NO MORE GUESSING: A VERIFIABLE GRADIENT INVERSION ATTACK IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing gradient inversion attacks in federated learning can fail to disentangle aggregated batches and generally lack an intrinsic mechanism to certify reconstruction success. In vision and language domains, an attacker may rely on human inspection to assess reconstruction plausibility, but this can be costly and may limit attack efficacy and scalability under tight time constraints. For numerical tabular records, however, human inspection provides no comparable means of assessing reconstruction plausibility. We propose a verifiable gradient inversion attack (VGIA) that provides explicit certificates for record isolation and reconstruction. VGIA exploits the geometry of fully connected layers by interpreting activation boundaries as hyperplanes that partition the input space. It then applies an algebraic subspace test to detect when a hyperplane-delimited region contains exactly one record. Once a single record has been certified, our method analytically recovers the input features and uses lightweight optimization to reconstruct targets. Certified regions are then removed from further exploration, allowing VGIA to adaptively allocate subsequent hyperplane queries to unresolved regions. Experiments on tabular and image benchmarks demonstrate exact recovery under large batch sizes in regimes where prior methods fail or cannot assess fidelity. Thanks to its adaptive search strategy, VGIA also requires fewer attack rounds than previous approaches.

1 INTRODUCTION

Federated Learning (FL) (McMahan et al., 2017) is a framework that enables distributed training of machine learning models without transmission of private data from clients to a central server. By design, FL limits communication to local updates, such as gradients or parameters, under the assumption that keeping the data on the client guarantees privacy. However, recent research has demonstrated that shared updates retain a significant imprint of the training data, allowing adversaries to reconstruct private inputs through Gradient Inversion Attacks (GIA) (Zhu et al., 2019; Geiping et al., 2020; Carletti et al., 2025).

Early research primarily operated under the *honest-but-curious* threat model, ignoring the risks posed by an untrusted third-party server or compromised infrastructure. To address this gap, Fowl et al. (2022) and Boenisch et al. (2023) explored the *malicious server* scenario, where the server actively manipulates model parameters to force clients' data leakage. Recent advances in this domain have focused on overcoming batch size limitations (Diana et al., 2025), improving noise robustness (Shi et al., 2025) and reducing detectability (Garov et al., 2024).

Despite substantially improving the practical effectiveness of gradient inversion, existing attacks generally lack an intrinsic way to certify that a reconstruction matches the true underlying sample. In vision and language domains, this issue is often mitigated through human inspection; for example, Fredrikson et al. (2015) rely on human evaluation to assess the recognizability of reconstructed images. However, in a malicious-server attack, the adversary may have only limited time between communication rounds to decide which reconstructions to trust and how to adapt subsequent queries, making human inspection a costly bottleneck and limiting scalability of the attack. The problem is even more acute for tabular data, which is central to many real-world federated learning applications (Upadhyay et al., 2025; Ogier du Terrail et al., 2025). Numerical records lack the semantic cues available for images or text, making erroneous reconstructions difficult to recognize. As noted

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

by Vero et al. (2023, Section 3 and Fig. 1), for an uninformed adversary it can be close to impossible to assess reconstruction quality in tabular settings, substantially complicating reliable privacy risk estimation. Figure 1 further illustrates this difficulty by showing erroneous tabular reconstructions that cannot be readily distinguished from genuine records.

To rigorously audit privacy risks in FL, we present a novel reconstruction attack under the malicious-server threat model. By crafting specific model parameters, the server isolates individual samples within an aggregated batch, while our subspace verification test certifies when this isolation has succeeded. This shows that gradient inversion attacks can be more practically dangerous than previously suggested: the attacker can identify correct reconstructions automatically, without human inspection. Once a sample is certified, our framework analytically recovers its input features and reconstructs the corresponding target value, resulting in a complete breach of the record.



Sample	Bedrooms	Sqft living	Sqft above	Sqft basement	Year built
Reconstruction	4	3215.00	2102.00	1113.00	1998
Original	4	2900.00	2900.00	0.00	2005
Original	4	3350.00	1760.00	1590.00	1995

Figure 1: Original samples and their erroneous reconstruction by CTP Diana et al. (2025). Top: images from CIFAR10 Krizhevsky (2009). Bottom: tabular records from King County Housing (KCH).

Our contributions are summarized as follows:

- We introduce VGIA, the first verifiable gradient inversion attack under a malicious server in FL. VGIA is an analytical attack that certifies whether a reconstruction is correct, establishing a more rigorous baseline for privacy auditing than heuristic validation methods.
- We empirically validate VGIA on tabular and image datasets, covering both regression and classification tasks. Our results demonstrate exact record recovery under large-batch aggregation across different data modalities. They also show that VGIA requires fewer attack rounds than previous geometric attacks.

2 Background and Related Work

Federated Learning (FL) enables a set of clients $\mathcal{C}$, coordinated by a central server, to collaboratively learn a shared global model θ. The training objective is to minimize the weighted average of the clients' empirical risks:

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

where $\mathcal{D}^c = \{(\mathbf{x}_i^c, y_i^c)\}_{i=1}^{n^c}$ denotes the local dataset of client c and $p^c > 0$ is a client weight satisfying $\sum_{c \in \mathcal{C}} p^c = 1$, typically chosen proportional to the dataset size or set to $1/|\mathcal{C}|$ for all clients. In each round, participating clients download the current global model, compute local updates using their data, and send these updates to the server. Depending on the algorithm, clients may perform either a single stochastic gradient update (FedSGD; Chen et al., 2016) or multiple stochastic gradient steps (FedAvg; McMahan et al., 2017).

Recent work shows that during FedSGD and FedAvg training, a server can reconstruct private client data via gradient inversion attacks (GIAs) either by passively observing client updates in the *honest-but-curious* setting (Zhu et al., 2019; Zhao et al., 2020; Li et al., 2023; Dimitrov et al., 2022; Geng et al., 2022; Ma et al., 2023; Dimitrov et al., 2024; Bakarsky et al., 2025) or by actively manipulating the global model parameters in the *malicious* setting (Fowl et al., 2022; Wen et al., 2022; Boenisch et al., 2023; Shi et al., 2025; Diana et al., 2025).

Following Carletti et al. (2025), GIAs can fall into three families: optimization-based, generative-model-based, and analytical. Below, we briefly review malicious attacks and verifiability limitations in the three approaches. Appendix A provides further details.

Optimization-based attacks were the first approach to recover client inputs from FL updates (Zhu et al., 2019). Starting from dummy inputs and labels, the attacker reconstructs data by minimizing the discrepancy between gradients computed on the dummy data and the observed gradients. Wen et al. (2022); Zhang et al. (2024); Shan et al. (2025) extend this line of work with attacks that selectively amplify per-sample gradient contributions. Generative-model-based attacks reduce the cost of direct input optimization by training a reconstruction module offline, typically using auxiliary

data (Shi et al., 2025; Garov et al., 2024). The server then uses this module to map information extracted from client updates to reconstructed inputs. Alternatively, analytical attacks exploit structural properties of gradients in fully connected layers with ReLU activations to recover *exact* inputs; we review the underlying theory in Sec. 4.1. In particular, these attacks craft ReLU activation patterns that separate individual samples from the rest of the batch. Some methods (Boenisch et al., 2023; Zhang et al., 2023b) set parameters so that certain neurons activate on a single sample, whereas others (Fowl et al., 2022; 2023; Chu et al., 2023; Zhao et al., 2024; Diana et al., 2025) use pairs of neurons whose activation sets differ by exactly one sample.

Verifiability Across these families, existing attacks may produce a large fraction of spurious reconstructions that do not correspond to any single input in the batch. This issue is often partly obscured by how GIAs are evaluated: reconstruction success is commonly assessed using reference-based metrics such as PSNR, SSIM, and LPIPS for images, or BLEU and ROUGE for text. However, passing a metric-based success threshold does not necessarily imply faithful recovery of an individual input. For example, matching an image's background may yield a high PSNR despite failing to recover the foreground object. Figure 11 illustrates such failures for an existing attack.

Some prior work has indeed documented substantial rates of spurious reconstruction: Shan et al. (2025, Figure 6) report that more than 35% of reconstructed samples are spurious for a CIFAR batch of size 256, while Boenisch et al. (2023, Table 5) report more than 48% for an ImageNet batch of size 100. Even when favorable metric values correspond to correctly reconstructed data, these scores require access to ground-truth samples and therefore do not address the need, emphasized by Vero et al. (2023), for an attacker to assess reconstruction quality without knowing the original data.

Human inspection can sometimes mitigate this issue for image or text data, but it must be performed on each tentative reconstruction, often under tight time constraints during the attack, as identifying successful reconstructions may be necessary to guide subsequent attack steps. This makes manual inspection a costly bottleneck for large batches. For tabular records, manual validation is largely infeasible, since numerical records do not provide comparable semantic cues (Fig. 1). This distinction is important: even an attack that consistently recovers a small fraction of samples, e.g., 10%, can pose a serious privacy threat, but its practical impact is reduced if the attacker cannot identify which reconstructions are correct.

Ideally, GIAs should therefore provide an intrinsic verification mechanism. Despite its practical importance, verifiability has been largely overlooked in the GIA literature; only a few works, mostly in the honest-but-curious setting, offer any form of verification, typically under restrictive conditions.

Vero et al. (2023) estimate feature-wise reconstruction uncertainty using entropy across reconstructions obtained from repeated, potentially expensive optimization runs. Their assessment is empirically validated, but does not provide formal certificates of reconstruction correctness. Analytical attacks from Dimitrov et al. (2024) and Bakarsky et al. (2025) incorporate internal reconstruction-validation mechanisms, but require the batch size to be smaller than the input dimension—a restrictive condition for tabular data. Moreover, even in high-dimensional settings that permit large batches, these honest-but-curious methods remain limited in their ability to disentangle increasingly large batches from aggregated gradients. Finally, in a malicious setting, CTP (Diana et al., 2025) relies on a stopping threshold on the minimum projected separation between inputs (see Sec. 4.1).

3 Threat model

We consider the standard malicious-server threat model for FL, where the central server can modify the model parameters sent to clients, which apply them without verification before computing and returning their updates. This setting is shared with prior malicious-server attacks (Wen et al., 2022; Fowl et al., 2022; Boenisch et al., 2023; Shi et al., 2025; Diana et al., 2025). We discuss client-side verifications in App. H.

Following Diana et al. (2025), we assume that the inputs lie in a bounded domain and that the attacker knows lower and upper bounds on each input feature after preprocessing. In other respects, our threat model grants the attacker fewer capabilities and less prior knowledge than assumed in several closely related works. First, the server is not allowed to modify the model architecture, for example by adding layers or modules (Fowl et al., 2022; 2023; Zhao et al., 2024; Shi et al., 2025),

changing internal connections or routing mechanisms (Fowl et al., 2023; Chu et al., 2023), or altering activation functions (Fowl et al., 2022). Second, we do not assume access to auxiliary or public datasets, unlike (Boenisch et al., 2023; Zhang et al., 2023b; Garov et al., 2024; Shi et al., 2025), nor knowledge of the data distribution beyond these bounds on the input features, unlike (Fowl et al., 2022; 2023; Shi et al., 2025). In the following, we focus on clients updating the global model with full-batch FedSGD. We also provide empirical results for FedAvg in App. I.

4 Our attack

4.1 Preliminaries

Analytic Gradient Inversion Our work builds upon the fundamental observation that the gradients of a fully connected (FC) layer followed by a ReLU activation can contain sufficient information to analytically recover input data (Phong et al., 2017; Geiping et al., 2020; Fowl et al., 2022).

Consider a model whose first (attacked) layer is a linear layer with N neurons, parameterized by weights $\mathbf{W}^a \in \mathbb{R}^{N\times d}$ and biases $\mathbf{b}^a \in \mathbb{R}^N$. For neuron i, the post-activation output is

$$z_i^a = \mathrm{ReLU}(\mathbf{W}_i^a \mathbf{x} + b_i^a)\,,$$

where $\mathbf{x} \in \mathbb{R}^d$ denotes the input to the attacked layer.

In FedSGD, the server receives the gradient of the batch-averaged loss $\mathcal{L} = \frac{1}{B}\sum_{j=1}^{B} \mathcal{L}_j$ over the batch $\mathcal{B} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{B}$. As shown in prior work (Zhang et al., 2023b), gradients with respect to $\mathbf{W}^a$ and $\mathbf{b}^a$ reveal an *observation* $\mathbf{o}_i$, given by a linear combination of the batch inputs:

$$\mathbf{o}_i \;=\; \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i^a}\left(\frac{\partial \mathcal{L}}{\partial b_i^a}\right)^{-1} \;=\; \sum_{j=1}^{B} \alpha_j \mathbf{x}_j, \quad \text{where} \quad \alpha_j \;=\; \frac{\frac{\partial \mathcal{L}_j}{\partial b_i^a}}{\sum_{k=1}^{B} \frac{\partial \mathcal{L}_k}{\partial b_i^a}}. \tag{2}$$

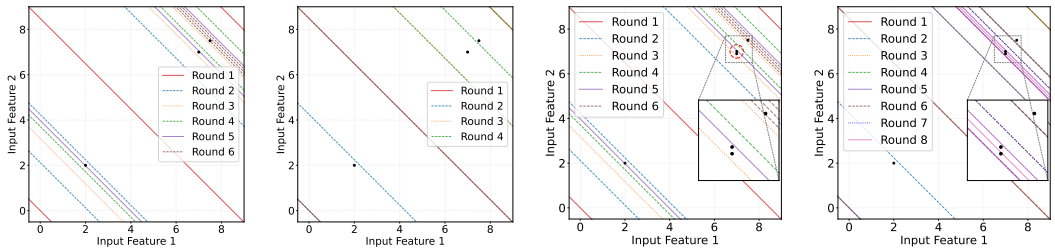
Only samples that activate neuron i, i.e., inputs $\mathbf{x}_j$ such that $z_{i,j}^a > 0$, have a nonzero coefficient α_j and therefore contribute to $\mathbf{o}_i$, where we use $z_{i,j}^a$ to denote z_i^a evaluated on $\mathbf{x}_j$. Disentangling individual samples from the aggregate $\mathbf{o}_i$ is the central challenge in analytic gradient inversion.

Geometric Isolation To address this, Diana et al. (2025) adopt a geometric view of ReLU-based leakage through the activation boundary. For neuron i, the weights $\mathbf{W}_i^a$ and bias b_i^a define the hyperplane $\mathcal{H}_i = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{W}_i^a \mathbf{x} + b_i^a = 0\}$, which partitions the batch into inputs on the ReLU-active side $\mathbf{W}_i^a \mathbf{x} + b_i^a > 0$ and on the inactive side $\mathbf{W}_i^a \mathbf{x} + b_i^a \leq 0$. Fixing $\mathbf{W}_i^a$ and varying b_i^a translates $\mathcal{H}_i$ along its normal direction, yielding a family of parallel hyperplanes that *sweep* across the batch. Building on this interpretation, Diana et al. (2025) propose the CTP (Cutting Through Privacy) algorithm, which searches iteratively over b_i^a to identify thresholds $\hat{b}_{i,1}^* < \cdots < \hat{b}_{i,B+1}^*$ such that the *slice* between two consecutive hyperplanes contains a single batch element $\mathbf{x}_k$. CTP is the state-of-the-art GIA, capable of recovering up to 1024 exact ImageNet images without requiring an auxiliary dataset or modifications to the model architecture.

