indeed, if there is no input point between the two hyperplanes, the same set of inputs activates neuron i and $\alpha_{i,k} = \alpha_{i,k+1}$ (because of Assumption 4.1), then $\mathbf{g}_k = \mathbf{g}_{k+1}$. If $\mathbf{g}_k \neq \mathbf{g}_{k+1}$, then the server knows that there is at least one input in-between and, by binary search, can progressively isolate the inputs as far as their projections along the direction $(\mathbf{W}_i^{(1)})^\intercal$ differ by more than a threshold ε (see App. D for its configuration rule).

After isolating all the input points, the attacker can reconstruct the inputs sequentially from $\mathbf{x}_1$ to $\mathbf{x}_n$, where we relabel the inputs so that $\mathbf{x}_j \in \mathcal{B}_{\mathbf{x}}$ is the input between the two hyperplanes corresponding to $\hat{b}_{i,j-1}^{(1)}$ and $\hat{b}_{i,j}^{(1)}$. Input $\mathbf{x}_1$ can be directly reconstructed using (2) with the client's loss gradient computed for $\hat{b}_{i,1}^{(1)}$. We note that the attacker also knows $\partial\mathcal{L}_1/\partial b_i^{(1)}$ for $b^{(1)} = \hat{b}_{i,1}^{(1)}$, i.e., when $\mathbf{x}_1$ activates neuron i. Suppose now that the attacker already reconstructed the first k inputs $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and the corresponding partial derivatives $\partial\mathcal{L}_1/\partial b_i^{(1)}, \ldots, \partial\mathcal{L}_k/\partial b_i^{(1)}$ when the corresponding inputs activate neuron i. Let h_k denote the partial derivative of the loss for $\hat{b}_{i,k}^{(1)}$, i.e., $h_k = \partial\mathcal{L}/\partial b_i^{(1)}$. Due to (4) and Ass. 4.1, the attacker can reconstruct $\partial\mathcal{L}_{k+1}/\partial b_i^{(1)}$ by

$$\frac{\partial\mathcal{L}_{k+1}}{\partial b_i^{(1)}} = n(h_{k+1} - h_k). \tag{7}$$

The attacker can then compute the set of coefficients $\alpha_{j,k+1}$ in the linear combination of $\mathbf{g}_{k+1}$ in (6) and reconstruct $\mathbf{x}_{k+1}$ by

$$\mathbf{x}_{k+1} = \frac{\mathbf{g}_{k+1} - \sum_{j=1}^k \alpha_{j,k+1}\mathbf{x}_j}{\alpha_{j,k+1}}. \tag{8}$$

While we have described the process in two consecutive phases—first isolating the inputs with the hyperplanes and then reconstructing the inputs—we note that these procedures can be intertwined. In particular, inputs $\mathbf{x}_1, \ldots, \mathbf{x}_k$ can be reconstructed as soon as the corresponding bias values $\hat{b}_{i,1}^{(1)}, \ldots, \hat{b}_{i,k}^{(1)}$ have been identified.

4.3 CONTROLLING THE GRADIENTS

Assumption 4.1 requires that the gradients $\partial\mathcal{L}_j/\partial b_i^{(1)}$ are constant across attack rounds (i.e., for different values of $b_i^{(1)}$, as far as the corresponding input activates neuron i).

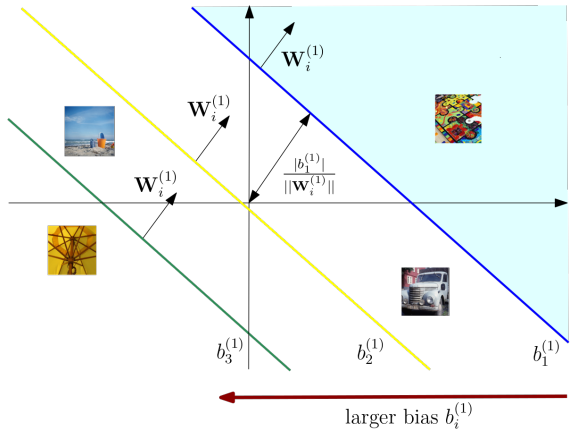


Figure 1: Illustration of our attack strategy, in which hyperplanes are positioned to isolate a single image within the strip they define. The light-blue region represents the half-space where $\mathbf{W}_i^{(1)}\mathbf{x} + b_1^{(1)} > 0$.

Here we show how we can satisfy this assumption for any classification task over the set of classes $\mathcal{C}$. Consider a model that employs a standard cross-entropy loss function. Given a sample $(\mathbf{x}_j, y_j)$, we have

$$\mathcal{L}_j = -\log\left(\frac{\exp\left(z_{y_j}^{(2)}\right)}{\sum_{c\in\mathcal{C}}\exp\left(z_c^{(2)}\right)}\right).$$

Simple calculations (App. B) show that if image j activates neuron i, then

$$\frac{\partial\mathcal{L}_j}{\partial b_i^{(1)}} = -w_{y_j,i}^{(2)} + \sum_{k\in\mathcal{C}} w_{k,i}^{(2)}\frac{\exp\left(z_k^{(2)}\right)}{\sum_{c\in\mathcal{C}}\exp\left(z_c^{(2)}\right)}, \tag{9}$$

where $z_k^{(2)} = \mathbf{W}_k^{(2)}(\text{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_j + \mathbf{b}^{(1)})) + b_k^{(2)}$. If we now set large enough bias values in the second layer, we obtain $\frac{\exp\left(z_k^{(2)}\right)}{\sum_{c\in\mathcal{C}}\exp\left(z_c^{(2)}\right)} \approx \frac{1}{|\mathcal{C}|}$, and then

$$\frac{\partial\mathcal{L}_j}{\partial b_i^{(1)}} \approx -w_{y_j,i}^{(2)} + \sum_{k\in\mathcal{C}} w_{k,i}^{(2)}\frac{1}{|\mathcal{C}|}. \tag{10}$$

In this way, the partial derivatives become independent of $b_i^{(1)}$ and α_j can be artificially set to a non-negligible value without knowledge of the y_j.While this reasoning holds for any choice of the matrix $\mathbf{W}^{(2)}$, we require $\partial\mathcal{L}_j/\partial b_i^{(1)} \neq 0$ for the input $\mathbf{x}_j$ to have a non-zero coefficient in the aggregate gradient. Furthermore, for the efficient parallel search described in the next section, we want $\partial\mathcal{L}_j/\partial b_{i_1}^{(1)} = \partial\mathcal{L}_j/\partial b_{i_2}^{(1)}$ across all neurons, which requires the columns of $\mathbf{W}^{(2)}$ to be identical. These constraints are easily satisfied since the attacker has control over all model parameters.

Algorithm 1 Parallel attack

Input: $[l_1, u_1]$ initial search space of bias values, the set of attack rounds $\mathcal{T} = \{1, ..., T\}$

$\mathcal{G} \leftarrow \{\}$
$\mathcal{I} \leftarrow \{[l_1, u_1]\}$
Draw a vector $\mathbf{w} \in \mathbb{R}^d$ and a vector $\mathbf{v} \in \mathbb{R}^C$, set each row of $\mathbf{W}^{(1)}$ to $\mathbf{w}$ and each column of $\mathbf{W}^{(2)}$ to $\mathbf{v}$
Set all entries of $\mathbf{b}^{(2)}$ to the same large value
for $t \in \{1, ..., T\}$ **do**
  $\mathbf{b}^{(1)} \leftarrow \text{UpdateHyperplanes}(\mathcal{I})$
  server sends malicious parameters θ to the client
  server receives gradient updates $\frac{\partial\mathcal{L}}{\partial\mathbf{W}^{(1)}}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{b}^{(1)}}$
  server computes $\mathbf{g}_i$ using (5), $\forall i \in \{1, ..., N\}$
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{(\mathbf{g}_i, \frac{\partial\mathcal{L}}{\partial b_i^{(1)}}, b_i^{(1)}), \forall i = \{1, ..., N\}\}$
  $\mathcal{G}, \mathcal{I} \leftarrow \text{UpdateSearchState}(\mathcal{G})$
Reconstruct the inputs from $\mathcal{G}$ using (8)

4.4 PARALLEL SEARCH

We now present the complete version of our attack. So far, we have described a method in which the attacker tests a different bias value for neuron i in different rounds. This approach can be parallelized across all the N neurons in the layer, significantly reducing the number of required communication rounds. The procedure is described in Algorithm 1.

The initial search space for the bias values, $[l_1, u_1]$, is determined based on the range of possible input feature values. In the first communication round, this interval is divided into N equal-length subintervals. The procedure UpdateSearchState in Algorithm 2 identifies which subintervals require further exploration. Subintervals containing no inputs, as well as those whose size becomes smaller than the threshold ε, are discarded.

At each round, $\mathcal{I}$ represents the set of remaining subintervals. The attacker tests N new bias values within these subintervals, distributing the values roughly evenly among them. More precisely, if N cannot be evenly divided among the subintervals, the extra bias values are assigned to the longer intervals. This procedure, called UpdateHyperplanes, is detailed in Algorithm. 3.