CTP can identify empty slices and avoid querying them further, but it cannot decide whether a non-empty slice contains one sample or several. It therefore keeps bisecting every non-empty slice until its width is below a user-chosen threshold ε. This dependence on ε results in two limitations. First, if two inputs have projections to the normal direction of $\mathbf{W}_i^a$ closer than ε, CTP may stop before separating them and return an incorrect reconstruction. Second, if ε is chosen too small, it may waste rounds refining slices that already contain a single sample. Figure 2a illustrates a successful case, where CTP recovers three inputs after 6 rounds, using 3 hyperplanes per round and 18 hyperplanes in total; in contrast, Fig. 2c shows a failure case in which two inputs remain in the same ε-wide slice. In a realistic blind threat model, the adversary does not know the relevant sample separations in advance and therefore cannot tune ε reliably.

4.2 Verifiable Isolation and Reconstruction

VGIA, the attack we propose, addresses the two main limitations of CTP. First, it introduces an algebraic verification step that allows the attacker to certify when a slice contains a single input. Second,



(a) (b) (c) (d)

Figure 2: A two-dimensional example of the search process. Figure 2a illustrates CTP (Diana et al., 2025), while Figure 2b shows our method. In this example, our method requires fewer rounds because it stops refining slices once isolation is certified, whereas CTP continues until their width falls below ε. Conversely, when ε is too large, CTP may stop before all samples are separated, as shown in Figure 2c, whereas our method (Figure 2d) continues refining the unresolved slices until all samples are isolated.

once a slice has been certified, VGIA removes it from further refinement and reallocates future hyperplane queries to unresolved regions. This avoids unnecessary dissections of already isolated samples and reduces the number of communication rounds required for full-batch reconstruction.

Figures 2b and 2d illustrate how our attack adapts its search to the specific batch. In Figure 2b, by identifying slices that already contain a single input, it stops refining them and reconstructs all samples in only 4 rounds (12 hyperplanes), compared to 6 rounds (18 hyperplanes) for CTP (Fig. 2a). In Figure 2d, instead, it continues refining unresolved slices for 8 rounds until all inputs are isolated, whereas CTP (Fig. 2c) stops too early and produces an incorrect reconstruction.

We present the attack for an L-layer fully connected neural network with a scalar output and ReLU activation. The attacked first layer is parameterized by weights $\mathbf{W}^a \in \mathbb{R}^{N\times d}$ and biases $\mathbf{b}^a \in \mathbb{R}^N$ and is followed by a ReLU activation. The last layer is parameterized by weights $\mathbf{W}^L \in \mathbb{R}^{1\times M}$ and bias $b^L \in \mathbb{R}$, where M is the number of neurons in layer $L-1$. Extensions to multi-dimensional outputs, to FedAvg, to different activation functions such as SiLU and GELU, and to other architectures including convolutional neural networks and transformers are provided in Appendices D.1, I, J, and K, respectively. For FedAvg and smooth activations such as SiLU and GELU, our verification is approximate. For CNNs and Transformers, the guarantees apply to suitable intermediate embeddings, while reconstructing the original inputs requires a decoder, as in Shi et al. (2025).

Verifiable Isolation The first step of our attack is to extract the contribution of the inputs lying between two consecutive hyperplanes. Within a single round, the attacker can test multiple parallel hyperplanes with normal vector $\mathbf{w}$ by targeting different neurons and setting their weight vectors to $\mathbf{w}$ (e.g., $\mathbf{W}^a_i = \mathbf{W}^a_{i'} = \mathbf{w}$ when attacking neurons i and i'), while assigning them different biases (so $b^a_i \neq b^a_{i'}$). For simplicity, we assume that all neurons in the layer are targeted.

We first analyze the gradient of the loss with respect to neuron i's weights $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_i}$.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_i} = \frac{1}{B}\sum_{j=1}^{B} \frac{\partial \mathcal{L}_j}{\partial z^a_i}\,\mathbf{x}_j\,\mathbb{1}_{z^a_{i,j}>0} = \frac{1}{B}\sum_{j\in\mathcal{A}_i} \frac{\partial \mathcal{L}_j}{\partial z^a_i}\,\mathbf{x}_j = \frac{1}{B}\sum_{j\in\mathcal{A}_i} \frac{\partial \mathcal{L}_j}{\partial z^L}\frac{\partial z^L}{\partial z^a_i}\mathbf{x}_j, \tag{3}$$

where the indicator function $\mathbb{1}_{z^a_{i,j}>0}$ specifies whether neuron i is activated by sample $\mathbf{x}_j$, B is the batch size, and $\mathcal{A}_i = \{j \in [B] \mid \mathbf{W}^a_i\mathbf{x}_j + b^a_i > 0\}$ denotes the indices of the batch samples that activate neuron i.

In general, the partial derivative $\frac{\partial z^L}{\partial z^a_i}$ depends on the input $\mathbf{x}$, because downstream ReLU gates may switch on/off as $\mathbf{x}$ changes. However, the malicious server can choose the parameters of the subsequent layers $(\mathbf{W}^{a+1:L}, \mathbf{b}^{a+1:L})$ so that the activation pattern of all ReLUs after layer a is fixed for all inputs of interest (see App. G). Once this is achieved, z^L becomes an affine function of z^a, and the server can ensure that the Jacobian entry $\frac{\partial z^L}{\partial z^a_i}$ is a nonzero constant, denoted by g_i.

Consider two consecutive bias values $\hat{b}_k < \hat{b}_{k+1}$ tested at two attacked neurons in the same round. Without loss of generality, assume these neurons are i and $i+1$. We call the region delimited by the two hyperplanes $(\mathbf{w}, \hat{b}_k)$ and $(\mathbf{w}, \hat{b}_{k+1})$ the *k-th slice*. Dividing (3) by the constant g_i and subtracting it from the corresponding expression for neuron $i+1$, the server obtains the *slice vector* $\mathbf{s}_k$, a linear combination of the inputs lying in the slice between the two corresponding hyperplanes:

$$\mathbf{s}_k = \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_{i+1}}}{g_{i+1}} - \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_{i}}}{g_i} = \frac{1}{B} \sum_{j \in \mathcal{A}_{i+1}} \frac{\partial \mathcal{L}_j}{\partial z^L} \mathbf{x}_j - \frac{1}{B} \sum_{j \in \mathcal{A}_i} \frac{\partial \mathcal{L}_j}{\partial z^L} \mathbf{x}_j = \frac{1}{B} \sum_{j \in \mathcal{A}_{i+1} \setminus \mathcal{A}_i} \frac{\partial \mathcal{L}_j}{\partial z^L} \mathbf{x}_j. \quad (4)$$

This quantity is computable by the server because it observes the client gradient and knows the constants g_i and g_{i+1}. When the interval between two consecutive hyperplanes contains no data point, the active sets $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ coincide, and the corresponding slice vector is zero. Such empty slices can be discarded: in subsequent rounds, the server places new hyperplanes only inside intervals with nonzero slice vectors, thereby pruning the search space.

The key remaining question is whether a nonzero slice vector corresponds to a single input or to a combination of several inputs. To answer this question, the attacker refines a non-empty slice by placing $q \geq 3$ hyperplanes across it. Formally, consider a non-empty slice at round t, delimited by the bias values $\hat{b}^t_k$ and $\hat{b}^t_{k+1}$, with slice vector $\mathbf{s}^t_k$. At round $t+1$, the attacker places q bias values in $[\hat{b}^t_k, \hat{b}^t_{k+1}]$, such that $\hat{b}^{t+1}_k = \hat{b}^t_k < \cdots < \hat{b}^{t+1}_{k+q-1} = \hat{b}^t_{k+1}$, and computes the resulting $q-1$ sub-slice vectors $\mathbf{s}^{t+1}_k, \ldots, \mathbf{s}^{t+1}_{k+q-2}$. The following proposition, whose full statement and proof are given in App. B, provides a criterion for certifying that all samples in the parent slice have been isolated.

Proposition 4.1 (Verifiable isolation)**.** *Suppose that the slice at round t contains $m \leq d$ linearly independent inputs. Assume absolutely continuous sampling of $\mathbf{w}$ and the downstream parameters, with the latter independently resampled across rounds while preserving the input-independent gating described above. Assume also that all level sets of $\partial \mathcal{L}_j / \partial z^L$ are Lebesgue-null. If*

$$\mathbf{s}^t_k \in \operatorname{span}\left(\mathbf{s}^{t+1}_k, \ldots, \mathbf{s}^{t+1}_{k+q-2}\right),$$

then, almost surely, every non-empty sub-slice contains exactly one input. Consequently, the original slice contains as many inputs as there are non-null sub-slice vectors, and hence at most $q-1$ inputs.

Note that linear independence is required only within each slice, not across the full batch, which may therefore exceed d. As discussed in App. B.1, this is a mild local assumption since slices typically contain few samples; even undetected violations need not cause reconstruction errors. We observed no such violations in our experiments. The assumption on $\frac{\partial \mathcal{L}_j}{\partial z^L}$ holds for any loss whose derivative varies with the activation values, such as the logistic, exponential, focal, and squared losses. Proposition 4.1 tells the attacker whether the slice must be further explored with additional hyperplanes, or—when the condition holds—whether all samples in the slice have been correctly separated and can therefore be reconstructed using the procedure described below.

Reconstruction Suppose that the slice vector $\mathbf{s}^{t+1}_k$, obtained from the two hyperplanes tested at neurons i and $i+1$, contains a single input $\mathbf{x}_j$. From (4), we have $\mathbf{s}^{t+1}_k = \beta^{t+1}_j \mathbf{x}_j$, where $\beta^{t+1}_j := \frac{1}{B} \frac{\partial \mathcal{L}_j}{\partial z^L}$. Recovering $\mathbf{x}_j$ therefore only requires estimating the scalar β^{t+1}_j. This coefficient can be obtained from the gradients with respect to the attacked-layer biases. Expanding $\frac{\partial \mathcal{L}}{\partial b^a_i}$ as in (3) and applying the same rescaling and subtraction as in (4) gives

$$\frac{\frac{\partial \mathcal{L}}{\partial b^a_{i+1}}}{g_{i+1}} - \frac{\frac{\partial \mathcal{L}}{\partial b^a_{i}}}{g_i} = \frac{1}{B} \sum_{\ell \in \mathcal{A}_{i+1}} \frac{\partial \mathcal{L}_\ell}{\partial z^L} - \frac{1}{B} \sum_{\ell \in \mathcal{A}_i} \frac{\partial \mathcal{L}_\ell}{\partial z^L} = \frac{1}{B} \sum_{\ell \in \mathcal{A}_{i+1} \setminus \mathcal{A}_i} \frac{\partial \mathcal{L}_\ell}{\partial z^L} = \frac{1}{B} \frac{\partial \mathcal{L}_j}{\partial z^L} = \beta^{t+1}_j. \quad (5)$$

Using (4) and (5), the attacker therefore recovers the private input exactly as $\hat{\mathbf{x}}_j = \mathbf{s}^{t+1}_k / \beta^{t+1}_j = \mathbf{x}_j$. Finally, the corresponding target can be reconstructed with an optimization-based procedure, detailed in App. D.2.

4.3 Final Algorithm

Algorithm 1 presents the full VGIA procedure. As assumed in Section 3, the attacker knows bounds on the input features. These bounds induce an initial interval $[l, u]$ for the projections $\mathbf{wx}$, and therefore for the bias values to be queried. The procedure SetHyperplanes (Algorithm 2) constructs

Algorithm 1 Verifiable Gradient Inversion Attack (VGIA)

Input: $[l, u]$ initial search space of bias values, total attack rounds T

$\mathcal{I} \leftarrow (([l, u],))$ // $\mathcal{I}$ is used to track the searched slices and corresponding slice vectors
Draw a vector $\mathbf{w} \in \mathbb{R}^d$ and set each row of $\mathbf{W}^a$ to $\mathbf{w}$.
for $t \in \{1, ..., T\}$ **do**
  $(\mathbf{b}^a, \mathcal{I}) \leftarrow$ SETHYPERPLANES$(\mathcal{I})$ // Set malicious bias values w.r.t. $\mathcal{I}$
  Server samples downstream layers (G.1) and sends malicious parameters θ to the client
  Server receives gradient updates $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}^a}$
  Server computes $\mathbf{s}_k^t$ using (4) and β_k^t using (5) for $k \in \{1, 2, ..., N-1\}$
  $\mathcal{S} \leftarrow ((b_1^a, b_2^a, \mathbf{s}_1^t, \beta_1^t), \ldots, (b_{N-1}^a, b_N^a, \mathbf{s}_{N-1}^t, \beta_{N-1}^t))$ // Collect all the sub-slices
  $(\mathcal{I}, \mathcal{R}) \leftarrow$ CHECKISOLATION$(\mathcal{I}, \mathcal{S})$ // Verify slices by Prop. 4.1, put verified ones in $\mathcal{R}$
  If $\mathcal{R} \neq \emptyset$: Server reconstructs the inputs and targets in $\mathcal{R}$ from (4), (5), and (16)
  If $\mathcal{I} = \emptyset$: break

the N hyperplanes queried by the attacker, one per attacked neuron. In the first communication round, the biases associated with these hyperplanes partition $[l, u]$ into $N-1$ slices. From the second round onward, each selected interval is queried with at least three hyperplanes, including its two boundary hyperplanes and at least one new interior hyperplane. The boundary hyperplanes must be queried again at each round to compute the new slice vectors, because, unlike CTP (Diana et al., 2025), the parameters $\mathbf{W}^{a+1:L}$ need not remain fixed and are updated at every round to make the attack less detectable (App. G.1). Consequently, the gradients associated with the same neuron in (3) may vary across rounds. The procedure CHECKISOLATION (Algorithm 3 in App. D.3) uses Proposition 4.1 to certify when a slice contains exactly one sample. Certified slices are added to $\mathcal{R}$ and are ready for reconstruction. The same procedure also detects empty slices through a null slice vector, in which case the corresponding interval is discarded.

Round Complexity Under the same assumptions as Proposition 4.1, if $N \geq 3B$, our algorithm can verify all inputs in a batch with high probability in at most $3 \cdot \left\lceil \log_2 \left(\frac{W}{N \cdot \varepsilon_{\mathbf{w}}} \right) \right\rceil + 1$ rounds, where W is the width of the initial search interval and $\varepsilon_{\mathbf{w}}$ is the minimum projected separation of samples in $\mathcal{D}$ along direction $\mathbf{w}$, i.e., $\varepsilon_{\mathbf{w}} = \min_{\mathbf{x}_1 \neq \mathbf{x}_2 \in \mathcal{D}} \left| \mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) \right|$. The proof is in App. C.

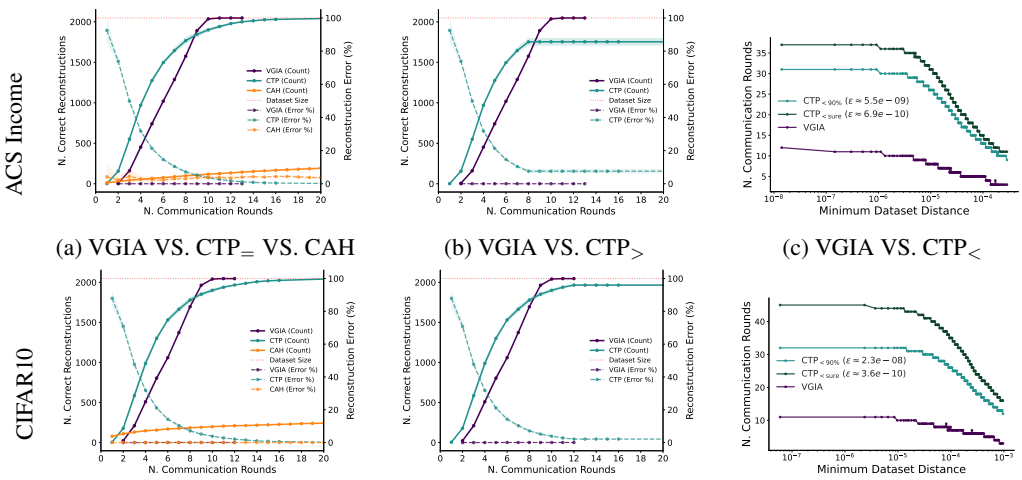
Algorithm 2 SetHyperplanes

Input: search intervals $\mathcal{I} = (([l_1, u_1], \mathbf{s}_1), ..., ([l_{|\mathcal{I}|}, u_{|\mathcal{I}|}], \mathbf{s}_{|\mathcal{I}|}))$

sort $\mathcal{I}$ so that $l_1 < l_2 < \cdots < l_{|\mathcal{I}|}$
$M \leftarrow \min(|\mathcal{I}|, \lfloor N/3 \rfloor)$; $r \leftarrow N \bmod M$; $j \leftarrow 1$
for $k = 1, \ldots, M$ **do**
  $q \leftarrow \lfloor N/M \rfloor + \mathbb{1}_{k \leq r}$
  $\mathbf{b}^a_{j:j+q-1} \leftarrow \begin{cases} \{l_k + i \frac{u_k - l_k}{q} | i = 1, \ldots, q\} & \textbf{if } k > 1 \textbf{ and } l_k = u_{k-1} \\ \{l_k + i \frac{u_k - l_k}{q-1} | i = 0, \ldots, q-1\} & \textbf{otherwise} \end{cases}$
  $j \leftarrow j + q$
Return $\mathbf{b}^a, \mathcal{I}$

5 EXPERIMENTAL EVALUATION

Setup We evaluate the attacks on five benchmarks: three tabular datasets—ACS Income (Ding et al., 2021) and King County Housing (KCH) for regression, and Human Activity Recognition Using Smartphones (HARUS) for classification—and two image datasets, CIFAR10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). Dataset details are provided in App. E.1. Each client trains a three-layer fully connected neural network (FC-NN) with ReLU activations, using $N = 1000$ neurons in the first hidden layer and 100 neurons in the second hidden layer. Unless otherwise stated, clients update the model with full-batch FedSGD. Due to space constraints, we report only the results on ACS Income and CIFAR10 in the main text, and defer additional experiments on the remaining datasets to App. F.



(a) VGIA VS. CTP$_=$ VS. CAH (b) VGIA VS. CTP$_>$ (c) VGIA VS. CTP$_<$

(d) VGIA VS. CTP$_=$ VS. CAH (e) VGIA VS. CTP$_>$ (f) VGIA VS. CTP$_<$

Figure 3: Attack performance of VGIA, CAH, and CTP when the target client trains a three-layer fully connected neural network, using full-batch FedSGD, on a local dataset of $B = 2048$ samples from ACS Income (top) or CIFAR10 (bottom). (a,b,d,e) Number of correct samples and reconstruction error under varying attack-round budgets. (c,f) Number of rounds required for the adversary to identify all samples (i.e., achieve verifiability) across target datasets with different values of $\varepsilon_{\mathbf{w}}$. CTP$_=$ assumes knowledge of $\varepsilon_{\mathbf{w}}$. CTP$_>$ corresponds to an overly optimistic choice of $\varepsilon = 10^{-4} > \varepsilon_{\mathbf{w}}$. CTP$_<$ corresponds to a conservative choice, with ε such that $\varepsilon < \varepsilon_{\mathbf{w}}$.

Baselines We compare VGIA against CAH (Boenisch et al., 2023) and CTP (Diana et al., 2025), the state-of-the-art attack under the malicious threat model described in Sec. 3. We also compare against Scale-MIA (Shi et al., 2025) on ResNet and ViT, adapting VGIA to a stronger threat model that allows architecture modifications and access to auxiliary data (App. K). As detailed in Section 4.1, CTP stops its search when the slice width falls below a user-chosen threshold ε. Let $\mathcal{D}$ be the attacked client's local dataset, and remember that $\varepsilon_{\mathbf{w}}$ denotes the minimum projected separation between any two distinct samples in $\mathcal{D}$ along direction $\mathbf{w}$. We evaluate CTP in three regimes: $\varepsilon < \varepsilon_{\mathbf{w}}$, $\varepsilon = \varepsilon_{\mathbf{w}}$, and $\varepsilon > \varepsilon_{\mathbf{w}}$, denoted by CTP$_<$, CTP$_=$, and CTP$_>$, respectively. We run each method over three random seeds. For each seed, we sample a Gaussian vector $\mathbf{w}$ defining the direction of the queried hyperplanes. For a fair comparison, the three attacks use the same $\mathcal{D}$. Detailed attack configurations are provided in App. E.2.

Metrics Since the attack can yield correct reconstructions, spurious reconstructions, or no reconstruction, two metrics suffice to characterize performance at a fixed attack budget: the number of correct reconstructions and the proportion of spurious ones among all reconstructions, which we call reconstruction error. A sample is considered correctly reconstructed if its ℓ_2 distance to the corresponding ground-truth sample is below 10^{-4} and, for images, its SSIM is larger than 0.99. Although spurious reconstructions may in principle arise from numerical errors, and thus also affect VGIA, all erroneous reconstructions observed in the reported experiments are due to combinations of samples that have not been properly isolated. VGIA provides a provable certificate of reconstruction correctness through Proposition 4.1. CTP, instead, treats a sample as isolated once it lies in a slice of width at most ε; this criterion is sound only when $\varepsilon \leq \varepsilon_{\mathbf{w}}$, namely for CTP$_<$ and CTP$_=$. Finally, CAH has no intrinsic verification procedure and can only rely on human inspection.

Attack performance We report results on ACS Income (Figs. 3a and 3b) and CIFAR10 (Figs. 3d and 3e) for a batch size of 2048 under different attack-round budgets. On both datasets, VGIA recovers and certifies all 2048 samples within 12 rounds, whereas CTP$_=$, the most favorable setting for CTP, requires 20 rounds (Figs. 3a and 3d). Although CTP recovers more correct samples than VGIA for budgets between 2 and 8 rounds, the attacker cannot distinguish true samples from spurious reconstructions. For instance, on Income, after 4 rounds, CTP has reconstructed nearly 1000 correct samples, but also produced about 430 spurious reconstructions, corresponding to a recon-

struction error of approximately 30%. After 8 rounds, CTP recovers 1750 correct samples but still produces 120 spurious samples. In contrast, VGIA produces no spurious reconstructions throughout this experiment, empirically demonstrating the robustness of our verification procedure even in the presence of numerical errors. Reconstruction examples are reported in App. L.

In Figs. 3b and 3e, we consider $\text{CTP}_{>}$, where $\varepsilon = 10^{-4} > \varepsilon_{\mathbf{w}}$, i.e., the stopping threshold overestimates the true projected separation. In this setting, CTP fails to recover all samples correctly, regardless of the attack-round budget. For example, on ACS Income, even with large attack budgets, approximately 8% of the reconstructions are spurious, while still passing CTP's heuristic verification criterion. This issue is even more pronounced on King County Housing, where samples are closer to each other: VGIA still recovers all samples within 12 rounds, whereas $\text{CTP}_{>}$ recovers at most 1000 samples and produces approximately 420 spurious reconstructions; see Fig. 5b in the Appendix.

Verifiability study In Figs. 3c and 3f, we evaluate how many rounds each attack requires to verify all reconstructed samples, across different target client datasets sampled from the full dataset $\mathcal{S}$. The ground-truth projected separation $\varepsilon_{\mathbf{w}}$ ranges from $\varepsilon_{\min} = 6.9\times10^{-10}$ to 2.9×10^{-5} for ACS Income and from $\varepsilon_{\min} = 3.6 \times 10^{-10}$ to 9.6×10^{-4} for CIFAR10. CTP can successfully verify all samples only when $\varepsilon \leq \varepsilon_{\mathbf{w}}$. In practice, the attacker does not know $\varepsilon_{\mathbf{w}}$, which depends on the target dataset. In this experiment, however, we give CTP access to the CDF of $\varepsilon_{\mathbf{w}}$ and consider two choices: (i) $\varepsilon = \varepsilon_{\min}$, which guarantees correct verifiability for any dataset drawn from $\mathcal{S}$ ($\text{CTP}_{<\text{sure}}$); and (ii) ε set to the (0.1)-quantile of the CDF, which yields a 90% probability of correct verifiability ($\text{CTP}_{<90\%}$). The figures show that both $\text{CTP}_{<\text{sure}}$ and $\text{CTP}_{<90\%}$ require at least twice the attack budget of VGIA to achieve verification across all sampled target datasets. The same observation holds for the HARUS, KCH and ImageNet datasets (see Figs. 4c, 5c, and 6c in the Appendix).

Additional attack results Results in App. K show that VGIA correctly verifies more than 99% of the embeddings in batches of 1024 CIFAR10 images for both ResNet18 and Transformer-based architectures; the few failures are due to numerical errors. Similar results hold for FC-NNs with SiLU/GELU activations, despite our verification procedure being approximate in this setting (see App. J). VGIA is less robust under FedAvg, but still reconstructs at least twice as many samples as $\text{CTP}_{=}$ and yields at least six times fewer spurious candidates across all tested settings (App. I).

Detectability In App. G, we show that VGIA can be made difficult to detect by both the state-of-the-art detection rules proposed by Carletti et al. (2026) and stronger detection tests that we introduce on the client side. At a 5% false-positive rate, a two-direction variant of VGIA, which distributes attacked neurons around opposite directions $\mathbf{w}$ and $-\mathbf{w}$, evades detection in at least 90% of cases across the considered settings and recovers more than 95% of the samples (App. G.4).

6 Discussion and Conclusions

In this work we introduced VGIA, a novel analytical attack in federated learning that provides explicit algebraic certificates of reconstruction correctness, without requiring access to auxiliary datasets, statistical knowledge of the target dataset, or costly human inspection. This makes the attack broadly applicable when prior information is limited, more effective under tight time constraints, and scalable. Moreover, by ruling out empty or already certified regions, VGIA explores the search space more efficiently and requires fewer attack rounds than previous geometric approaches to reconstruct all samples.

VGIA assumes knowledge of input-feature bounds, which need not be tight. Tighter bounds can reduce the required number of attack rounds, as reflected in the round-complexity bound in Sec. 4.3. VGIA remains applicable under secure aggregation (Bonawitz et al., 2017): this mechanism increases the number of samples contributing to the observed aggregate in (2), but does not prevent their geometric isolation given enough attack rounds. Although a comprehensive evaluation against practical defenses remains an important direction for future work, our preliminary experiments in App. H already demonstrate near-complete recovery under the tested clipping thresholds and substantial recovery under Gaussian noise. Our detectability experiments further show that suitable parameter configurations can evade several client-side detection rules (App. G). Together, these results demonstrate that VGIA can remain effective under several tested defenses, while its algebraic certificates provide a principled way to assess reconstruction success under the stated assumptions.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs and in the implementation of methods. We have not used generative AI tools for the other tasks requiring disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to help formulate the paper title, edit software code, support brainstorming, and improve the readability and presentation of the manuscript. We reviewed all AI-assisted work. In particular, AI-assisted mathematical arguments were manually checked by the authors for correctness and consistency with the formal derivations, while AI-assisted code and implementations were reviewed and tested before being used in the experiments. All AI-assisted text was subsequently revised and verified by the authors. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies privacy vulnerabilities in federated learning and introduces a reconstruction attack that allows a malicious server to recover and verify individual client records from model updates. As with other research on privacy and security attacks, our method has a dual-use nature. While the attack is intended to improve the understanding and evaluation of privacy risks in federated learning, the same techniques could potentially be misused to recover private information from users in a real deployment. Our experiments are conducted in controlled settings using publicly available benchmark datasets, and we do not target deployed federated learning systems, interact with real users, or collect additional personal data. The threat model, assumptions, and limitations of the attack are explicitly described in the paper, including the requirement that the adversary can manipulate the model parameters sent to clients. We believe that demonstrating the feasibility and verifiability of such attacks is important for developing more reliable privacy auditing methodologies and motivating stronger client-side defenses against malicious servers. We therefore present VGIA as a tool for studying and exposing privacy risks in federated learning, with the goal of supporting the design and evaluation of more privacy-preserving learning systems. We also discuss existing and potential defense mechanisms and highlight the need for robust client-side validation of server-provided models.

REPRODUCIBILITY STATEMENT

We provide the information and resources needed to reproduce both the theoretical and experimental results of this work. The complete VGIA procedure is specified in Algorithms 1, 2, and 3, with additional algorithmic details provided in App. D.3. The assumptions and complete proof supporting the verification mechanism are provided in App. B, while the round-complexity analysis is given in App. C. Details on the datasets and preprocessing are reported in App. E.1, and the attack configurations, parameter initialization, baseline hyperparameters, numerical precision, and hardware used in the experiments are described in App. E.2. Additional experimental results on three datasets are provided in App. F. We include evaluations under FedAvg in App. I, with SiLU and GELU activations in App. J, and on ResNet and Transformer architectures in App. K. Finally, the supplementary material contains the implementation of VGIA, together with scripts and instructions for reproducing the main experimental results.

REFERENCES

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. 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.

A. Bakarsky, D. I. Dimitrov, M. Baader, and M. Vechev. Spear++: Scaling gradient inversion via sparsely-used dictionary learning. *arXiv preprint arXiv:2510.24200*, 2025.

F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Sym-*

posium on Security and Privacy (EuroS&P), pages 175–199, 2023. doi: 10.1109/EuroSP57164.2023.00020.

K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. 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.

V. Carletti, P. Foggia, C. Mazzocca, G. Parrella, and M. Vento. Sok: Gradient inversion attacks in federated learning. In L. Bauer and G. Pellegrino, editors, *34th USENIX Security Symposium, USENIX Security 2025, Seattle, WA, USA, August 13-15, 2025*, pages 6439–6459. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/carletti.