Algorithm 2 UpdateSearchState

Input: The set of strips $\mathcal{G}$.

$\mathcal{I} \leftarrow \{\}$, $\mathcal{G}_{\text{new}} \leftarrow \{\}$
sort $\mathcal{G}$ by ascending value of the bias values
for $i = 2, \ldots, |\mathcal{G}|$ **do**
  if $\mathbf{g}_i \neq \mathbf{g}_{i-1}$ **and** $b_i - b_{i-1} \geq \varepsilon$ **then**
    $\mathcal{G}_{\text{new}} \leftarrow \mathcal{G}_{\text{new}} \cup \{\left(\mathbf{g}_i, \frac{\partial\mathcal{L}}{\partial b_i^{(1)}}, b_i^{(1)}\right)\}$
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{[b_{i-1}^{(1)}, b_i^{(1)}]\}$
Return $\mathcal{G}_{\text{new}}, \mathcal{I}$

Algorithm 3 UpdateHyperplanes

Input: set of search intervals $\mathcal{I} = \{[l_1, u_1], [l_2, u_2], \ldots, [l_k, u_k], \ldots\}$

1: $M = |\mathcal{I}|$
2: **if** $M = 1$ **then**
3: $\quad \mathbf{b}^{(1)} \leftarrow \left\{l_1 + i\frac{u_1 - l_1}{N+1} | i = 1, \ldots, N\right\}$
4: **else**
5: $\quad$ sort $\mathcal{I}$ by descending length $(u_k - l_k)$
6: $\quad r \leftarrow N \bmod M,\ q \leftarrow \lfloor N/M \rfloor,\ j \leftarrow 0$
7: $\quad$ **for** $k = 1, ..., M$ **do**
8: $\quad\quad$ **if** $k < r$ **then**
9: $\quad\quad\quad \mathbf{b}^{(1)}_{j:j+q} \leftarrow \{l_k + i\frac{u_k - l_k}{q+2} | i = 1, \ldots, q+1\}$
10: $\quad\quad\quad j \leftarrow j + q + 1$
11: $\quad\quad$ **else**
12: $\quad\quad\quad \mathbf{b}^{(1)}_{j:j+q-1} \leftarrow \{l_k + i\frac{u_k - l_k}{q+1} | i = 1, \ldots, q\}$
13: $\quad\quad\quad j \leftarrow j + q$
14: Return $\mathbf{b}^{(1)}$

4.5 COMPLEXITY ANALYSIS

To evaluate the time complexity of the attack (Alg. 1), we decompose the operations performed by the server during each attack round, and subsequently analyze the total computational cost.

At each round, the server first prepares the malicious hyperplanes for the targeted client (line 6). This procedure requires $\mathcal{O}(n \log n)$ operations, primarily due to the sorting operation on the set of subintervals $\mathcal{I}$ of size at most n (line 5 in Alg. 3).[2] After receiving the gradients from clients, the server then computes N observations $\mathbf{g}_i$ of size d, which requires $\mathcal{O}(Nd)$ operations, and adds them to the current set of strips $\mathcal{G}$ (lines 9 and 10). After this step, the size of $\mathcal{G}$ becomes at most $n + N$. To prepare the next-round attack (Alg. 2), the server updates the search space over the strips, which involves sorting the set $\mathcal{G}$ and iterating over it to compare vectors $\mathbf{g}_i$. This step incurs a computational cost of $\mathcal{O}(\max((N+n)\log(n+N), (n+N)d))$.

We then analyze the number of rounds T required to isolate all inputs (i.e., achieve full batch recovery) with high probability. Assuming that the hyperplane $\mathbf{w}$ is drawn from a standard normal distribution and that $N \geq n$, we show that T is $\mathcal{O}(\log \frac{n}{N})$ when the threshold ϵ is set to $\mathcal{O}(1/n^2)$ (Appendix. D).

The final reconstruction step requires an additional $\mathcal{O}(nd)$ operations (line 12). With hardware optimized for parallel computation, the dimensional dependency on d in the above steps can be effectively mitigated, resulting in a total complexity of $\mathcal{O}\left(\log \frac{n}{N}((N+n)\log(n+N))\right)$.

[2]As we can see from Alg. 2, $\mathcal{I}$ and $\mathcal{G}_{new}$ consist of non-overlapping intervals, each containing at least one point. Therefore, given a batch size of n, we have $|\mathcal{I}| \leq n$ and $|\mathcal{G}_{new}| \leq n$.

5 EXPERIMENTS

We conduct experiments to demonstrate the efficacy of our proposed attack on both image and tabular data [3]. Our evaluation encompasses two distinct datasets, each representing a different data modality. For image classification tasks, we use the ImageNet ILSVRC 2012 dataset [Russakovsky et al., 2015], that comprises a total of 1000 classes. As preprocessing steps, all images are rescaled to the $[0, 1]$ range and resized to 224×224 pixels. To evaluate performance on tabular data, we test the Human Activity Recognition Using Smartphones dataset [Anguita et al., 2013] (HARUS), which includes recordings of 30 subjects performing various activities while carrying a waist-mounted smartphone equipped with embedded sensors. The dataset comprises 561 features and 6 classification labels, representing the activity performed by each user. All features are scaled between -1 and 1. Each client trains an FC-NN with two layers. Unless stated otherwise, the first layer consists of $N = 1000$ neurons. For all the experiments, we assume that training is performed using FedSGD with full-batch updates. An image is considered reconstructed if it has SSIM ≥ 0.99 relative to the true image (Figures 9 and 10 provide examples of reconstructed images). For tabular data, we consider an input to be perfectly reconstructed if the L2-norm difference between the true and recovered data point is < 0.1.

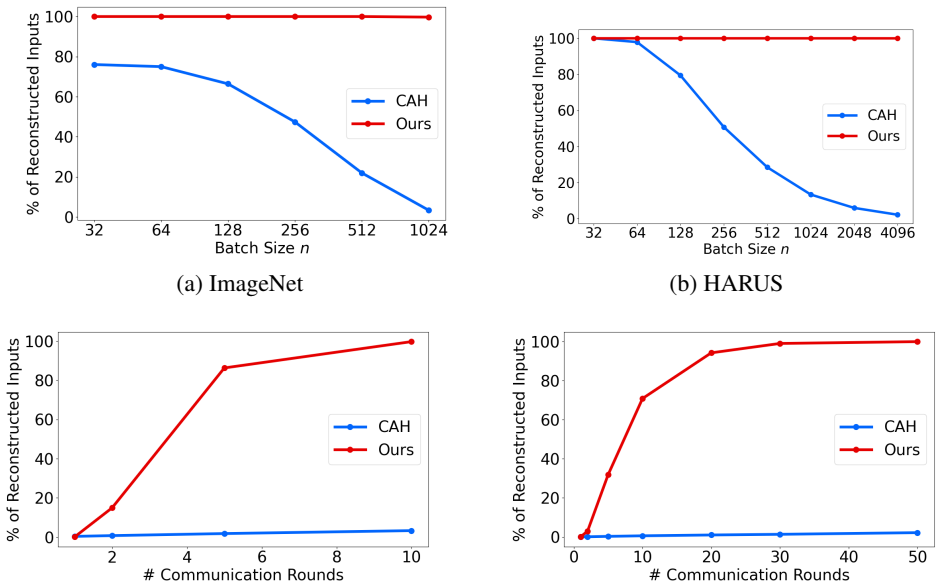
We compare our method to the *Curious Abandon Honesty* (CAH) attack [Boenisch et al., 2023], allowing, for a fair evaluation, their attack to be performed across multiple communication rounds. Each experiment is repeated three times with different random seeds to ensure consistency and reliability of the results. Details on the configuration of both attacks can be found in App. E.

5.1 EXPERIMENTAL RESULTS

The experimental results demonstrate the superior performance of our attack over the baseline across both ImageNet and HARUS datasets. On ImageNet, after 10 communication rounds, our attack consistently outperforms the baseline across all tested batch sizes, successfully recovering the entire batch in every scenario, see Figure 2a. Similarly, on HARUS, our attack achieves significant improvements, recovering 4,096 data points after 50 rounds, see Figure 2b. On both datasets, we observe a decrease in the fraction of samples reconstructed by CAH as the batch size increases. This demonstrates the baseline's limitations in recovering large batches when the input dimension is held constant, which is in accordance with our theoretical findings in Sec. 3.

Figures 2c and 2d further highlight the advantages of our attack and its parallel search strategy. Initially, the percentage of recovered inputs remains below 2% of the full batch

[3]Code: `https://github.com/francescodiana99/cutting-through-privacy`



(a) ImageNet (b) HARUS

(c) ImageNet (d) HARUS

Figure 2: The percentage of perfectly reconstructed inputs on ImageNet and HARUS for a two-layer FC NN with $N = 1000$ neurons in the first layer. In 2a the reconstruction is evaluated after 10 communication rounds, while in 2b it is measured after 50 rounds. Figure 2c illustrates the impact of the number of communication rounds for batch size $n = 1024$, and 2d for $n = 4096$.