V. Carletti, P. Foggia, C. Mazzocca, G. Parrella, and M. Vento. On the detectability of active gradient inversion attacks in federated learning. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 2346–2365, 2026. doi: 10.1109/SP63933.2026.00193.

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

H.-M. Chu, J. Geiping, L. H. Fowl, M. Goldblum, and T. Goldstein. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=A9WQaxYsfx.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

F. Diana, A. Nusser, C. Xu, and G. Neglia. Cutting through privacy: A hyperplane-based data reconstruction attack in federated learning. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025. URL https://openreview.net/forum?id=tYtzMBfrTn.

D. I. Dimitrov, M. Balunovic, N. Konstantinov, and M. Vechev. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=e7A0B99zJf.

D. I. Dimitrov, M. Baader, M. N. Mueller, and M. 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.

F. Ding, M. Hardt, J. Miller, and L. Schmidt. Retiring adult: new datasets for fair machine learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. 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.

L. H. Fowl, J. Geiping, S. Reich, Y. Wen, W. Czaja, M. Goldblum, and T. Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r0BrY4BiEXO.

M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813677. URL https://doi.org/10.1145/2810103.2813677.

K. Garov, D. I. Dimitrov, N. Jovanović, and M. 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=krx55l2A6G.

J. Geiping, H. Bauermeister, H. Dröge, and M. 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.

J. Geng, Y. Mou, F. Li, Q. Li, O. Beyan, S. Decker, and C. Rong. Towards general deep leakage in federated learning, 2022. URL https://arxiv.org/abs/2110.09074.

B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134012. URL https://doi.org/10.1145/3133956.3134012.

J. Jeon, J. Kim, K. Lee, S. Oh, and J. Ok. Gradient inversion with generative image prior. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=x9jS8pX3dkx.

S. Kariyappa, C. Guo, K. Maeng, W. Xiong, G. E. Suh, M. K. Qureshi, and H.-H. S. Lee. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. 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.

KCH. URL https://www.openml.org/search?type=data&status=active&id=42092&sort=runs.

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

B. Li, H. Gu, R. Chen, J. Li, C. Wu, N. Ruan, X. Si, and L. Fan. Temporal Gradient Inversion Attacks With Robust Optimization . *IEEE Transactions on Dependable and Secure Computing*, 22 (04):3383–3397, July 2025. ISSN 1941-0018. doi: 10.1109/TDSC.2025.3532339. URL https://doi.ieeecomputersociety.org/10.1109/TDSC.2025.3532339.

Z. Li, L. Wang, G. Chen, Z. Zhang, M. Shafiq, and Z. Gu. E2egi: End-to-end gradient inversion in federated learning. *IEEE Journal of Biomedical and Health Informatics*, 27(2):756–767, 2023. doi: 10.1109/JBHI.2022.3204455.

X. Liu, S. Cai, Q. Zhou, S. Guo, R. Li, and K. Lin. Mjölnir: Breaking the shield of perturbation-protected gradients via adaptive diffusion. In *AAAI Conference on Artificial Intelligence*, 2024. URL https://api.semanticscholar.org/CorpusID:271051379.

J. Lu, X. S. Zhang, T. Zhao, X. He, and J. 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.

K. Ma, Y. Sun, J. Cui, D. Li, Z. Guan, and J. Liu. Instance-wise batch label restoration via gradients in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FIrQfNSOoTr.

B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

J. Meng, T. Huang, H. Chen, C. Hou, and G. Zheng. Enhanced privacy leakage from noise-perturbed gradients via gradient-guided conditional diffusion models, 2025. URL https://arxiv.org/abs/2511.10423.

J. Ogier du Terrail, Q. Klopfenstein, H. Li, I. Mayer, N. Loiseau, M. Hallal, M. Debouver, T. Camalon, T. Fouqueray, J. Arellano Castro, Z. Yanes, L. Dahan, J. Taieb, P. Laurent-Puig, J.-B. Bachet, S. Zhao, R. Nicolle, J. Cros, D. Gonzalez, R. Carreras-Torres, A. Garcia Velasco, K. Abdilleh, S. Doss, F. Balazard, and M. Andreux. Fedeca: federated external control arms for causal inference with time-to-event data in distributed settings. *Nature Communications*, 16(1):7496, 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-62525-z. URL https://doi.org/10.1038/s41467-025-62525-z.

L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning: Revisited and enhanced. In L. Batten, D. S. Kim, X. Zhang, and G. Li, editors, *Applications and Techniques in Information Security*, pages 100–110, Singapore, 2017. Springer Singapore.

J. Qian, K. Wei, Y. Wu, J. Zhang, J. Chen, and H. Bao. Gi-smn: Gradient inversion attack against federated learning without prior knowledge. In *Advanced Intelligent Computing Technology and Applications: 20th International Conference, ICIC 2024, Tianjin, China, August 5–8, 2024, Proceedings, Part VIII*, page 439–448, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-981-97-5602-5. doi: 10.1007/978-981-97-5603-2_36. URL https://doi.org/10.1007/978-981-97-5603-2_36.

H. Ren, J. Deng, and X. Xie. Grnn: Generative regression neural network—a data leakage attack for federated learning. *ACM Trans. Intell. Syst. Technol.*, 13(4), May 2022. ISSN 2157-6904. doi: 10.1145/3510032. URL https://doi.org/10.1145/3510032.

J. Shan, Z. Zhao, J. Lu, R. Zhang, S. M. Yiu, and K.-H. Chow. Geminio: Language-guided gradient inversion attacks in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2718–2727, 2025.

S. Shi, Y. Xiao, C. Zhang, Y. Shi, Y. Hou, and W. Lou. Scale-mia: A scalable model inversion attack against secure federated learning via latent space reconstruction. 01 2025. doi: 10.14722/ndss.2025.240644.

E. Sotthiwat, L. Zhen, C. Zhang, Z. Li, and R. S. M. Goh. Generative image reconstruction from gradients. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1):21–31, 2025. doi: 10.1109/TNNLS.2024.3383722.

S. Upadhyay, S. Tickoo, A. Varshney, S. Kundu, J. Lee, A. Navale, J. Padmakumar, B. E. Perlmutter, V. Singh, R. Anandhan, A. Beatty, C. Y. Jiao, S. Shetty, V. Somashekar, T. Movva, and U. Farooq. Project aikya: Enhanced anomaly detection in financial transactions through decentralized ai. White paper, Kinexys by J.P. Morgan & BNY Mellon, August 2025. URL https://www.jpmorgan.com/kinexys/documents/kinexys-project-aikya-enhanced-anomaly-detection-through-decentralized-ai.pdf. Accessed: 2025-12-17.

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

F. Wang, S. Velipasalar, and M. C. 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.

Y. Wen, J. A. Geiping, L. Fowl, M. Goldblum, and T. Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning*, pages 23668–23684. PMLR, 2022.

R. Wu, X. Chen, C. Guo, and K. Q. Weinberger. Learning to invert: simple adaptive attacks for gradient inversion in federated learning. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI '23. JMLR.org, 2023.

X. Xu, P. Liu, W. Wang, H.-L. Ma, B. Wang, Z. Han, and Y. Han. Cgir: Conditional generative instance reconstruction attacks against federated learning. *IEEE Transactions on Dependable and Secure Computing*, 20(6):4551–4563, 2023. doi: 10.1109/TDSC.2022.3228302.

D. Xue, H. Yang, M. Ge, J. Li, G. Xu, and H. Li. Fast generation-based gradient leakage attacks against highly compressed gradients. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023. doi: 10.1109/INFOCOM53939.2023.10229091.

H. Yang, M. Ge, K. Xiang, and J. Li. Using highly compressed gradients in federated learning for data reconstruction attacks. *IEEE Transactions on Information Forensics and Security*, 18: 818–830, 2023. doi: 10.1109/TIFS.2022.3227761.

H. Yang, D. Xue, M. Ge, J. Li, G. Xu, H. Li, and R. Lu. Fast generation-based gradient leakage attacks: An approach to generate training data directly from the gradient. *IEEE Transactions on Dependable and Secure Computing*, 22(1):132–145, 2025. doi: 10.1109/TDSC.2024.3387570.

H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. 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.

K. Yue, R. Jin, C.-W. Wong, D. Baron, and H. Dai. Gradient obfuscation gives a false sense of security in federated learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6381–6398, Anaheim, CA, Aug. 2023. USENIX Association. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/yue.

C. Zhang, X. Zhang, E. Sotthiwat, Y. Xu, P. Liu, L. Zhen, and Y. Liu. Generative gradient inversion via over-parameterized networks in federated learning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5103–5112, 2023a. doi: 10.1109/ICCV51070.2023.00473.

R. Zhang, S. Guo, and P. Li. Gradfilt: Class-wise targeted data reconstruction from gradients in federated learning. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 698–701, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701726. doi: 10.1145/3589335.3651514. URL https://doi.org/10.1145/3589335.3651514.

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

B. Zhao, K. R. Mopuri, and H. Bilen. idlg: Improved deep leakage from gradients, 2020. URL https://arxiv.org/abs/2001.02610.

J. C. Zhao, A. Sharma, A. R. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. 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.

J. Zhu and M. B. Blaschko. R-{gap}: Recursive gradient attack on privacy. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=RSU17UoKfJF.

L. Zhu, Z. Liu, and S. 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.

A Related Work

A.1 Optimization-based attacks

In the honest-but-curious setting, Zhu et al. (2019) introduced the first gradient inversion attack, demonstrating that private inputs can be reconstructed from shared gradients in Federated Learning. Their approach optimizes dummy inputs and labels by minimizing the distance between the gradients induced by the dummy data and the observed updates.

Subsequent work has significantly refined this paradigm. To improve reconstruction fidelity and optimization convergence, several methods decouple the recovery of inputs and labels (Zhao et al., 2020; Li et al., 2023; Dimitrov et al., 2022; Geng et al., 2022; Ma et al., 2023), often augmenting the optimization objective with regularization terms or domain-specific priors (Geiping et al., 2020; Yin et al., 2021; Dimitrov et al., 2022; Geng et al., 2022). More recent attacks (Yang et al., 2023; Yue et al., 2023; Liu et al., 2024) further show that reconstruction remains feasible even when gradients are perturbed. Despite these advances, optimization-based attacks still face fundamental limitations. They scale poorly with input dimensionality and batch size, and reconstruction quality degrades rapidly when batch size exceeds a few dozen samples. To mitigate this issue, Kariyappa et al. (2023) propose using Independent Component Analysis to separate per-sample gradient contributions, while Li et al. (2025) exploit information accumulated across multiple communication rounds. Nevertheless, these approaches remain constrained in practice, with performance deteriorating significantly for batch sizes larger than 256 samples.

A complementary line of work considers active, optimization-based attacks in which a malicious server modifies the model to amplify or filter specific gradient contributions. Wen et al. (2022) introduce gradient magnification, where the malicious server modifies model parameters so that the gradient contribution of selected samples dominates the batch-averaged update, enabling optimization-based reconstruction from larger batches. Zhang et al. (2024) propose a class-wise targeted attack that modifies the classification layer to suppress gradients from non-target classes while preserving those of attacker-specified classes, thereby focusing reconstruction on samples of interest. Shan et al. (2025) extend this targeted perspective by leveraging a pretrained vision-language model to optimize a malicious global model whose gradients retain samples matching a natural-language query, enabling semantically guided reconstruction from large batches.

Finally, due to the intrinsic difference between approximate optimization-based reconstructions and true records, most prior work focuses on the image domain, where human evaluation of the reconstruction is possible, leaving tabular settings underexplored (Carletti et al., 2025). Vero et al. (2023) propose the only optimization-based attack tailored to tabular data; however, even their attack does not achieve exact record recovery and remains limited to small batch regimes.

A.2 Generative-based attacks

Generative-based attacks leverage generative models to reconstruct private inputs from shared gradients, typically by constraining or learning the data distribution underlying the reconstruction process. Early works incorporate Generative Adversarial Networks (GANs) into the attack pipeline. Hitaj et al. (2017) propose a setting where a malicious client trains a GAN by treating the global model as a discriminator to recover victim data.

Rather than training generators online, subsequent work leverages pre-trained generative models to constrain the search space from the high-dimensional input domain to a lower-dimensional latent manifold. Jeon et al. (2021); Qian et al. (2024) formulate gradient inversion as an optimization problem over the latent code of a generator pre-trained on public data.

A complementary line of work focuses on directly training a generator to match observed gradients (Ren et al., 2022; Wu et al., 2023; Zhang et al., 2023a; Xu et al., 2023; Sotthiwat et al., 2025; Xue et al., 2023; Yang et al., 2025). In this setting, the input is fixed while the generator parameters are optimized so that the generated samples produce gradients consistent with the observations.

Recent advancements have incorporated diffusion models to enhance reconstruction fidelity. Notably, Meng et al. (2025) designed a gradient-guided conditional diffusion framework robust to noise and Differential Privacy (DP) mechanisms (Dwork, 2006).

Despite their flexibility, these methods typically remain limited to small batch sizes. To improve scalability while reducing the detectability of malicious-server attacks, recent generative-based approaches combine model manipulation with encoder–decoder architectures trained on auxiliary data. Garov et al. (2024) propose SEER, in which the malicious server jointly trains the shared model with a secret decoder and chooses model parameters so that client gradients encode private data in a latent representation that can later be decoded by the server. This design makes the malicious model less distinguishable from a benign one while enabling reconstruction at larger batch sizes and under secure aggregation. Shi et al. (2025) similarly target reconstruction in latent space: inspired by the imprinting mechanism of Fowl et al. (2022), the attacker uses specially crafted linear layers in the latent space to recover latent-space representations from aggregated gradients, and then applies a fine-tuned generative decoder to map these representations back to the input space. These methods improve scalability compared with direct optimization-based reconstruction, but rely on auxiliary data to train the decoder and provide approximate reconstructions rather than intrinsic certificates of correctness.

However, they still remain limited to image data, where human verification is possible. For tabular data it is impossible to distinguish between a correctly reconstructed record and a plausible sample generated from the model prior. In addition, many of these methods often rely on auxiliary datasets that share the same distribution as the private data, which may not be available in practice.

A.3 Analytical attacks

A third group of attacks explores analytical properties of the gradients to retrieve perfect reconstructions of client data. In the honest-but-curious setting, Phong et al. (2017) demonstrated that it is possible to recover samples in closed form from FC-layers followed by a ReLU activation and Zhu and Blaschko (2021) show that recovery is possible also from convolutional neural networks (CNNs). Lu et al. (2022) demonstrate that the self-attention mechanism of Transformer architectures can also be subject to closed-form analytical inversion. All these attacks are limited to the case of a single sample or very few samples in the batch. Dimitrov et al. (2024) and later Bakarsky et al. (2025) showed that by exploiting the low-rank structure and sparsity of gradients it is possible to recover samples from batches of up to 200 samples. However, they require a batch size smaller than the input dimension. Overall, batch size remains the main limitation in this threat model.