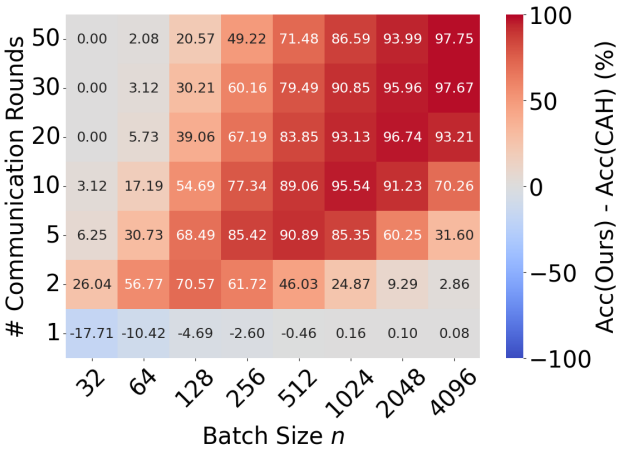


Figure 3: Difference in the percentage of correctly reconstructed inputs rate between our attack and the CAH attack on the HARUS dataset.

for both the attacks but, as the number of communication rounds increases, the success rate of our method improves significantly. In contrast, the baseline shows only marginal gains, as it relies on a randomized search approach.

Figure 3 illustrates the performance difference between our attack and CAH on the HARUS dataset. Two key trends emerge from our results. First, the performance gap increases with both the number of communication rounds and the batch size. This is clearly visible in the figure, with the largest performance difference (97.75 percentage points) observed at 50 rounds and a batch size of 4096. Second, different regions of the heatmap reveal the relative strengths of each approach. The top-left region (small batches, many rounds) suggests CAH's effectiveness on smaller batches; the bottom row highlights its advantage in one-shot attacks. However, this advantage relies on their trap weights initialization, which requires the attacker to have access to some representative data to tune some hyperparameters (see App. E). This relative advantage disappears as the batch size increases. The top-right region (large batches, many rounds) highlights our attack's ability to scale, demonstrating its potential for larger batches. This supports our claim that a sufficient number of rounds enables recovery for arbitrary large batches.

We also evaluate the effect of the number of neurons N in the fully connected attack layer and its relationship with attack accuracy, see Table 1. While we observe that the percentage of reconstructed data increases with a larger number of neurons for both attacks, our attack outperforms the baseline in all but one combination of rounds and neuron counts. In many cases, our attack reconstructs nearly 100 times more inputs than CAH. We highlight a key distinction: the baseline reconstructs around 3.6% of the input after 50 rounds with 2000 neurons, i.e., using 100 000 different random hyperplanes. On the other hand, our attack manages to reconstruct 38% of the batch after only 10 rounds with only 500 neurons, i.e., using only 5000 different hyperplanes. In comparison, our attack reconstructs more than 10 times as

Table 1: Effect of number of neurons N in the reconstruction layer. Each value indicates the percentage of data exactly recovered in attacks on HARUS dataset, when $n = 4096$.

# Rounds	# Neurons						
		100	200	500	1000	1500	2000
1	Ours	0.00±0.00	0.2±0.03	0.05±0.02	0.15±0.06	0.15±0.10	0.21±0.09
	CAH	0.01±0.03	0.02±0.01	0.04±0.01	0.07± 0.04	0.13±0.02	0.15±0.05
5	Ours	1.11±0.31	2.82±0.37	9.33±4.66	31.93±1.80	49.60±0.80	64.80±2.08
	CAH	0.07±0.03	0.08±0.01	0.20±0.04	0.33±0.06	0.50±0.03	0.66±0.09
10	Ours	3.11±0.88	8.33±2.70	38.02±1.54	70.87±1.44	86.54±0.44	93.17±0.19
	CAH	0.08±0.06	0.15±0.03	0.31±0.07	0.62±0.01	0.81±0.09	1.06±0.08
30	Ours	21.10±1.44	51.64±1.57	88.38±0.58	99.06±0.29	99.85±0.10	99.98±0.03
	CAH	0.25±0.07	0.37±0.09	0.78±0.09	1.38±0.08	1.85±0.13	2.46±0.20
50	Ours	42.52±2.07	75.36±1.68	97.84±0.21	99.98±0.03	100.00±0.00	100.00±0.00
	CAH	0.37±0.07	0.53±0.12	1.24±0.24	2.23±0.10	2.70±0.12	3.62±0.47

many input points using 20 times fewer hyperplanes than the baseline, thanks to its efficient search procedure for correctly positioning the hyperplanes that isolate the inputs.

In terms of computational overhead, we evaluate our attack on the most computationally demanding setting—ImageNet with $n = 1024$. In this case, the attack requires less than two seconds per round (primarily for lines 9 and 11 in Alg. 1) and under four seconds for the final image reconstruction phase (line 12), using an NVIDIA GeForce RTX 2080 Ti GPU. Since attacks are carried out across consecutive rounds, this additional overhead of approximately two seconds per round is negligible compared to the tens of seconds typically required to transmit the model in a single communication round.

We provide additional experimental results on HARUS, ImageNet, and CIFAR-10 [Krizhevsky, 2009] in App. F, including tests on a CNN architecture and extension to the multiple local steps setting, relaxing the assumption of full-batch updates. We also evaluate the effect of data heterogeneity and the effect of noise. These results confirm the same qualitative trends observed in our main results, further validating our conclusions.

6 DISCUSSION

Secure aggregation [Bonawitz et al., 2017] is often proposed as a defense mechanism against privacy attacks, as it aggregates updates from multiple clients without revealing individual client updates to the server. However, the server still obtains the average of the gradients computed over the entire union of client datasets. With a sufficient number of communication rounds, our attack could still recover all underlying data, though not necessarily associate it with a specific client.

Therefore, the only robust defense against our attack that we are aware of would involve implementing local differential privacy (LDP) [Dwork, 2006, Bhowmick et al., 2019] on the client side. By adding noise to updates, LDP alters the coefficients associated to each input in (3), thus hindering precise localization and subsequent reconstruction of the data points, at the cost of decreased model utility. We conducted a preliminary evaluation of our attack's performance when clients add Gaussian noise to their updates. Results in App. F show that noise perturbation decreases the proportion of inputs that our attack can perfectly recover. Nevertheless, our method remains effective: even when clients add noise, data reconstruction is still feasible and we consistently outperform the baseline in terms of reconstruction accuracy.

7 FUTURE WORKS AND CONCLUSION

In this work, we proposed a novel data reconstruction attack in Federated Learning for classification tasks, demonstrating that a malicious server can manipulate model parameters to perfectly recover clients' input data, regardless of the batch size.

Exploring the applicability of our attack to other machine learning tasks and architectures remains a promising direction for future work. Although our current approach is broadly applicable—by first recovering embeddings from the first fully connected layer and then applying a model inversion attack, following the methodology of Kariyappa et al. [2023] and Zhang et al. [2023]—the resulting reconstructions would still not be perfect. Therefore, developing analytical attacks that enable accurate, end-to-end reconstruction from more complex architectures and from different tasks remains an important and promising topic for future research.

Extending this attack to settings where clients perform multiple local epochs presents also an interesting challenge. In App. F, we carried out a preliminary evaluation by considering a scenario with one local epoch using minibatch stochastic gradient descent. However, effectively handling cumulative minibatch updates over multiple local epochs remains an open question.

Acknowledgements

This research was supported in part by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002 and in part by the European Network of Excellence dAIEDGE under Grant Agreement Nr. 101120726. It was also funded in part by the Groupe La Poste, sponsor of the Inria Foundation, in the framework of the FedMalin Inria Challenge and by the EU HORIZON MSCA 2023 DN project FINALITY (G.A. 101168816). André Nusser was supported by the French government through the France 2030 investment plan managed by the National Research Agency (ANR), as part of the Initiative of Excellence of Université Côte d'Azur under reference number ANR-15-IDEX-01. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *The European Symposium on Artificial Neural Networks*, 2013. URL https://api.semanticscholar.org/CorpusID:6975432.

Jon Louis Bentley, H. T. Kung, Mario Schkolnick, and Clark D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, 25 (4):536–543, 1978. doi: 10.1145/322092.322095. URL https://doi.org/10.1145/322092.322095.

Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning, 2019. URL https://arxiv.org/pdf/1812.00984.

Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199, 2023. doi: 10.1109/EuroSP57164.2023.00020.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3133982. URL https://doi.org/10.1145/3133956.3133982.

Bradley CA Brown, Anthony L. Caterini, Brendan Leigh Ross, Jesse C Cresswell, and Gabriel Loaiza-Ganem. Verifying the union of manifolds hypothesis for image data. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Rvee9CAX4fi.

Chen Chen, Lingjuan Lyu, Han Yu, and Gang Chen. Practical Attribute Reconstruction Attack Against Federated Learning . *IEEE Transactions on Big Data*, 10(06):851–863, December 2024. ISSN 2332-7790. doi: 10.1109/TBDATA.2022.3159236. URL https://doi.ieeecomputersociety.org/10.1109/TBDATA.2022.3159236.

Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*, 2016. URL https://arxiv.org/abs/1604.00981.

Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1964–1974. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/choquette-choo21a.html.

Hong-Min Chu, Jonas Geiping, Liam H Fowl, Micah Goldblum, and Tom Goldstein. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *The Eleventh International Conference on Learning Representations*, 2023.

Francesco Diana, Othmane Marfoq, Chuan Xu, Giovanni Neglia, Frédéric Giroire, and Eoin Thomas. Attribute inference attacks for federated regression tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):16271–16279, Apr. 2025. doi: 10.1609/aaai.v39i15.33787. URL https://ojs.aaai.org/index.php/AAAI/article/view/33787.

Dimitar Iliev Dimitrov, Mislav Balunovic, Nikola Konstantinov, and Martin Vechev. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=e7A0B99zJf.

Dimitar Iliev Dimitrov, Maximilian Baader, Mark Niklas Mueller, and Martin Vechev. SPEAR: Exact gradient inversion of batches in federated learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lPDxPVS6ix.

Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fwzUgo0FM9v.

Liam H Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Kostadin Garov, Dimitar Iliev Dimitrov, Nikola Jovanović, and Martin Vechev. Hiding in plain sight: Disguising data stealing attacks in federated learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=krx5512A6G.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Sanjay Kariyappa, Chuan Guo, Kiwan Maeng, Wenjie Xiong, G. Edward Suh, Moinuddin K Qureshi, and Hsien-Hsin S. Lee. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15884–15899. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kariyappa23a.html.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Jiahao Lu, Xi Sheryl Zhang, Tianli Zhao, Xiangyu He, and Jian Cheng. April: Finding the achilles' heel on privacy for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019. doi: 10.1109/SP.2019.00029.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/8a1e808b55fde9455cb3d8857ed88389-Paper.pdf.

Milad Nasr, Reza Shokri, and Amir Houmansadr. 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)*, pages 739–753, 2019. doi: 10.1109/SP.2019.00065.

Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2429–2443, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394505. doi: 10.1145/3548606.3560557. URL https://doi.org/10.1145/3548606.3560557.

Andrew J. Paverd and Andrew C. Martin. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. 2014. URL https://api.semanticscholar.org/CorpusID:211141069.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning: Revisited and enhanced. In Lynn Batten, Dong Seong Kim, Xuyun Zhang, and Gang Li, editors, *Applications and Techniques in Information Security*, pages 100–110, Singapore, 2017. Springer Singapore.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XJk19XzGq2J.

H. Raynaud. Sur l'enveloppe convexe des nuages de points aleatoires dans rn. i. *Journal of Applied Probability*, 7(1): 35–48, 1970. doi: 10.2307/3212146.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

Mark Vero, Mislav Balunović, Dimitar I. Dimitrov, and Martin Vechev. Tableak: tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Feng Wang, Senem Velipasalar, and M. Cenk Gursoy. Maximum knowledge orthogonality reconstruction with gradients in federated learning. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3872–3881, 2024. doi: 10.1109/WACV57701.2024.00384.

Yuxin Wen, Jonas A Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning*, pages 23668–23684. PMLR, 2022.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16332–16341, 2021. doi: 10.1109/CVPR46437.2021.01607.

Shuaishuai Zhang, Jie Huang, Zeping Zhang, Peihao Li, and Chunyang Qi. Compromise privacy in large-batch federated learning via model poisoning. *Information Sciences*, 647:119421, 2023. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2023.119421. URL https://www.sciencedirect.com/science/article/pii/S002002552301006X.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients, 2020. URL https://arxiv.org/abs/2001.02610.

Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation . In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1287–1305, Los Alamitos, CA, USA, May 2024. IEEE Computer Society. doi: 10.1109/SP54263.2024.00030. URL https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00030.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

Cutting Through Privacy: A Hyperplane-Based Data Reconstruction Attack in Federated Learning (Supplementary Material)

Francesco Diana[1,2] **André Nusser**[1,2,3,4] **Chuan Xu**[1,2,3,4] **Giovanni Neglia**[1,2]

[1]Université Côte d'Azur , France
[2]Inria, , France
[3]CNRS, France
[4]I3S, France

A PROOF OF THEOREM 3.1

Proof. By definition of the attack method, a point can only be reconstructed using (2) if it is linearly separated by injecting malicious weights. Thus, the number of recovered points is upper-bounded by the number of linearly separable points in the batch $\mathcal{B}_{\mathbf{x}}$, which corresponds to the number of vertices on its convex hull. This value is bounded by:

1. $O(n^{(d-1)/(d+1)})$ if $\mathcal{B}_{\mathbf{x}}$ is sampled uniformly at random from a unit ball [Raynaud, 1970], or
2. $O(\log^{d-1} n)$ if $\mathcal{B}_{\mathbf{x}}$ is sampled uniformly at random from the unit hypercube [Bentley et al., 1978], or
3. $O(\log^{(d-1)/2} n)$ if $\mathcal{B}_{\mathbf{x}}$ is sampled from a normal distribution with covariance matrix $\mathbf{I}_d$ [Raynaud, 1970]. □

B PARTIAL DERIVATIVE WITH RESPECT TO THE BIAS

By application of the chain rule, the gradient $\frac{\partial \mathcal{L}_j}{\partial z_i^{(1)}}$ can be computed as

$$\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}} = \frac{\partial \mathcal{L}_j}{\partial z_i^{(1)}} = \frac{\partial \mathcal{L}_j}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial z_i^{(1)}}. \tag{11}$$

Considering the k-th element in $\mathbf{z}^{(2)}$, the first term in (11) is

$$\frac{\partial \mathcal{L}_j}{\partial z_k^{(2)}} = \begin{cases} -1 + \frac{\exp(z_k^{(2)})}{\sum_{c\in\mathcal{C}} \exp(z_c^{(2)})}, & \text{if } k = y_j \\ \\ \frac{\exp(z_k^{(2)})}{\sum_{c\in\mathcal{C}} \exp(z_c^{(2)})}, & \text{if } k \neq y_j \end{cases}, \tag{12}$$

and the second term in (11) is

$$\frac{\partial z_k^{(2)}}{\partial z_i^{(1)}} = w_{k,i}^{(2)}. \tag{13}$$

By substituting (12) and (13) into (11), we obtain:

$$\begin{aligned} \frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}} &= w_{y_j,i}^{(2)} \left(-1 + \frac{\exp(z_{y_j}^{(2)})}{\sum_{c\in\mathcal{C}} \exp(z_c^{(2)})} \right) + \sum_{\substack{k\in\mathcal{C} \\ k\neq y_j}} w_{k,i}^{(2)} \frac{\exp(z_k^{(2)})}{\sum_{c\in\mathcal{C}} \exp(z_c^{(2)})} \\ &= -w_{y_j,i}^{(2)} + \sum_{k\in\mathcal{C}} w_{k,i}^{(2)} \frac{\exp(\mathbf{W}_k^{(2)}(\mathbf{W}^{(1)}\mathbf{x}_j + \mathbf{b}^{(1)}) + b_k^{(2)})}{\sum_{c\in\mathcal{C}} \exp(\mathbf{W}_c^{(2)}(\mathbf{W}^{(1)}\mathbf{x}_j + \mathbf{b}^{(1)}) + b_c^{(2)})}. \end{aligned} \tag{14}$$

Table 2: Effect of the number of hidden layers. Each value indicates the percentage of data exactly recovered on attacks on ImageNet, when $n = 512$.

# Rounds	**# Hidden Layers**			
	1	2	3	4
1	0.20±0.00	0.20±0.00	0.20±0.00	0.20±0.00
2	35.61±3.12	35.16±3.32	35.29±3.33	35.29±3.33
3	78.38±2.79	77.99±2.93	78.26±2.66	78.19±2.74
5	97.14±0.98	97.01±1.19	97.14±0.98	97.14±0.98
7	99.74±0.45	99.74±0.45	99.74±0.45	99.74±0.45
10	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00

C MODIFICATIONS FOR NEURAL NETWORKS WITH MULTIPLE LAYERS

Our method can be easily adapted to any fully-connected neural network with L layers with ReLU activation function. In the following, to simplify the notation we will omit the ReLU activation, assuming that the input is propagated through each layer only if the corresponding output is positive.