In the malicious setting, Fowl et al. (2022) designed an imprint module assuming auxiliary knowledge to separate samples' gradient contributions, and Zhao et al. (2024) adapted the attack to break secure aggregation. Boenisch et al. (2023) and Zhang et al. (2023b) designed specific malicious distributions to have neurons activated by a single sample. However, the accuracy of these attacks degrades with large batch sizes as shown by Diana et al. (2025), who propose a geometrical framework to overcome batch size limitations, but require suspicious and easily detectable model modifications. Finally, to address more complex model architectures and classification tasks, Wang et al. (2024) designed specific weight modifications to reduce detectability and obtain analytical recovery of inputs in CNNs; Fowl et al. (2023) and Chu et al. (2023) adapted the work from Fowl et al. (2022) to recover text sequences from transformers.

B Proof of Proposition 4.1

Proposition B.1 (Complete Statement of Proposition 4.1)**.** *Consider the non-empty slice delimited at round t by the hyperplanes $\mathbf{w}^\top \mathbf{x} + \hat{b}_k^t = 0$ and $\mathbf{w}^\top \mathbf{x} + \hat{b}_{k+1}^t = 0$, and suppose that it contains $m \leq d$ linearly independent inputs. Assume that the common first-layer direction $\mathbf{w}$ is drawn once from a distribution absolutely continuous with respect to the Lebesgue measure. At each attack round, assume that the vectorized weights and biases of the layers following the attacked layer are redrawn independently of the past from absolutely continuous distributions, supported on parameters for which each downstream hidden layer has at least two fixed ReLU gates active on all inputs in the slice. Finally, assume that, for every sample j, all level sets of $\partial \mathcal{L}_j / \partial z^L$ are Lebesgue-null.*

Then, almost surely, the span condition

$$\mathbf{s}_k^t \in \operatorname{span}\left(\mathbf{s}_k^{t+1}, \ldots, \mathbf{s}_{k+q-2}^{t+1}\right)$$

implies that every non-empty sub-slice induced at round $t+1$ contains exactly one input. Hence, when the span condition holds, the original slice contains as many inputs as there are non-null induced slice vectors, and therefore at most $q-1$ inputs.

Proof. For each sample j, define

$$\psi_j(z) := \frac{1}{B}\frac{\partial \ell(z, y_j)}{\partial z}.$$

Every level set $\psi_j^{-1}(c)$ has one-dimensional Lebesgue measure zero. Let $\mathcal{X}$ denote the non-empty slice delimited at round t by the hyperplanes $\mathbf{w}\mathbf{x} + \hat{b}_k^t = 0$ and $\mathbf{w}\mathbf{x} + \hat{b}_{k+1}^t = 0$. Recall that

$$\mathbf{s}_k^t = \sum_{j=1}^{m} \beta_j^t \mathbf{x}_j, \qquad \beta_j^t = \psi_j(z_j^{L,t}).$$

Let the round-$(t+1)$ refinement partition $\mathcal{X}$ into $q-1$ sub-slices $\{\mathcal{X}_r\}_{r=k}^{k+q-2}$, where $\mathcal{X} = \{\mathbf{x}_j : \mathbf{w}^\top \mathbf{x}_j + \hat{b}_{k+1}^t > 0 \text{ and } \mathbf{w}^\top \mathbf{x}_j + \hat{b}_k^t \le 0\}$ and $\mathcal{X} = \biguplus_{r=k}^{k+q-2} \mathcal{X}_r$, where $\biguplus$ denotes a disjoint union. The induced slice vectors are

$$\mathbf{s}_r^{t+1} = \sum_{\mathbf{x}_j \in \mathcal{X}_r} \beta_j^{t+1} \mathbf{x}_j, \qquad \beta_j^{t+1} = \psi_j(z_j^{L,t+1}).$$

For each j, let $r(j)$ denote the index of the refined sub-slice containing $\mathbf{x}_j$.

We first state a probabilistic fact we need later. Fix two distinct samples $\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}$. Since $\mathbf{x}_a - \mathbf{x}_b \neq 0$ and $\mathbf{w}$ is drawn from an absolutely continuous distribution, we have that $\mathbf{w}^\top(\mathbf{x}_a - \mathbf{x}_b) \neq 0$ almost surely. Thus the projections of $\mathbf{x}_a$ and $\mathbf{x}_b$ along $\mathbf{w}$ are distinct.

Consider a round $\tau \in \{t, t+1\}$ in which $\mathbf{x}_a$ and $\mathbf{x}_b$ lie in the same queried slice. For any two samples lying in the same slice, exactly one of the two boundary neurons is active for both samples. In that coordinate, the first-layer activations are $\mathbf{w}^\top \mathbf{x}_a + b_i^\tau$ and $\mathbf{w}^\top \mathbf{x}_b + b_i^\tau$, which are distinct. Hence the first-layer post-activation representations of $\mathbf{x}_a$ and $\mathbf{x}_b$ are distinct.

We now show that the downstream random layers make the corresponding final logits jointly absolutely continuous while preserving the prescribed ReLU gating pattern. Fix a downstream hidden layer and let $K \subset \mathbb{R}_+^n$ be a compact set containing all representations entering this layer for the inputs under consideration. Since these representations are outputs of ReLU layers, they are non-negative coordinatewise.

Let the affine map before the ReLU be

$$F_{W,b}(u) = Wu + b, \qquad W \in \mathbb{R}^{p \times n}, \quad b \in \mathbb{R}^p,$$

with $p \ge 2$. We assume that the first two gates are prescribed to be open. Fix a margin $\gamma > 0$. For $i = 1, 2$, define the activation threshold $\underline{b}_i(W_{i,:}) := \gamma - \min_{u \in K} W_{i,:}u$. The attacker samples $W_{i,:}$ from an absolutely continuous distribution and, conditional on $W_{i,:}$, samples $b_i = \underline{b}_i(W_{i,:}) + \eta_i$, where η_i is drawn independently from an absolutely continuous distribution supported on an interval $I = (\eta_-, \eta_+) \subset (0, \infty)$. Then, for every $u \in K$,

$$W_{i,:}u + b_i = W_{i,:}u + \gamma - \min_{v \in K} W_{i,:}v + \eta_i \ge \gamma + \eta_i > 0.$$

Hence the first two ReLU gates are strictly active for all inputs under consideration.

Moreover, the joint distribution of $(W_{i,:}, b_i)$ is absolutely continuous: indeed, it is obtained from the absolutely continuous pair $(W_{i,:}, \eta_i)$ through the triangular change of variables

$$(W_{i,:}, \eta_i) \mapsto (W_{i,:}, \underline{b}_i(W_{i,:}) + \eta_i),$$

whose Jacobian determinant with respect to the bias coordinate is one.

Let $u_a, u_b \in K$ be two fixed distinct representations entering the layer, and set $\Delta := u_a - u_b \neq 0$. Consider the determinant of the first two affine output coordinates:

$$P(W, b) = \det \begin{pmatrix} (F_{W,b}(u_a))_1 & (F_{W,b}(u_b))_1 \\ (F_{W,b}(u_a))_2 & (F_{W,b}(u_b))_2 \end{pmatrix}.$$

This is a polynomial in the entries of W and b. We claim that P is not identically zero. Choose a row vector r in the support of the row distribution such that $r^\top \Delta \neq 0$. Such an r exists whenever the support of the row distribution contains a non-empty open set, because the set $\{r : r^\top \Delta = 0\}$ is a proper hyperplane. Let $m_r := \min_{u \in K} r^\top u$. Choose two distinct slack values $\eta_1, \eta_2 \in I$, with $\eta_1 \neq \eta_2$. Now define a particular parameter choice (W_0, b_0) by setting

$$(W_0)_{1,:} = r^\top, \qquad (W_0)_{2,:} = r^\top,$$

and

$$(b_0)_1 = \gamma - m_r + \eta_1, \qquad (b_0)_2 = \gamma - m_r + \eta_2.$$

The remaining rows and biases may be chosen arbitrarily in their prescribed supports.

This witness satisfies the open-gate constraint. Indeed, for $i = 1, 2$ and every $u \in K$,

$$(W_0)_{i,:}u + (b_0)_i = r^\top u + \gamma - m_r + \eta_i \geq \gamma + \eta_i > 0.$$

Thus the first two gates are strictly active for this witness.

Let

$$A := r^\top u_a + \gamma - m_r, \qquad B := r^\top u_b + \gamma - m_r.$$

Then

$$(F_{W_0,b_0}(u_a))_{1:2} = \begin{pmatrix} A + \eta_1 \\ A + \eta_2 \end{pmatrix}, \qquad (F_{W_0,b_0}(u_b))_{1:2} = \begin{pmatrix} B + \eta_1 \\ B + \eta_2 \end{pmatrix}.$$

Therefore

$$P(W_0, b_0) = \det \begin{pmatrix} A + \eta_1 & B + \eta_1 \\ A + \eta_2 & B + \eta_2 \end{pmatrix} = (\eta_2 - \eta_1)(A - B).$$

Since

$$A - B = r^\top (u_a - u_b) = r^\top \Delta \neq 0$$

and $\eta_2 - \eta_1 \neq 0$, we obtain $P(W_0, b_0) \neq 0$. Thus P is a nonzero polynomial.

The zero set of a nonzero polynomial has Lebesgue measure zero. Since the actual parameters (W, b) are drawn from an absolutely continuous distribution, even after restricting the biases to satisfy the open-gate lower bounds, we have

$$\mathbb{P}(P(W, b) = 0) = 0.$$

Consequently, with probability one, the first two active affine coordinates of $F_{W,b}(u_a)$ and $F_{W,b}(u_b)$ are linearly independent. Since the first two ReLU gates are active, the corresponding post-ReLU representations are also linearly independent with probability one.

Repeating the preceding argument over the finitely many downstream hidden layers shows that the penultimate representations $z_a^{L-1,\tau}$ and $z_b^{L-1,\tau}$ are linearly independent almost surely. Condition now on all layers up to layer $L - 1$. The final scalar logits are

$$z_a^{L,\tau} = W^{L,\tau} z_a^{L-1,\tau} + b^{L,\tau}, \qquad z_b^{L,\tau} = W^{L,\tau} z_b^{L-1,\tau} + b^{L,\tau}.$$

The linear map

$$(W^{L,\tau}, b^{L,\tau}) \mapsto \left(W^{L,\tau} z_a^{L-1,\tau} + b^{L,\tau}, W^{L,\tau} z_b^{L-1,\tau} + b^{L,\tau} \right)$$

has rank two because $z_a^{L-1,\tau}$ and $z_b^{L-1,\tau}$ are linearly independent. Since $(W^{L,\tau}, b^{L,\tau})$ is absolutely continuous, it follows that $(z_a^{L,\tau}, z_b^{L,\tau})$ has a joint density in $\mathbb{R}^2$. In particular, each $z_j^{L,\tau}$ has a density. Since the zero level set $\psi_j^{-1}(0)$ has Lebesgue measure zero, we have

$$\mathbb{P}(\beta_j^\tau = 0) = \mathbb{P}(\psi_j(z_j^{L,\tau}) = 0) = 0.$$

Thus all coefficients β_j^t and β_j^{t+1} are nonzero almost surely.

Next, we prove that two coefficients corresponding to two distinct samples $\mathbf{x}_a$ and $\mathbf{x}_b$ are not aligned across rounds, that is, $(\beta_a^{t+1}, \beta_b^{t+1})$ is not proportional to (β_a^t, β_b^t). Condition on the parameters at round t and on the round-$(t+1)$ first-layer parameters. In particular, condition on the realized values

$$c_a := \beta_a^t, \qquad c_b := \beta_b^t.$$

The downstream parameters at round $t+1$ are independent of this conditioning, so, on the event that $\mathbf{x}_a$ and $\mathbf{x}_b$ lie in the same refined sub-slice, the pair $(z_a^{L,t+1}, z_b^{L,t+1})$ still has a joint density.

Consider the alignment event $c_b\beta_a^{t+1} - c_a\beta_b^{t+1} = 0$. Since $\beta_j^{t+1} = \psi_j(z_j^{L,t+1})$, this event is equivalent to $(z_a^{L,t+1}, z_b^{L,t+1}) \in S_c$, where

$$S_c = \left\{(u,v) \in \mathbb{R}^2 : c_b\psi_a(u) - c_a\psi_b(v) = 0\right\}.$$

Because c_a and c_b are not both zero almost surely, the set S_c has two-dimensional Lebesgue measure zero. To see this, suppose for example that $c_a \neq 0$. Then for each fixed u, the vertical section of S_c is

$$\left\{v : \psi_b(v) = \frac{c_b}{c_a}\psi_a(u)\right\},$$

which is a level set of ψ_b and therefore has one-dimensional Lebesgue measure zero. By Fubini's theorem, S_c has two-dimensional Lebesgue measure zero. The cases $c_a = 0, c_b \neq 0$ and $c_b = 0, c_a \neq 0$ are analogous. Since $(z_a^{L,t+1}, z_b^{L,t+1})$ has a density, we conclude that

$$\mathbb{P}\left[\beta_b^t\beta_a^{t+1} - \beta_a^t\beta_b^{t+1} = 0\right] = 0$$

for every fixed pair $\mathbf{x}_a \neq \mathbf{x}_b$ that lies in the same refined sub-slice.

We now prove the proposition. Assume that the span condition holds:

$$\mathbf{s}_k^t \in \operatorname{span}\left(\mathbf{s}_k^{t+1}, \ldots, \mathbf{s}_{k+q-2}^{t+1}\right).$$

Then there exist scalars $\{\alpha_r\}_{r=k}^{k+q-2}$ such that

$$\mathbf{s}_k^t = \sum_{r=k}^{k+q-2} \alpha_r \mathbf{s}_r^{t+1}.$$

Expanding both sides gives

$$\sum_{j=1}^{m} \beta_j^t \mathbf{x}_j = \sum_{r=k}^{k+q-2} \alpha_r \sum_{\mathbf{x}_j \in \mathcal{X}_r} \beta_j^{t+1}\mathbf{x}_j = \sum_{j=1}^{m} \alpha_{r(j)}\beta_j^{t+1}\mathbf{x}_j.$$

Since $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are linearly independent, the coefficients in this linear expansion are unique. Therefore, for every j we have $\beta_j^t = \alpha_{r(j)}\beta_j^{t+1}$.

Suppose, toward a contradiction, that some refined sub-slice $\mathcal{X}_r$ contains at least two samples. Choose two of them, say $\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}_r$. Then $r(a) = r(b) = r$, so the coefficient identities above give

$$\beta_a^t = \alpha_r\beta_a^{t+1}, \qquad \beta_b^t = \alpha_r\beta_b^{t+1}.$$

Eliminating α_r yields

$$\beta_b^t\beta_a^{t+1} - \beta_a^t\beta_b^{t+1} = 0.$$

But, as shown above, this alignment event has probability zero for any fixed pair of distinct samples in the same refined sub-slice. Taking a finite union over all pairs of samples in $\mathcal{X}$, we conclude that, almost surely, the span condition cannot hold if any refined sub-slice contains two or more samples.

Therefore, whenever the span condition holds, every non-empty refined sub-slice contains at most one input. Since a non-empty sub-slice contains at least one input by definition, every non-empty refined sub-slice contains exactly one input.