Let us start by considering the case of a network with two hidden layers, such that

$$\begin{aligned}\mathbf{z}^{(1)} &= \text{ReLU}(\mathbf{W}^{(1)}\mathbf{x}_j + \mathbf{b}^{(1)}),\\ \mathbf{z}^{(2)} &= \text{ReLU}(\mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)})\\ \mathbf{z}^{(3)} &= \mathbf{W}^{(3)}\mathbf{z}^{(2)} + \mathbf{b}^{(3)},\end{aligned}$$

where $\mathbf{W}^{(3)} \in \mathbb{R}^{C\times m}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{m\times N}$, and $\mathbf{W}^{(1)} \in \mathbb{R}^{N\times d}$. For a fixed neuron i, we want to compute the gradient $\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}}$. Using the chain rule, we can express it as:

$$\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}} = \frac{\partial \mathcal{L}_j}{\partial z_i^{(1)}} = \frac{\partial \mathcal{L}_j}{\partial \mathbf{z}^{(3)}} \cdot \frac{\partial \mathbf{z}^{(3)}}{\partial \mathbf{z}^{(2)}} \cdot \frac{\partial \mathbf{z}^{(2)}}{\partial z_i^{(1)}}, \tag{15}$$

Now, substituting the result from (14) into this expression, we get:

$$\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}} = \sum_m \left(\sum_{k\in\mathcal{C}} \left(-\mathbb{1}_{k=y_j} + \frac{\exp\left(z_k^{(3)}\right)}{\sum_{c\in\mathcal{C}} \exp\left(z_c^{(3)}\right)} \right) w_{k,m}^{(3)} \right) w_{m,i}^{(2)}, \tag{16}$$

where $\mathbb{1}_{k=y_j}$ indicates that the term is 1 only when $k = y_j$, (i.e., when the class index matches the true label y_j). To extend this derivation to a network with L layers, we can recursively apply the chain rule and obtain:

$$\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}} = \sum_{m_1} \cdots \sum_{k\in\mathcal{C}} \left(\gamma_k^{(L)} \prod_{l=1}^{L} w_{m_l,m_{l-1}} \right) w_{m_1,i}, \tag{17}$$

where $\gamma_k^{(L)} = \frac{\exp\left(z_k^{(L)}\right)}{\sum_{c\in\mathcal{C}} \exp\left(z_c^{(L)}\right)}$, and $m_1, ..., m_L$ indicate the number of rows in $\mathbf{W}^{(L)}$.

Therefore, by assigning identical random values to each column of the hidden layers $\mathbf{W}^{(2)}...\mathbf{W}^{(L-1)}$, we can ensure identical inputs for all classification neurons, reducing potential errors due to varying gradient contributions $\frac{\partial \mathcal{L}_j}{\partial b_i^{(1)}}$.

To validate our claim, we conduct experiments on ImageNet using models with 1, 2, 3, and 4 hidden layers. The first layer has $N = 1000$, while each subsequent layer consists of 100 neurons. We set the malicious parameters of the input and classification layers as described in App. E, and generate the columns of each additional hidden layer from $\mathcal{N}(0, 10^{-6})$. The corresponding bias terms are also scaled by a 10^{-3} factor. Table 2 demonstrates that, regardless of the number of hidden layers, the reconstruction accuracy of the attack exhibits only minor fluctuations, confirming that the number of layers does not significantly affect the performance of our attack.

D BOUND ON THE NUMBER OF ROUNDS

Assume that the number of neurons is at least the number of input points that we want to recover, i.e., $N \geq n$. In the first step we cut the initial interval of width W into intervals of width $\frac{W}{N}$. In any later step, we cut each interval at least $\lfloor \frac{N}{n} \rfloor$ times and hence, the intervals in the next round have a width that is reduced by a factor of at least $\lfloor \frac{N}{n} \rfloor + 1$. We stop when the interval width reaches ε. Hence, the number of steps is bounded by

$$\left\lceil \log_{\lfloor \frac{N}{n} \rfloor + 1} \left(\frac{W}{N \cdot \varepsilon} \right) \right\rceil + 1 \leq \left\lceil \log_2 \left(\frac{W}{N \cdot \varepsilon} \right) \right\rceil + 1. \tag{18}$$

It is easy to show that this analysis is tight by an example where each interval initially contains a single image.

Finally, we show how to select the value for ε. Let $\mathcal{B}_{\mathbf{x}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be the set of samples that the attacker wants to reconstruct. Assume that inputs have a minimum distance

$$\Delta = \min_{\substack{i,j \\ i \neq j}} \|\mathbf{x}_i - \mathbf{x}_j\|. \tag{19}$$

Draw a random direction vector $\mathbf{v} \sim \mathcal{N}(0, \mathbf{I}_{d \times d})$. The distance of two projections along $\mathbf{v}$ of points in $\mathcal{B}_{\mathbf{x}}$ will be the random variable

$$\mathrm{p}_{\mathbf{v}}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{v} \rangle \sim \mathcal{N}(0, \|\mathbf{x}_i - \mathbf{x}_j\|). \tag{20}$$

Therefore, we can compute:

$$\begin{aligned}
\mathbb{P}\left(|\mathrm{p}_{\mathbf{v}}(\mathbf{x}_i, \mathbf{x}_j) < \varepsilon|\right) &= \mathbb{P}\left(q \in \left[\frac{-\varepsilon}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \frac{\varepsilon}{\|\mathbf{x}_i - \mathbf{x}_j\|} \right] \right) && (21) \\
&\leq \mathbb{P}\left(q \in \left[\frac{-\varepsilon}{\Delta}, \frac{\varepsilon}{\Delta} \right] \right) && (22) \\
&\leq \frac{1}{\sqrt{2\pi}} 2 \frac{\varepsilon}{\Delta} = \sqrt{\frac{2}{\pi}} \frac{\varepsilon}{\Delta}, && (23)
\end{aligned}$$

where $q \sim \mathcal{N}(0, 1)$. Here, the last inequality uses the fact that the Gaussian pdf is maximized at 0 and our integration interval is symmetric around 0. Then, applying the union bound, we get:

$$\begin{aligned}
\mathbb{P}\left(\exists\, i, j \ \wedge\ i \neq j : |\mathrm{p}_{\mathbf{v}}(\mathbf{x}_i, \mathbf{x}_j) < \varepsilon|\right) &= \mathbb{P}\left(\bigcup_{\substack{i,j \\ i \neq j}} \{|\mathrm{p}_{\mathbf{v}}(\mathbf{x}_i, \mathbf{x}_j) < \varepsilon|\} \right) && (24) \\
&\leq \sum_{\substack{i,j \\ i \neq j}} \mathbb{P}\left(|\mathrm{p}_{\mathbf{v}}(\mathbf{x}_i, \mathbf{x}_j) < \varepsilon|\ \right) && (25) \\
&\leq \frac{n^2}{2} \sqrt{\frac{2}{\pi}} \frac{\varepsilon}{\Delta}. && (26)
\end{aligned}$$

Hence, by selecting $\varepsilon = \sqrt{2\pi} \frac{\Delta}{n^2} \delta$, all the projections will be at least ε-distant with high probability. After substituting the value of ε in (18), the number of rounds becomes:

$$\left\lceil \log_{\lfloor \frac{N}{n} \rfloor + 1} \left(\frac{W n^2}{\sqrt{2\pi} N \Delta \delta} \right) \right\rceil + 1 \leq \left\lceil \log_2 \left(\frac{W n^2}{\sqrt{2\pi} N \Delta \delta} \right) \right\rceil + 1, \tag{27}$$

with probability at least $1 - \delta$.

E ATTACKS CONFIGURATION

In our attacks, we used the same initialization strategies for ImageNet and HARUS. Each row of the input weights $\mathbf{W}^{(1)}$ is initialized with identical values drawn from a normal distribution $\mathbf{W}_i^{(1)} \sim \mathcal{N}(0, 10^{-2})$, while classification weights'

column values are drawn from $\mathbf{W}_i^{(2)} \sim \mathcal{N}(0, 10^{-2})$. However, the choice of the distribution does not represent a limiting factor, and different distribution can be employed. The classification bias values $\mathbf{b}_i^{(2)}$ are set to 10^{25}. We chose a large value for the classification bias according to (10). We did not select a specific value for ε, instead, we iteratively selected as many bias values as were compatible with the number of communication rounds T available to the attacker. Consequently, a total of NT bias values were selected following Alg. 3.

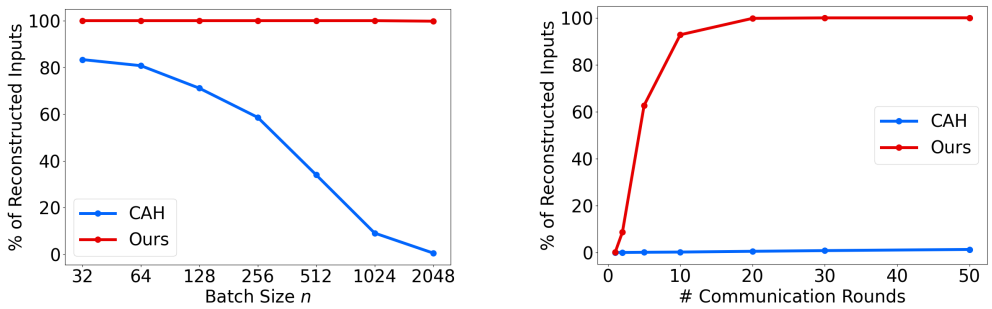
To test the CAH attack, we draw the weights $\mathbf{W}^{(1)} \sim \mathcal{N}(0, \frac{1}{2})$ and set the scaling factor $s = 0.99$ for ImageNet experiments, following the setup described in [Boenisch et al., 2023]. For experiments on HARUS, we draw the input weights from a normal distribution $\mathbf{W}^{(1)} \sim \mathcal{N}(0, 1)$ and select $s = 0.97$ after a tuning process.

To minimize reconstruction errors arising from numerical inaccuracy, the results in the main paper are obtained using double precision. However, additional single precision experiments are presented in App. F.

F ADDITIONAL EXPERIMENTAL RESULTS

F.1 CIFAR-10

FC-NN We conduct additional experiments on the CIFAR-10 dataset Krizhevsky [2009], evaluating our attack on the same two-layers neural network described in the main paper. For these experiments, our malicious model parameters are initialized according to the setup used in the ImageNet experiments, described in App. E, while for CAH, we set $s = 0.95$, according to the setup proposed in [Boenisch et al., 2023]. We perform the attack for batch size in {32, 64, 128, 256, 512, 1024, 2048} and evaluate after {1, 2, 5, 10, 20} rounds. Figure 4a illustrates the percentage of reconstructed images after 20 attack rounds. Our attack achieves perfect reconstruction for 2048 data points, while the baseline recovers less than 5% of the samples. Figure 4b shows the effect of the number communication rounds for $n = 2048$. Our attack reaches perfect reconstruction after 20 rounds, while CAH's performance remains below 5% of accuracy. Finally, Figure 5 provides a comprehensive comparison of the accuracy difference between our attack and the baseline across various batch sizes and communication rounds. CAH initially outperforms our method on image data after a single communication round. However, our attack surpasses the baseline after just two rounds, achieving a remarkable 99% accuracy advantage for $n = 2048$ after 20 rounds and confirming the same trends observed on ImageNet and HARUS.



(a) (b)

Figure 4: a) Effect of batch size on CIFAR-10 after 20 communication rounds. b) Effect of the number of rounds on CIFAR-10 on a batch containing 2048 images.

CNN To assess performance on a convolutional architecture, we test our attack on a VGG-style network. This network comprised three convolutional layers followed by two dense layers, consistent with the model architecture detailed in [Boenisch et al., 2023, Table 9]. The experiments are executed in single precision. Although we observe slight numerical instability for the largest batch sizes that prevents us from fully recover the batch, we successfully reconstruct over 98% of a 512-data point batch, demonstrating the attack's effectiveness even in a CNN setting.

Table 3 compares our attack and the baseline on CIFAR-10. While the baseline effectively recovers small batches, its accuracy degrades significantly with larger batch sizes. In contrast, our attack consistently recovers over 98% of the batch, even at larger sizes. Small reconstruction errors are likely due to numerical precision issues.

Table 3: Effect of of batch size and number of communication rounds. Each value indicates the percentage of data exactly recovered by attacks on CIFAR-10 dataset.

# Rounds	**Batch size**					
		32	64	128	256	512
1	Ours	32.29±11.83	13.54±3.93	5.47±1.56	2.86±1.13	0.65±0.30
	CAH	66.67±6.51	56.77±5.02	45.31±5.12	23.57±0.45	6.71±1.57
2	Ours	95.83±7.22	91.15±2.39	85.42±1.80	69.27±3.13	46.88±2.82
	CAH	70.83±4.77	64.58±3.61	53.12±4.88	34.11±3.03	11.00±2.14
5	Ours	100.00±0.00	100.00±0.00	100.00±0.00	98.31±2.29	98.50±0.98
	CAH	80.21±1.80	70.83±2.39	60.68±3.25	44.40±3.61	20.38±2.36
10	Ours	100.00±0.00	100.00±0.00	100.00±0.00	98.31±2.29	98.50±0.98
	CAH	83.33±1.80	75.00±2.71	65.89±2.26	51.82±0.81	27.34±2.49
20	Ours	100.00±0.00	100.00±0.00	100.00±0.00	98.31±2.29	98.50±0.98
	CAH	87.50±3.13	78.65±2.39	71.09±2.07	57.42±1.35	34.77±2.17

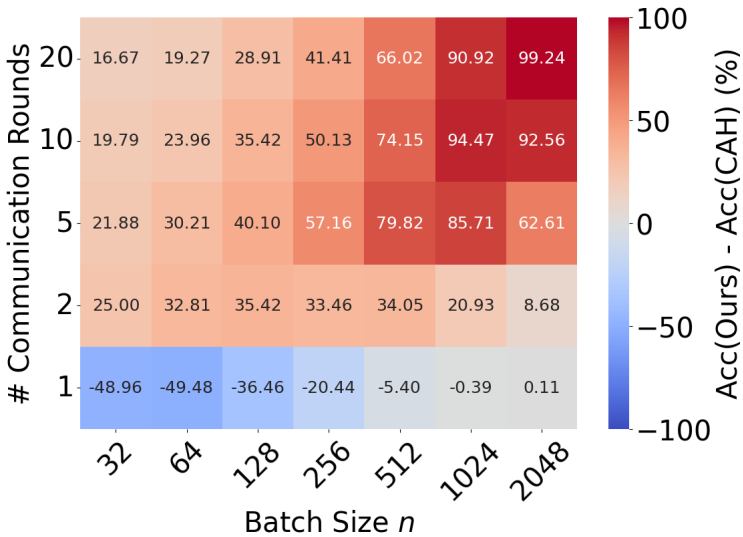


Figure 5: Percentage difference in reconstructed inputs on CIFAR-10. Red indicates cases where our attack achieves better reconstruction, while blue highlights instances where the baseline attack performs better.

Multiple Local Steps In Sec 5.1 we assumed updates based on full-batch gradient computations. Here, we extend our attack to the setting in which clients perform $K > 1$ local gradient steps. As discussed in Sec. 7, performing multiple local updates modifies the hyperplanes directions and positions, since $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ are updated at each step. This complicates the precise isolation of individual data points in the local dataset of the attacked client. To mitigate this effect, we propose a strategy that limits the influence of local updates. In particular, we aim to keep the gradients $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}$ small compared to the magnitude of $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$, thereby ensuring that parameter updates remain minimal over multiple steps. We consider a two-layer fully connected neural network, as described in the main paper. From (10) we observe that we can effectively limit the magnitude of the gradients by carefully designing the output weights. Specifically, by assigning a large value β to half of the elements in $\mathbf{W}_i^{(2)}$ and a slightly smaller value $\beta - \eta$ to the other half (with η being a small positive constant), the resulting gradients with respect to the bias $b_i^{(1)}$ and the input weights $\mathbf{W}_i^{(1)}$ remain small—approximately bounded by η and $\eta \mathbf{x}_j$, respectively. In our implementation, we use a vector $\mathbf{v} \in \mathbb{R}^C$, where the first half of its components are set to 10^5, and the second half to $10^5 - \eta$ with $\eta = 10^{-2}$. The vector is then repeated across all the columns in $\mathbf{W}^{(2)}$. To avoid gradient cancellation effects (i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}} = 0$) in case the training batch contains an equal number of samples from each half of the classes, we add small Gaussian noise $\mathcal{N}(0, 10^{-4})$ to each component of $\mathbf{v}$.

In parallel, we also aim to minimize updates to $\mathbf{W}^{(2)}$ itself across local steps. By application of the chain rule, we have $\frac{\partial \mathcal{L}_j}{\partial \mathbf{W}^{(2)}} = \frac{\partial \mathcal{L}_j}{\partial \mathbf{z}^{(2)}} \frac{\partial \mathbf{z}^{(2)}}{\partial \mathbf{W}^{(2)}}$, where the first term is given by (12), and the second term is $\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x}_j + \mathbf{b}^{(1)}$. As the loss

gradient $\frac{\partial \mathcal{L}j}{\partial \mathbf{z}^{(2)}}$ depends on classification prediction, and $\mathbf{b}^{(1)}$ is modified to perform the binary search (Alg.3), we can only directly control $\mathbf{W}^{(1)}$. Assuming that adjacent pixels in images have similar values, we set weights with alternating signs to drive the dot product $\mathbf{W}_i^{(1)}\mathbf{x}_j$ to be close to zero. Specifically, we sample the even elements of $\mathbf{W}_i^{(1)}$ from $U(1,2)$ and for odd ones as $\mathbf{W}_i^{(1)} \sim U(-2,-1)$.

Due to the presence of local updates, it is no longer sufficient to determine whether two consecutive hyperplanes isolate a point by directly comparing their corresponding observations for exact equality. Instead, we assess whether the observations are sufficiently similar by projecting the observation associated with the larger bias value onto the other, and computing the Euclidean norm of the resulting projection vector. If this value is smaller than 10^{-4}, we conclude that the corresponding hyperplanes do not separate any image.