Finally, if $\mathcal{X}_r = \{\mathbf{x}_j\}$ is a singleton sub-slice, then $\mathbf{s}_r^{t+1} = \beta_j^{t+1}\mathbf{x}_j$. As $\mathbf{x}_j \neq 0$ by linear independence and $\beta_j^{t+1} \neq 0$ almost surely, we have $\mathbf{s}_r^{t+1} \neq 0$ almost surely. Conversely, if $\mathcal{X}_r = \emptyset$, then $\mathbf{s}_r^{t+1} = 0$. Hence the number of inputs in the original slice is equal to the number of non-null induced slice vectors. Since there are only $q-1$ induced sub-slices, this number is at most $q-1$. $\square$

B.1 Consequences of Violating the Linear Independence Assumption

Proposition 4.1 provides sufficient conditions for the soundness of the isolation certificate. If these assumptions are violated, the span condition may in principle hold for a slice containing multiple samples, leading to the reconstruction of a linear combination of its inputs. In practice, however, this behavior is rare and remains localized to the corresponding slice.

Importantly, linear independence is required only among the samples contained in the same slice, rather than across the full batch. For example, King County Housing has only $d = 18$ input features, while our experiments in App. F use $B = 2048$ samples and $N = 1000$ attacked neurons. After the initial partition, each slice therefore contains only about two samples on average, so the independence condition applies only to these small local subsets. The full batch itself need not be linearly independent.

Moreover, linear dependence within a slice does not necessarily result in an incorrect certification. If the span condition does not hold, VGIA continues subdividing the slice, and the resulting smaller subsets may become linearly independent. Any incorrect certification is also local to the corresponding slice and does not affect the verification of other slices.

Finally, the attacker can mitigate violations of the linear-independence assumption in two ways. First, some violations can be detected during the search. For example, if refining a parent slice reveals more than d non-empty sub-slices, then the parent must contain more than d samples and therefore cannot satisfy the assumption; the attacker can then continue refining the remaining unresolved regions instead of certifying them. Second, the attacker can adopt a more conservative strategy even when no violation is detected, by postponing certification for an additional refinement round after the span condition is first satisfied. This extra round may expose hidden dependencies by further separating samples that had previously remained in the same slice, at the cost of additional communication rounds.

C Bounding the Number of Rounds

In this section we consider theoretical bounds on the number of rounds that our attack takes. Assume that $N/3 \geq B$, i.e., there are at least three times as many neurons in the attack layer as there are points in the batch. The upper bound on the number of rounds for our attack can be derived in an analogous way as for the CTP attack (Diana et al., 2025, Appendix D), with two differences:

1. As opposed to CTP, we need to re-send hyperplanes in each round, while CTP can re-use the information from previous rounds. Hence our upper bound on the number of rounds has an additional factor of 3, as our approach needs 3 hyperplanes for a single isolated slice in which we want to search in the worst case, while CTP only needs a single hyperplane. Note though that if we have many consecutive slices that contain points, then this constant tends to be closer to 2.

2. On the other hand, recall that in CTP a parameter ε that is a bound on the slice width until which we search has to be provided. If ε is chosen too large in CTP, their approach fails to reconstruct part of the batch, as shown in Figs. 3b, 4b, 5b, and 3e. Hence, to make full reconstruction more probable, a small ε should be chosen. This in turn leads to more rounds, as their tight bound on the number of rounds is $\left\lceil \log_2 \left(\frac{W}{N \cdot \varepsilon} \right) \right\rceil + 1$. On the other hand, our approach does not require a parameter ε but instead detects if a slice contains only a single point and hence does not search further.

The above implies that while CTP needs $\left\lceil \log_2 \left(\frac{W}{N \cdot \varepsilon} \right) \right\rceil + 1$ rounds in the worst case, our approach needs $3 \cdot \left(\left\lceil \log_2 \left(\frac{W}{N \cdot \varepsilon_{\mathbf{w}}} \right) \right\rceil \right) + 1$ rounds in the worst case, where $\varepsilon_{\mathbf{w}}$ is the minimum distance between two points after projection to the hyperplane search direction and W is the width of the initial interval. Thus, at the latest when $\varepsilon \leq \frac{\varepsilon_{\mathbf{w}}^3 N^2}{W^2}$, our approach takes at most as many rounds as CTP and fewer if ε is chosen even smaller. In particular, note that ε can be arbitrarily smaller than $\varepsilon_{\mathbf{w}}$ as a good value for ε cannot be known a priori and has to be guessed in CTP.

D TECHNICAL DETAILS

D.1 EXTENSION TO MULTICLASS CLASSIFICATION

While the proposed attack is naturally formulated for scalar-output tasks, its core mechanism can be extended to multiclass classification. In this setting, the model outputs a vector $\mathbf{z}^L \in \mathbb{R}^C$, and the loss is typically the cross-entropy between $\mathbf{z}^L$ and a one-hot target vector $\mathbf{y} \in \mathbb{R}^C$.

A direct application of the attack is not possible in this case. Indeed, the gradient signal backpropagated to the attacked layer is a linear combination of error terms from all C classes, weighted by the downstream network. In particular, the gradient with respect to a bias b_i^a for a given sample $\mathbf{x}_j$ takes the form

$$\frac{\partial \mathcal{L}_j}{\partial b_i^a} = \sum_{c=1}^{C} \frac{\partial \mathcal{L}_j}{\partial z_c^L} g_{c,i} \mathbb{1}_{z_{i,j}^a > 0}, \tag{6}$$

where the coefficients $g_{c,i}$ depend on both the class c and the neuron i through the intermediate layers. As a consequence, the backpropagated signal is no longer a sample-dependent scalar multiplied by a neuron-dependent factor, but rather a neuron-dependent mixture of class residuals. This breaks the cancellation mechanism in (4), which is crucial for isolating individual samples. To overcome this limitation, one possible solution for the attacker is to modify the final classification layer so that the network behaves, during the attack phase, as a scalar-output model. Specifically, the attacker sets the last-layer weights to be rank one:

$$\mathbf{W}^L = \mathbf{c}\mathbf{u}^\top, \quad \text{where} \quad \mathbf{c} \in \mathbb{R}^C \text{ and } \mathbf{u} \in \mathbb{R}^{n_{L-1}} \text{ are random vectors.} \tag{7}$$

Thus, for each class c, for a given sample $\mathbf{x}_j$, the output of the model becomes:

$$z_{j,c}^L = c_c \mathbf{u}^\top \mathbf{z}_j^{L-1} + b_c^L. \tag{8}$$

Let us define the scalar projection $r_j := \mathbf{u}^\top \mathbf{z}_j^{L-1}$; then we have $z_{j,c}^L = c_c r_j + b_c^L$. Through our model modifications, the attacker can fix the parameters of all the intermediate layers to make them behave as an affine function. Hence, the map from $\mathbf{z}_j^a$ to r_j is fixed and the derivative with respect to neuron i in the attack layer is the scalar:

$$\frac{\partial r_j}{\partial z_i^a} = g_i = \left[\mathbf{u}^\top \prod_{\ell=a+1}^{L-1} \mathbf{D}_j^\ell \mathbf{W}^\ell \right]_i, \tag{9}$$

where $\mathbf{D}_j^\ell$ is the diagonal matrix of activation derivatives at layer ℓ and does not depend on $\mathbf{x}_j$. If we denote by $\mathbf{p}_j \in \mathbb{R}^C$ the output of the model after the softmax layer, then

$$\frac{\partial \mathcal{L}_j}{\partial z_{j,c}^L} = p_{j,c} - y_{j,c}, \tag{10}$$

therefore

$$\frac{\partial z_{j,c}^L}{\partial z_{j,i}^a} = c_c \frac{\partial r_j}{\partial z_{j,i}^a} = c_c g_i. \tag{11}$$

Applying the chain rule, the gradient of the loss for sample $\mathbf{x}_j$ with respect to b_i is

$$\frac{\partial \mathcal{L}_j}{\partial b_i^a} = \sum_{c=1}^{C} (p_{j,c} - y_{j,c}) \frac{\partial z_{j,c}^L}{\partial b_i^a} = g_i \left(\sum_{c=1}^{C} c_c (p_{j,c} - y_{j,c}) \right) \mathbb{1}_{z_{i,j}^a > 0} \tag{12}$$

Define

$$\rho_j := \sum_{c=1}^{C} c_c (p_{j,c} - y_{j,c}) = \mathbf{c}^\top (\mathbf{p}_j - \mathbf{y}_j), \tag{13}$$

then

$$\frac{\partial \mathcal{L}_j}{\partial b_i^a} = g_i \rho_j \mathbb{1}_{z_{i,j}^a > 0} \quad \text{and} \quad \frac{\partial \mathcal{L}_j}{\partial \mathbf{W}_i^a} = g_i \rho_j \mathbf{x}_j \mathbb{1}_{z_{i,j}^a > 0}. \tag{14}$$

Thus, (4) becomes:

$$\frac{1}{g_{i+1}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{i+1}^a} - \frac{1}{g_i}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i^a} = \frac{1}{B}\sum_{\substack{j\in\mathcal{A}_{i+1}\\ j\notin\mathcal{A}_i}} \rho_j \mathbf{x}_j. \tag{15}$$

The same process applies to β_j in (5) and allows perfect isolation and recovery of the input. Additionally, target recovery can be performed by directly solving (16), evaluating each candidate label $\hat{y}_k$ and selecting the one that minimizes the reconstruction error.

D.2 Target reconstruction

The final phase of the attack focuses on recovering the private target value y_k. While target inference has been widely studied in classification tasks (Yin et al., 2021; Zhao et al., 2020; Kariyappa et al., 2023; Li et al., 2023), comparatively little work has addressed continuous regression targets, where the lack of discrete labels complicates both inference and verification. Given an exact reconstruction of the input $\mathbf{x}_k$, target recovery can be cast as a univariate optimization problem. Specifically, the attacker seeks the target value $\hat{y}_k$ that minimizes the distance between the ground-truth partial derivative and a virtual gradient locally computed by the server:

$$\hat{y}_k \in \underset{y}{\operatorname{argmin}} \left(\frac{\partial \mathcal{L}_k}{\partial b_i^a} - \frac{\partial \mathcal{L}}{\partial b_i^a}(\mathbf{x}_k, y) \right)^2. \tag{16}$$

We note that $\frac{\partial \mathcal{L}_k}{\partial b_i^a} = \frac{\partial \mathcal{L}_k}{\partial z^L}\frac{\partial z^L}{\partial b_i^a}$. The attacker can compute $\frac{\partial z^L}{\partial b_i^a}$, because it knows the model parameters and the input $\mathbf{x}_k$. It can then obtain $\frac{\partial \mathcal{L}_k}{\partial z^L}$ from (5). The second term is a function of y the attacker can compute because it knows the model parameters and $\mathbf{x}_k$. If the derivative of the loss with respect to the network output $\partial \mathcal{L}/\partial z^L$ is monotone, as is the case for the squared loss, then the problem in (16) admits a unique solution.

D.3 Algorithm Details

Here, we present our subroutine CheckIsolation (Algo. 3), which appears in Algo. 1.

Algorithm 3 CheckIsolation

Input: search intervals $\mathcal{I}$, and list of sub-slices $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \dots)$

$\mathcal{I}_{\text{new}} \leftarrow ()$; $\mathcal{R} \leftarrow ()$
while $len(\mathcal{S}) > 0$ **do**
  $(([l,u], \mathbf{s}'), \mathcal{I}) \leftarrow (\mathcal{I}_1, (\mathcal{I}_2, \dots, \mathcal{I}_{|\mathcal{I}|}))$
    // Pop front of the list $\mathcal{I}$ and take the slice vector $\mathbf{s}'$ for the interval $[l,u]$
  $\mathcal{R}_{\text{temp}} \leftarrow ()$; $\mathcal{I}_{\text{temp}} \leftarrow ()$; $\mathcal{V}_{\text{temp}} \leftarrow ()$
  $(b', b'', \mathbf{s}, \beta) \leftarrow \mathcal{S}_1$
  while $b'' \in [l,u]$ **do**
    if $\mathbf{s} \neq \mathbf{0}$ **then**
      $\mathcal{V}_{\text{temp}} \leftarrow (\mathcal{V}_{\text{temp}}, \mathbf{s})$; $\mathcal{R}_{\text{temp}} \leftarrow (\mathcal{R}_{\text{temp}}, (b', b'', \mathbf{s}, \beta))$, $\mathcal{I}_{\text{temp}} \leftarrow (\mathcal{I}_{\text{temp}}, ([b', b''], \mathbf{s}))$
    $((b', b'', \mathbf{s}, \beta), \mathcal{S}) \leftarrow (\mathcal{S}_1, (\mathcal{S}_2, \dots, \mathcal{S}_{|\mathcal{S}|}))$
      // Take all the non-empty sub-slices within the interval $[l,u]$ in $\mathcal{V}_{temp}$
  if $|\mathcal{V}_{\text{temp}}| > 0$ **then**
    if $\dim(\text{span}(\mathcal{V}_{\text{temp}})) = \dim(\text{span}(\mathcal{V}_{\text{temp}}, \mathbf{s}'))$ **then**
      $\mathcal{R} \leftarrow (\mathcal{R}, \mathcal{R}_{\text{temp}})$ // Verified isolation by Prop. 4.1, collect sub-slices for reconstruction
    else
      $\mathcal{I}_{\text{new}} \leftarrow (\mathcal{I}_{\text{new}}, \mathcal{I}_{\text{temp}})$ // Add the intervals of sub-slices for future exploration
Return $(\mathcal{I}, \mathcal{I}_{\text{new}}), \mathcal{R}$

E Additional Experimental Details

E.1 Datasets

For regression tasks, we evaluate our attack on the ACS Income dataset (Ding et al., 2021) and the King County Housing dataset (KCH). The ACS Income dataset consists of census records from all

50 U.S. states and Puerto Rico and includes a diverse set of demographic attributes, with the goal of predicting individual income. In our experiments, we restrict the data to samples from the state of California. The King County Housing dataset contains information on residential property transactions in King County, Washington, and comprises 18 input features, with house price prediction as the target task. For both datasets, input features are rescaled to the $[0, 1]$ interval, while target variables are standardized.

For classification tasks, we evaluate our approach on three datasets: the Human Activity Recognition Using Smartphones (HARUS) dataset (Anguita et al., 2013), CIFAR10 (Krizhevsky, 2009), and ImageNet Deng et al. (2009). The HARUS dataset consists of motion signals collected from 30 subjects using embedded smartphone sensors and includes 561 input features and 6 activity labels corresponding to distinct physical activities. All input features are scaled to the $[-1, 1]$ range. CIFAR10 is a benchmark image classification dataset comprising 60,000 color images of size 32×32 across 10 object classes; for this dataset, pixel values are rescaled to the $[0, 1]$ interval. ImageNet is a large-scale image classification benchmark containing natural images from 1,000 object categories. Images are reshaped to $224 \times 224 \times 3$ and rescaled to the $[0, 1]$ interval.