In our experiment, we assume that the target client holds $|S| = 1024$ images—here, $|S|$ denotes the dataset size of the client, which differs from the training batch size n. We set the number of local steps to $K = |S|/n$, with $n = \{2^3, 2^4, ..., 2^{10}\}$, and use a learning rate of 10^{-4}, a standard choice for training on CIFAR10 to ensure convergence.

The results in Figure 6 demonstrate that increasing the number of local steps makes the attack more challenging; however, our method is still able to recover up to 35% of the images even with local steps. In contrast, the baseline attack (CAH) fails to recover more than 20% of the images under the same setting. It is also worth noting that in our setup we considered more local steps K and larger dataset size than what considered in the baseline Boenisch et al. [2023],as their original setting involves $K = 5$ local steps and a smaller dataset size ($|S| = 200$).

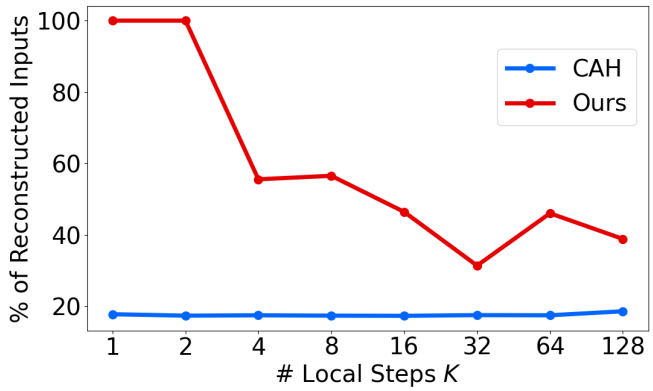


Figure 6: The percentage of perfectly reconstructed inputs for a two-layer FC NN with $N = 1000$ neurons in the first layer. The reconstruction is evaluated after 50 communication rounds.

Robustness to Noise To assess the robustness of our attack, we conduct experiments in the presence of noise, which may arise from imperfections in real-world communication channels or from defense mechanisms implemented by the client to defend against privacy attacks, such as local differential privacy[Dwork, 2006].

To improve robustness, the server can design model weights similarly to the strategy proposed to adapt to multiple local steps (check App. F.1), but with a different objective. In that case, the server modifies the classification weights $\mathbf{W}^{(2)}$ to get a small value for $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(1)}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(1)}}$, while here the objective is the opposite: the attacker needs to obtain gradients of large magnitude, so that noise can be filtered out and will not have an impact on the reconstruction quality.

Hence, we define a vector $\mathbf{v} \in \mathbb{R}^C$ by setting the first half of its components to 10^5 and the second half to $10^5 - \eta$, where $\eta = 10^3$. This vector is then replicated across all columns of $\mathbf{W}^{(2)}$.

Figure 7 illustrates the attack performance across different numbers of rounds and varying Gaussian noise variances. The client holds 1024 images, trains using FedSGD with full-batch updates, and uses a learning rate of 10^{-4}. As expected, the attack accuracy decreases with increasing noise variance; however, our method remains substantially more effective than the baseline (CAH). While our attack performance degrades gracefully as more noise is added on the client side, CAH is unable to reconstruct any sample for both values of σ considered (the two corresponding curves overlap). As a consequence, our attack consistently outperforms the baseline, recovering almost 40% more images.

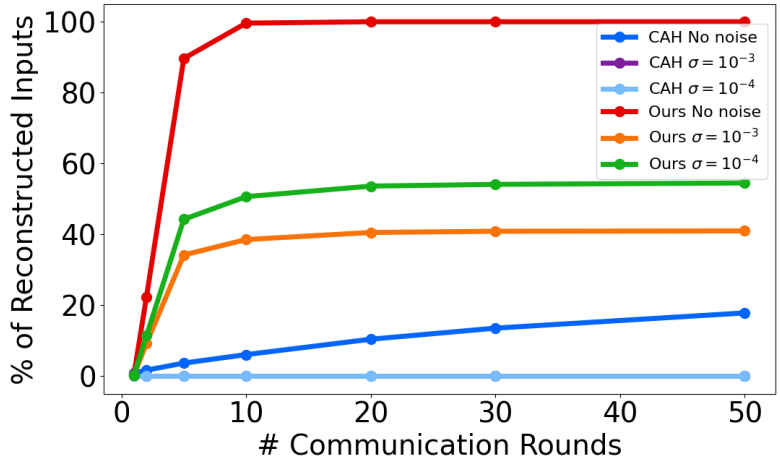


Figure 7: The percentage of perfectly reconstructed inputs on for two-layer fully connected neural network with $N = 1000$ neurons in the first layer, under varying numbers of attack rounds and different noise variances σ.

F.2 HARUS

We investigate the effect of numerical precision by performing additional HARUS experiments in single precision. Table 4 shows the results of both our attack and the baseline. The performance of our attack are comparable with the ones presented in the main paper, showing that the attack's success does not depend on double precision. However, also in this case precision errors can prevent us from recovering the full batch, as illustrated by the case of 1024 samples where extending the attack from 20 to 50 rounds does not improve the reconstruction rate.

Table 4: Impact of batch size and number of communication rounds. Each value indicates the percentage of data exactly recovered in attacks on HARUS dataset.

# Rounds		**Batch size**							
		32	64	128	256	512	1024	2048	4096
1	Ours	33.33±22.17	13.54±6.31	5.47±2.34	1.69±0.81	1.24±0.96	0.72±0.91	0.29±0.27	0.15±0.06
	CAH	51.04±4.77	23.96±1.80	10.16±0.78	4.30±1.41	1.69±0.41	0.55±0.06	0.20±0.08	0.07±0.04
2	Ours	97.92±3.61	95.83±4.77	86.20±4.71	71.09±4.75	48.96±6.88	25.72±6.31	9.47±1.50	2.90±0.60
	CAH	73.96±9.55	39.06±3.13	17.19±3.91	7.29±2.35	3.45±0.92	1.11±0.30	0.42±0.15	0.12±0.02
5	Ours	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	97.01±0.60	87.73±2.99	62.50±3.09	32.10±2.12
	CAH	93.75±3.13	69.27±5.02	31.51±0.45	14.58±0.81	6.77±0.11	2.83±0.35	1.14±0.12	0.33±0.06
10	Ours	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	99.87±0.23	99.41±0.34	92.95±0.79	70.61±2.26
	CAH	96.88±0.00	82.81±2.71	45.31±1.56	22.66±2.17	10.81±1.08	4.13±0.50	1.71±0.13	0.62±0.01
20	Ours	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.87±0.23	99.93±0.11	99.87±0.06	94.48±0.59
	CAH	100.00±0.00	94.27±0.90	60.94±2.07	32.81±1.95	16.15±2.29	6.87±1.41	2.99±0.31	1.04±0.05
30	Ours	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.87±0.23	99.93±0.11	99.93±0.06	99.04±0.14
	CAH	100.00±0.00	96.88±1.56	69.79±0.45	39.84±1.41	20.51±1.19	9.15±1.48	4.04±0.10	1.38±0.08
50	Ours	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.87±0.23	99.93±0.11	99.93±0.05	99.90±0.13
	CAH	100.00±0.00	97.92±2.39	79.43±3.16	50.78±2.03	28.52±1.19	13.41±1.94	6.01±0.42	2.23±0.10

F.3 IMAGENET

Figure 8 evaluates the difference in reconstruction accuracy between our proposed attack and the baseline across varying batch sizes and communication rounds on ImageNet, confirming our attack's superior performance over CAH in all multi-round scenarios. The prevalence of red, especially in the upper rows and to the right, clearly demonstrates that our attack generally outperforms the baseline in most settings. This highlights the effectiveness of our approach in leveraging multiple communication rounds and larger batch sizes to improve reconstruction accuracy. The blue cells at the bottom of

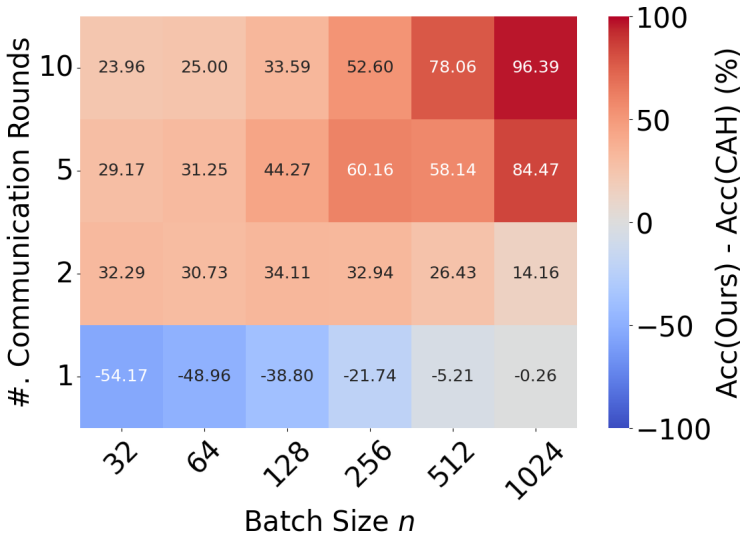


Figure 8: Percentage difference in reconstructed inputs on ImageNet. Red indicates cases where our attack achieves better reconstruction, while blue highlights instances where the baseline attack performs better.

the heatmap, corresponding to the first communication round, suggest that CAH might have an advantage in single-round attacks, particularly for smaller batch sizes, where the baseline benefits from the multiple-directions initialization. Overall, results confirm the same trends as those on HARUS in the main paper.

Non-IID setting We evaluated the performance of our attack under both IID and non-IID data distributions. Table 5 reports results on the ImageNet dataset using a single full-batch update, across different batch sizes and numbers of attack rounds.

In the IID setting, each client samples images uniformly at random from all 1,000 available classes. To simulate a non-IID scenario, we assign each client i a distinct subset of classes $\mathcal{C}_i$ such that $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, for any $i \neq j$. The result is that the local datasets of two clients do not share any common label. Each local dataset S_i is constructed by sequentially sampling 50 images per class from $\mathcal{C}_i$ until the desired dataset size is reached. Our results indicate that the attack remains highly effective in both settings. In particular, the attacker successfully reconstructs over 99% of the dataset within 5, 10, and 20 attack rounds for batch sizes of 256, 512, and 1024, respectively. While performance in the non-IID setting is occasionally slightly worse than in the IID case, this difference is marginal. One possible explanation lies in the spatial distribution of the data: the expected distance between two images randomly sampled from a subset of classes is smaller than the expected distance between two images sampled from the full set of classes. As a result, from a geometrical perspective, it becomes more challenging to position hyperplanes that isolate a single image, as Δ in (27) will be smaller than in the IID case.

Table 5: Effect of client data distribution. Each value indicates the percentage of data exactly recovered in attacks on ImageNet dataset.

# Rounds	**Batch size**						
		32	64	128	256	512	1024
2	non-IID	93.75±6.25	90.63±1.56	82.29±1.97	63.80±3.16	37.43±2.85	16.34±1.64
	IID	96.88±3.13	93.75±4.13	82.81±5.90	63.80±4.82	36.00±2.67	14.78±0.88
5	non-IID	100.00±0.00	100.00±0.00	100.00±0.00	99.74±0.45	95.38±1.30	81.64±2.71
	IID	100.00±0.00	100.00±0.00	100.00±0.00	99.74±0.45	97.27±1.74	86.10±0.73
10	non-IID	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.48±0.23	94.95±0.45
	IID	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.71±0.17
20	non-IID	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.87±0.23	99.32±0.10
	IID	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.84±0.06

Single precision We also perform single-precision ImageNet experiments, see Table 6. Consistently with the HARUS results, single precision does not significantly limit performance, and we achieve near-complete batch recovery in most settings. Minor reconstruction errors can be attributed to numerical errors.

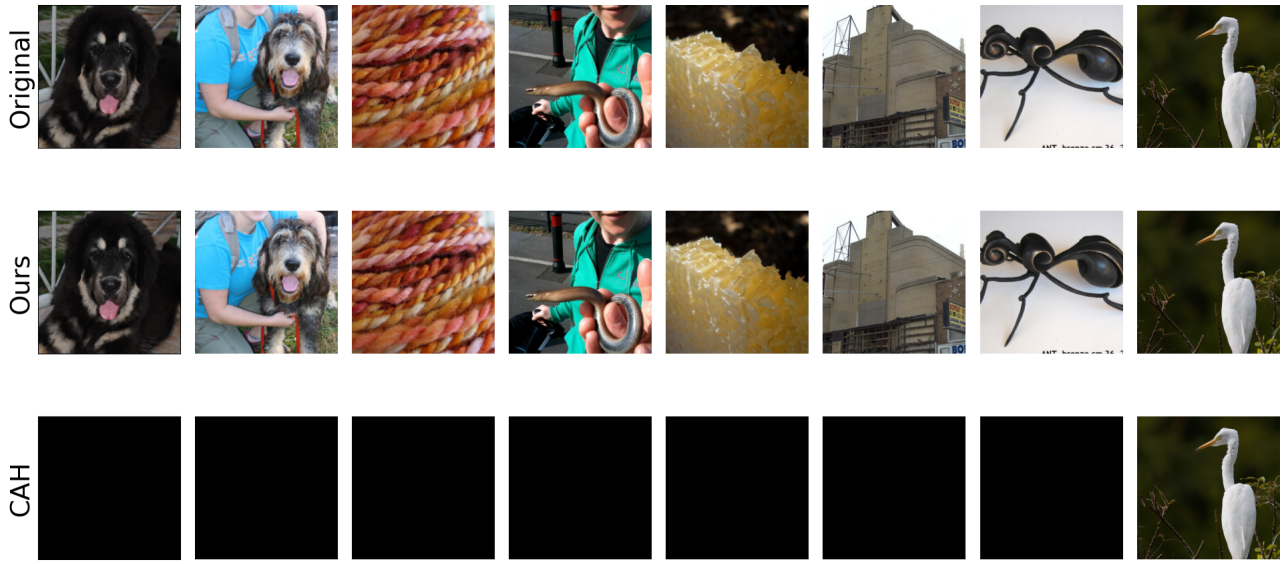


Figure 9: The recovered images from a batch containing 1024 samples from ImageNet. The first row shows the original images, the second row the recovery of our attack after 10 rounds, and the last row shows that the baseline is not able to recover most of the images.

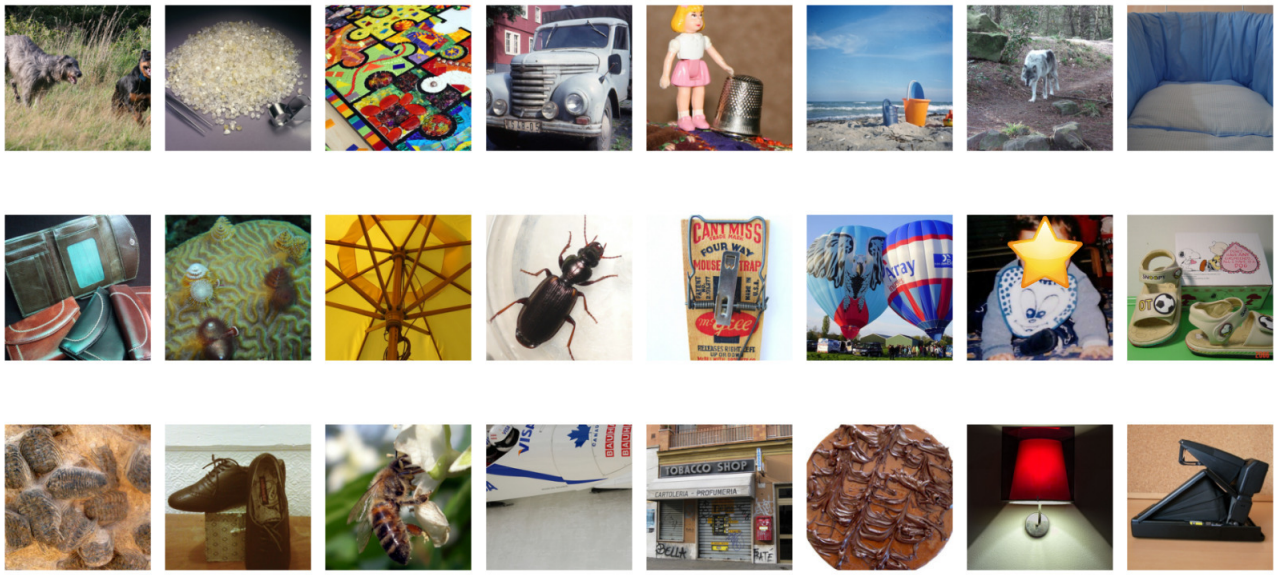


Figure 10: The recovered images from a batch containing 1024 samples from ImageNet.

Table 6: Effect of batch size and number of communication rounds. Each value indicates the percentage of data exactly recovered in attacks on ImageNet dataset.

# Rounds	**Batch size**						
		32	64	128	256	512	1024
1	Ours	3.13±0.00	2.60±0.90	0.78±0.78	0.13±0.23	0.20±0.00	0.10±0.00
	CAH	56.25±8.27	51.56±0.00	39.58±1.80	21.88±3.13	5.40±2.28	0.39±0.35
2	Ours	96.88±3.13	93.75±4.13	82.81±5.90	63.80±4.82	36.00±2.67	14.78±0.88
	CAH	63.54±4.77	59.38±3.13	47.14±0.90	29.69±3.34	9.18±3.53	0.78±0.64
5	Ours	100.00±0.00	100.00±0.00	100.00±0.00	99.74±0.45	97.27±1.74	86.10±0.73
	CAH	70.83±1.80	68.75±1.56	55.73±0.90	39.58±0.60	15.95±4.40	1.82±0.64
10	Ours	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.71±0.17
	CAH	76.04±4.77	75.00±3.13	66.41±3.41	47.40±0.60	21.94±5.38	3.32±1.28