E.2 Attack Configuration

To reduce the impact of numerical instabilities, all experiments are performed in double precision and gradients are computed sample-wise and then aggregated. For all datasets, VGIA samples one vector $\mathbf{w} \sim \mathcal{N}\left(\mathbf{0}, 10^{-4}\mathbf{I}_d\right)$ and assigns this vector to each row of the attacked weight matrix $\mathbf{W}^a$. For the classification datasets, namely HARUS, CIFAR-10, and ImageNet, all weights and biases in the intermediate layers after layer a are independently sampled from $\mathcal{U}[0, 1]$. For the output layer, the server independently samples $\mathbf{c} \sim \mathcal{N}\left(\mathbf{0}, 10^{-4}\mathbf{I}\right)$ and $\mathbf{u} \sim \mathcal{N}\left(\mathbf{0}, 10^{-4}\mathbf{I}\right)$, and sets $\mathbf{W}^L = \mathbf{c}\mathbf{u}^\top$. It also samples the output-layer biases from a zero-mean Gaussian distribution with variance 10^{-4}. For the King County Housing and Income regression tasks, all weights and biases after the attacked layer, including those of the output layer, are independently sampled from $\mathcal{U}[0, 1]$. These parameters are sampled again at each attack round.

For the CTP baseline, we follow the original implementation of Diana et al. (2025). In all datasets, the attack direction is sampled from $\mathcal{N}(0, 10^{-4})$, and intermediate-layer weights are initialized from the same distribution. Intermediate-layer biases are independently drawn from $\mathcal{N}(0, 10^{-8})$, while the final-layer bias is fixed to $\mathbf{b}^L = 10^{30}$. For regression settings, the output-layer weights are sampled from $\mathcal{N}(0, 10^{-8})$ when required by the original implementation.

For the CAH baseline, attack hyperparameters are selected after a tuning process. Specifically, the random attack directions are sampled from $\mathcal{N}(0, \frac{1}{2})$ for Income, HARUS, and ImageNet, and from $\mathcal{N}(0, 1)$ for King County Housing and CIFAR10. The scale factor is set to $s = 0.3$ for Income and King County Housing, $s = 0.99$ for HARUS and ImageNet, and $s = 0.9$ for CIFAR10.

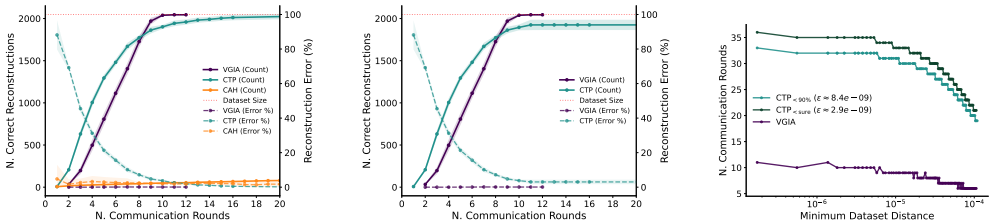
All experiments were conducted on a single NVIDIA L40S GPU equipped with 48 GB of VRAM. On average, reconstructing a complete input batch required approximately 30 seconds for the tabular datasets, 56 seconds for CIFAR10, and 220 seconds for ImageNet.

F Additional Experimental Results

F.1 HARUS

Figure 4 reports the performance of our VGIA attack on the HARUS dataset under the same experimental setting described in Section 5, showing the adaptability of our attack to classification tasks. Consistent with the trends observed in the main paper, after an initial phase in which our attack is slowed down by the verification process, VGIA achieves faster convergence on HARUS, recovering most of the dataset within 12 rounds.

Interestingly, when using the same ε value as in Figure 3b, the resulting performance is almost identical to that observed in Figure 4b. This highlights that tuning ε is a dataset-dependent procedure that requires careful calibration. In practice, without additional prior knowledge, an attacker cannot reliably select an appropriate value, limiting the applicability of CTP in the tabular domain.

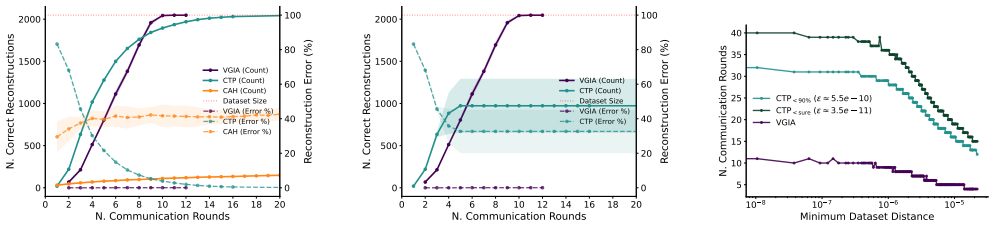


(a) VGIA VS. CTP$_=$ VS. CAH (b) VGIA VS. CTP$_>$ (c) VGIA VS. CTP$_<$

Figure 4: (a,b) Number of correct samples and reconstruction error under varying attack-round budgets on the HARUS dataset. (c) Number of attack rounds required for the adversary to identify all correct samples (i.e., achieve verifiability) under different target datasets with varying $\varepsilon_{\mathbf{w}}$. CTP$_=$ presents a baseline with knowledge of $\varepsilon_{\mathbf{w}}$ (the minimum absolute distance between the projections of any two samples along direction $\mathbf{w}$). CTP$_>$ presents an optimistic choice of ε such that $\varepsilon = 10^{-4} > \varepsilon_{\mathbf{w}}$. CTP$_<$ presents a pessimistic choice of ε such that $\varepsilon < \varepsilon_{\mathbf{w}}$.

Finally, Figure 4c demonstrates the advantage of our adaptive search strategy, which allows our attack to save at least 10 communication rounds when $\varepsilon = \varepsilon_{\mathbf{w}}$.

F.2 King County Housing



(a) VGIA VS. CTP$_=$ VS. CAH (b) VGIA VS. CTP$_>$ (c) VGIA VS. CTP$_<$

Figure 5: (a,b) Number of correct samples and reconstruction error under varying attack-round budgets on the King County Housing dataset. (c) Number of attack rounds required for the adversary to identify all correct samples (i.e., achieve verifiability) under different target datasets with varying $\varepsilon_{\mathbf{w}}$. CTP$_=$ presents a baseline with knowledge of $\varepsilon_{\mathbf{w}}$ (the minimum absolute distance between the projections of any two samples along direction $\mathbf{w}$). CTP$_>$ presents an optimistic choice of ε such that $\varepsilon = 10^{-4} > \varepsilon_{\mathbf{w}}$. CTP$_<$ presents a pessimistic choice of ε such that $\varepsilon < \varepsilon_{\mathbf{w}}$.

The results on the King County Housing dataset confirm the performance of our attack and its adaptability to different data distributions. As shown in Figure 5a, VGIA successfully recovers more than 99% of the dataset within 12 rounds and certifies the reconstruction quality by the subsequent round. In contrast, the CTP baseline is still attempting to isolate samples after 16 rounds.

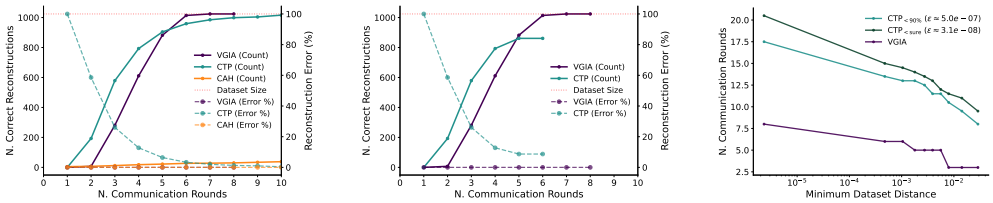
Furthermore, Figure 5b highlights another aspect of the critical dependency of CTP on the choice of ε. Unlike the results on HARUS, here we observe that an optimistic estimation of the separation distance compromises the reconstruction process, resulting in missing more than 50% of the samples. Conversely, Figure 5c illustrates the cost of the opposing condition: when $\varepsilon = \varepsilon_{\mathbf{w}}$, the baseline can require more than 40 communication rounds to verifiably isolate the inputs. In this regime, our adaptive approach saves almost 30 communication rounds compared to CTP.

F.3 ImageNet

We conduct additional experiments on the ImageNet dataset Deng et al. (2009) to demonstrate that our attack maintains the same level of performance even in the presence of very high-dimensional inputs, showing that its effectiveness does not depend on the complexity or dimensionality of the dataset. Due to computational constraints, we evaluate the attack on a batch of $B = 1024$ images randomly sampled from the test set. The results remain consistent with those observed across the

other experimental settings: the attacker is able to recover almost the entire victim dataset, up to negligible numerical errors.

Figure 6b further illustrates that VGIA converges more rapidly to the final solution even when the server selects an overly large value of ε, which prevents effective separation of the data points. For this dataset, setting $\varepsilon = 10^{-2}$ does not substantially hinder the CTP attacker. By contrast, when the server is assumed to have access to the minimum admissible distance ε_{min}, $\text{CTP}_{<\text{sure}}$ requires nearly four times as many rounds as VGIA, while $\text{CTP}_{<90\%}$ requires more than three times as many. Overall, these results highlight the efficiency of VGIA both in navigating the search space and in certifying the correctness of the reconstructed inputs. Results on ImageNet remain consistent



(a) VGIA VS. $\text{CTP}_{=}$ VS. CAH (b) VGIA VS. $\text{CTP}_{>}$ (c) VGIA VS. $\text{CTP}_{<}$

Figure 6: (a,b) Number of correct samples and reconstruction error under varying attack-round budgets on the ImageNet dataset. (c) Number of attack rounds required for the adversary to identify all correct samples (i.e., achieve verifiability) under different target datasets with varying $\varepsilon_{\mathbf{w}}$. $\text{CTP}_{=}$ presents a baseline with knowledge of $\varepsilon_{\mathbf{w}}$ (the minimum absolute distance between the projections of any two samples along direction $\mathbf{w}$). $\text{CTP}_{>}$ presents an optimistic choice of ε such that $\varepsilon = 10^{-2} > \varepsilon_{\mathbf{w}}$. $\text{CTP}_{<}$ presents a pessimistic choice of ε such that $\varepsilon < \varepsilon_{\mathbf{w}}$.

with those observed across all the other experimental settings, demonstrating that VGIA is not tied to a specific data modality and is therefore not restricted to tabular datasets. Our attack completely recovers the victim's dataset faster than $\text{CTP}_{=}$, which requires more than the 8 communication rounds needed by our method.

G Parameter Weight Configuration and Detectability

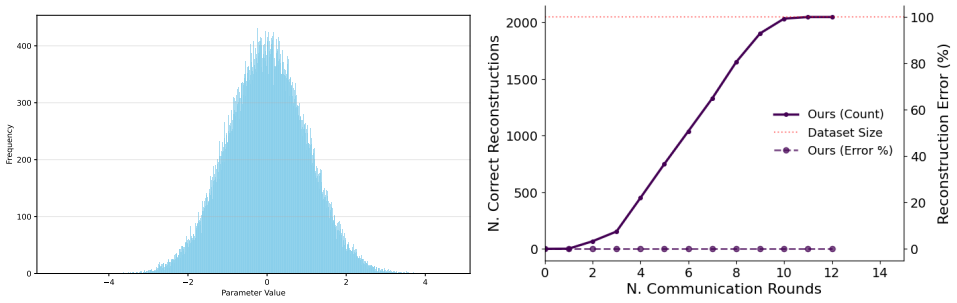
In Sec. 4.2, we explain that the attacker needs to set the intermediate layers of the model, from layer $a+1$ to layer $L-1$, so that the Jacobian entries $\frac{\partial z_i^L}{\partial z_i^a}$ are non-zero and input-independent.

A simple way to enforce input-independent downstream gating, when the layer inputs are non-negative, is to use positive weights and biases in the subsequent layers, so that all downstream pre-activations remain positive and all ReLUs are always active. This is how we set the weights in the experiments reported in the main text. This construction may be highly detectable by the client. We discuss below two stealthier approaches.

G.1 Sampling from Realistic Distributions

In Sec. 4.2, we explain how the attacker can modify the intermediate layers of the model, from layer $a+1$ to layer $L-1$, in order to control the Jacobian entries $\frac{\partial z_i^L}{\partial z_i^a}$ and force them to be non-zero constants.

In practice, this can be achieved by sampling all intermediate-layer parameters from a zero-centered Gaussian distribution and then assigning values in the tails of the distribution to the bias terms, while using values closer to zero for the weights. This construction allows the attacker to enforce a fixed activation pattern across all inputs, with approximately 50% of the neurons remaining always active, thereby preserving a realistic-looking activation behavior and avoiding suspicious parameter configurations. Since the attacker knows the bounds of the input features, it can verify that the desired activation pattern is always satisfied by propagating extreme input values through the network and checking the sign of each pre-activation at every layer.



(a) (b)

Figure 7: (a) Distribution of malicious parameters initialized from $\mathcal{N}(0, 1)$. (b) VGIA performance on the ACS Income dataset with a batch size of 2048 under Gaussian parameter initialization.

We empirically validate this initialization strategy on the ACS Income dataset. Figure 7a shows that the resulting malicious parameters still follow an approximately zero-mean Gaussian distribution, while Fig. 7b shows that this initialization strategy does not affect the reconstruction performance of VGIA.

G.2 Attacking a Subset of the Parameters

In the experiments presented in Sec. 5, the server assigns a search hyperplane to each neuron in the attacked layer. Here, we describe how an attacker can limit parameter modifications to reduce detectability while preserving the gradient structure required for reconstruction. To the best of our knowledge, this is the first work on active-server gradient inversion attacks to explicitly design and evaluate mechanisms for evading client-side detection.

The server can first limit the number of neurons involved in the attack by selecting a subset $\mathcal{P}$ of the N neurons in layer a, with $N_a = |\mathcal{P}| < N$. For each neuron $i \in \mathcal{P}$, the server sets $\mathbf{W}_i^a = \mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^d$ is a random vector shared across the selected neurons, and updates the bias b_i^a according to the VGIA search procedure in Algorithm 1. Neurons outside $\mathcal{P}$ retain their weights and biases, which continue to evolve through ordinary training updates.

The server additionally modifies the connections between layers a and $a + 1$. It selects a neuron κ in layer $a + 1$, which we refer to as the *carrier* neuron, and controls the incoming connections from the attacked neurons in layer a. For each attacked neuron $i \in \mathcal{P}$, the server either removes its connections to all non-carrier neurons in layer $a + 1$, or constrains these weights to be proportional to the corresponding connection to the carrier neuron.

As shown below, this construction factorizes the gradient associated with each attacked neuron into a known neuron-dependent term and a sample-dependent term that is shared across all attacked neurons. After normalization by the known neuron-dependent factor, each sample therefore contributes with the same coefficient to the gradients of all attacked neurons that it activates. Consequently, when the server subtracts the normalized gradients associated with two consecutive hyperplanes, the contributions of samples that activate both neurons cancel, as in Eq. (4). This extends the construction in Eq. (4) without requiring the downstream Jacobian from each attacked neuron to the model output to remain input-independent.

Let layer $a+1$ contain M neurons, such that $\mathbf{W}^{a+1} \in \mathbb{R}^{M \times N}$. The server selects the carrier neuron κ and assigns nonzero weights from every attacked neuron to the carrier,

$$W_{\kappa,i}^{a+1} \neq 0, \qquad \forall i \in \mathcal{P}.$$

At the same time, it removes the connections from the non-attacked neurons in layer a to the carrier by setting $W_{\kappa,i}^{a+1} = 0$ for all $i \notin \mathcal{P}$. The carrier preactivation therefore depends exclusively on the activations of the attacked neurons. The server then chooses the carrier bias b_κ^{a+1} such that the carrier remains active over the entire input domain. Since the server knows the input bounds and all model parameters, it can propagate these bounds through the network to obtain a lower bound on

the carrier preactivation and select b_κ^{a+1} so that this lower bound is strictly positive. Under ReLU activation, the carrier consequently operates in its linear regime for all admissible inputs.

The remaining connections from the attacked neurons in $\mathcal{P}$ to the non-carrier neurons of layer $a+1$ can be constructed in two ways. A simple choice is to set them all to zero. In this case, the influence of the attacked neurons on subsequent layers is routed entirely through the carrier neuron κ. However, this construction introduces a large number of zero-valued parameters, which may make the manipulation easier to detect. A less conspicuous alternative is to constrain these connections to be proportional to the corresponding carrier connections. For each non-carrier neuron $r \neq \kappa$, the server selects a scalar λ_r and sets

$$W_{r,i}^{a+1} = \lambda_r W_{\kappa,i}^{a+1}, \qquad \forall i \in \mathcal{P}.$$

The activation of the carrier neuron is then

$$z_\kappa^{a+1} = \mathrm{ReLU}\left(\sum_{i \in \mathcal{P}} W_{\kappa,i}^{a+1} z_i^a + b_\kappa^{a+1} \right),$$

while, for every non-carrier neuron $r \neq \kappa$,

$$z_r^{a+1} = \mathrm{ReLU}\left(\lambda_r \sum_{i \in \mathcal{P}} W_{\kappa,i}^{a+1} z_i^a + \sum_{i \notin \mathcal{P}} W_{r,i}^{a+1} z_i^a + b_r^{a+1} \right).$$

The downstream Jacobian with respect to an attacked neuron $i \in \mathcal{P}$ can then be written as

$$\frac{\partial z^L}{\partial z_i^a} = \sum_{r=1}^{M} \frac{\partial z^L}{\partial z_r^{a+1}} \mathbb{1}_{\{z_r^{a+1} > 0\}} W_{r,i}^{a+1}.$$

Since the carrier neuron κ is always active and $W_{r,i}^{a+1} = \lambda_r W_{\kappa,i}^{a+1}$ for every $r \neq \kappa$ and $i \in \mathcal{P}$, this becomes

$$\frac{\partial z^L}{\partial z_i^a} = W_{\kappa,i}^{a+1} \left[\frac{\partial z^L}{\partial z_\kappa^{a+1}} + \sum_{\substack{r=1 \\ r \neq \kappa}}^{M} \lambda_r \mathbb{1}_{\{z_r^{a+1} > 0\}} \frac{\partial z^L}{\partial z_r^{a+1}} \right].$$

For a sample $(\mathbf{x}_j, y_j)$, define

$$\rho_j = \frac{\partial \mathcal{L}_j}{\partial z^L} \left[\frac{\partial z^L}{\partial z_\kappa^{a+1}} + \sum_{\substack{r=1 \\ r \neq \kappa}}^{M} \lambda_r \mathbb{1}_{\{z_r^{a+1} > 0\}} \frac{\partial z^L}{\partial z_r^{a+1}} \right]_{\mathbf{x}=\mathbf{x}_j}.$$

Importantly, ρ_j may depend on the sample and on the activation pattern of the downstream network, but it is independent of the attacked neuron i. Hence,

$$\frac{\partial \mathcal{L}_j}{\partial z_i^a} = \frac{\partial \mathcal{L}_j}{\partial z^L} \frac{\partial z^L}{\partial z_i^a} = W_{\kappa,i}^{a+1} \rho_j.$$

Thus, the neuron-dependent term and the sample-dependent term factorize: the dependence on i is entirely captured by the known coefficient $W_{\kappa,i}^{a+1}$, while ρ_j is shared across all attacked neurons activated by sample j.

Accordingly, the slice construction in Eq. (4) becomes

$$\begin{aligned} \mathbf{s}_k &= \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{i+1}^a}}{W_{\kappa,i+1}^{a+1}} - \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i^a}}{W_{\kappa,i}^{a+1}} \\ &= \frac{1}{B} \sum_{j \in \mathcal{A}_{i+1}} \rho_j \mathbf{x}_j - \frac{1}{B} \sum_{j \in \mathcal{A}_i} \rho_j \mathbf{x}_j \\ &= \frac{1}{B} \sum_{j \in \mathcal{A}_{i+1} \setminus \mathcal{A}_i} \rho_j \mathbf{x}_j. \end{aligned} \tag{17}$$

The normalization by $W^{a+1}_{\kappa,i}$ therefore assigns the same coefficient ρ_j to a sample in every attacked neuron that it activates. Consequently, the contributions of samples belonging to both active sets cancel exactly when two consecutive normalized gradients are subtracted.

Similarly,

$$\beta_k = \frac{\frac{\partial \mathcal{L}}{\partial b^a_{i+1}}}{W^{a+1}_{\kappa,i+1}} - \frac{\frac{\partial \mathcal{L}}{\partial b^a_i}}{W^{a+1}_{\kappa,i}} = \frac{1}{B} \sum_{j \in \mathcal{A}_{i+1} \setminus \mathcal{A}_i} \rho_j. \tag{18}$$

If the slice contains a single sample $\mathbf{x}_j$ and $\rho_j \neq 0$, the normalized weight-gradient difference is proportional to $\mathbf{x}_j$, whereas the normalized bias-gradient difference yields the same proportionality coefficient. Their ratio therefore recovers $\mathbf{x}_j$ exactly.

This construction does not require the server to modify any layer beyond layer $a + 1$. It therefore applies to both regression and classification tasks without requiring modifications to the classification layer, as in App. D.1. Moreover, by restricting the manipulation to a smaller subset of model parameters, this construction can make the attack less suspicious.

However, leaving the downstream computation unconstrained introduces two potential limitations. First, the terms defining ρ_j may cancel, resulting in $\rho_j = 0$ and eliminating the contribution of sample j to the observed gradient. Second, the factorization above does not guarantee that the coefficients associated with distinct samples remain nonproportional across communication rounds. Consequently, some of the conditions required by Proposition 4.1 are no longer guaranteed to hold.

Despite the absence of these theoretical guarantees, we did not observe zero coefficients or verification failures attributable to this modification in our experiments or any measurable degradation in reconstruction performance.

G.3 Perturb Attack Directions

Restricting the attack to the neurons in $\mathcal{P}$ reduces the number of modified rows. However, assigning the same direction $\mathbf{w}$ to all attacked neurons still introduces a repeated structure in the attacked layer. To reduce this regularity, the server can apply a small, independent perturbation to the direction of each attacked neuron. This modification affects only the weight vectors of the attacked neurons, while the proportional connections introduced above remain unchanged.

At each attack round t, and for each attacked neuron $i \in \mathcal{P}$, the server samples

$$\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d),$$

and projects it onto the subspace orthogonal to the reference attack direction $\mathbf{w}$. Let

$$\widehat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

denote the normalized attack direction. The orthogonal component is

$$\widetilde{\boldsymbol{\xi}}_i = \boldsymbol{\xi}_i - \left(\boldsymbol{\xi}_i^\top \widehat{\mathbf{w}}\right) \widehat{\mathbf{w}}.$$

The server then normalizes this component as

$$\mathbf{u}_i = \frac{\widetilde{\boldsymbol{\xi}}_i}{\left\|\widetilde{\boldsymbol{\xi}}_i\right\|}.$$

Given a perturbation parameter $p \geq 0$, the server sets the weight vector of each attacked neuron $i \in \mathcal{P}$ to

$$\mathbf{W}^a_i = \|\mathbf{w}\| \frac{\widehat{\mathbf{w}} + p\mathbf{u}_i}{\sqrt{1+p^2}}. \tag{19}$$

The parameter p controls the magnitude of the orthogonal perturbation relative to the reference direction. Since $\mathbf{u}_i^\top \widehat{\mathbf{w}} = 0$ and $\|\mathbf{u}_i\| = 1$, the construction preserves the norm of the original attack direction, and its cosine similarity with $\mathbf{w}$ is

$$\frac{\left(\mathbf{W}^a_i\right)^\top \mathbf{w}}{\|\mathbf{W}^a_i\| \, \|\mathbf{w}\|} = \frac{1}{\sqrt{1+p^2}}.$$

Hence, the angle between each perturbed direction and the reference direction $\mathbf{w}$ is $\arctan(p)$. The weights of neurons outside $\mathcal{P}$ are left unchanged.

For $p > 0$, however, the hyperplanes associated with the attacked neurons are no longer exactly parallel. Consequently, the active sets $\mathcal{A}_i$ and $\mathcal{A}_{i'}$ associated with two consecutive queried neurons are no longer guaranteed to be nested. Their normalized gradient difference therefore becomes

$$\frac{1}{B}\sum_{j\in\mathcal{A}_{i'}\setminus\mathcal{A}_i}\rho_j\mathbf{x}_j - \frac{1}{B}\sum_{j\in\mathcal{A}_i\setminus\mathcal{A}_{i'}}\rho_j\mathbf{x}_j,$$

instead of the single set difference appearing in Eq. (17). In the parallel case, nesting of the active sets guarantees that the second term vanishes. Under direction perturbation, this property is no longer guaranteed, and samples may cross the two hyperplanes in either direction. Therefore, the exact slice identity underlying Eq. (17) is not guaranteed when $p > 0$. The VGIA search continues to select the bias values with respect to the reference direction $\mathbf{w}$. However, the perturbed hyperplanes can induce active sets that differ from those implied by the reference search, so the regions identified during the search do not necessarily coincide with the regions defined by the perturbed neurons. Since the perturbations are independently resampled across communication rounds, this mismatch can vary over time. As a result, reconstruction performance may degrade as the perturbation magnitude p increases. We evaluate this trade-off empirically for different values of p.

G.4 Experiments for Stealthier Attacks

We evaluate the impact of the proposed modifications on both the reconstruction performance and detectability of VGIA. We first study the effect of perturbing the attack directions on reconstruction performance. We then assess detectability through hypothesis testing, evaluating whether state-of-the-art metrics for detecting active gradient inversion attacks (Carletti et al., 2026) can distinguish attacked models from benign ones.

Effect of perturbation. We evaluate different values of the perturbation parameter p, with the results reported in Figure 8. As expected, reconstruction performance gradually decreases as the perturbation magnitude increases. On Income, the fraction of correctly reconstructed samples decreases from 100% for $p = 0$ to 98% for $p = 10^{-6}$ and 66% for $p = 10^{-4}$. Similarly, on CIFAR-10, it decreases from 100% for $p = 0$ to 88% for $p = 10^{-6}$ and 53% for $p = 10^{-4}$. We observe a similar trend across the remaining datasets.

Importantly, even very small perturbations are sufficient to eliminate the rank deficiency induced by the original parallel construction, yielding a full-rank attacked weight matrix.

We next assess whether monitoring the client's training loss is sufficient to detect the attack. We set $N_a = 100$ and let the server retain the client-updated values of all non-attacked parameters across communication rounds, modifying only the parameters required by the attack. As shown in Table 1, with $p = 10^{-7}$ and a fixed budget of 90 communication rounds, VGIA recovers nearly the entire client dataset with negligible reconstruction error while maintaining a full-rank attacked weight matrix. At the same time, the training loss continues to decrease during training, showing that loss monitoring alone does not provide a reliable indicator of the attack.

Detectability metrics We then evaluate the detection metrics proposed by Carletti et al. (2026). These metrics examine the rank of the weight matrix, the variability of the neuron weight vectors, the entropy of the quantized weight values, the ordering of the neuron biases, and the differences between consecutive biases. VGIA can directly avoid the last two tests. The server can assign the biases to arbitrary neuron indices and can scale the weight vector and bias of each attacked neuron by the same factor without changing its hyperplane. We therefore evaluate the matrix rank, neuron variability, and weight entropy. As shown in Fig. 8, the attacked weight matrix has a rank ratio equal to one for all tested perturbation levels and datasets. In particular, with $p = 10^{-7}$, VGIA retains a correctly reconstructed ratio above 95% on all datasets while producing a full-rank weight matrix. Thus, even a small direction perturbation prevents detection based on a rank deficiency without substantially reducing the reconstruction performance. As additional metrics, we consider the mean weight value and the mean pairwise cosine similarity between neuron weight vectors. We evaluate the cosine-similarity test both within individual communication rounds and over the

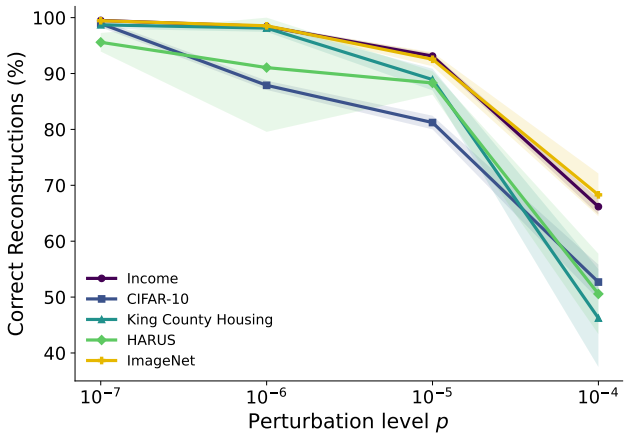


Figure 8: Percentage of correct reconstructions for increasing values of the direction perturbation p. Lines show the mean over three random seeds, and shaded regions show one standard deviation. We use $B = 2048$ for all datasets except ImageNet, for which $B = 1024$. Each client computes one full-batch FedSGD gradient per communication round.

Table 1: Training loss and VGIA correct reconstruction rate across communication rounds. Each cell reports the training loss (top) and the percentage of correct reconstructions (bottom), as mean $\pm$ standard deviation over three random seeds. The attacker uses $N_a = 100$ neurons. Each dataset has $B = 2048$ samples, except for ImageNet, for which $B = 1024$.

Round	Income	King County	HARUS	CIFAR-10	ImageNet
10	$(8.06 \pm 9.87) \times 10^{-1}$ 9.93 ± 0.40	1.06 ± 0.02 10.3 ± 0.45	1.93 ± 0.05 10.2 ± 0.78	2.61 ± 0.06 9.72 ± 0.51	12.9 ± 0.78 20.8 ± 0.64
20	$(4.27 \pm 4.96) \times 10^{-1}$ 22.0 ± 0.57	1.03 ± 0.03 22.2 ± 0.16	1.80 ± 0.05 22.9 ± 0.60	2.48 ± 0.03 22.4 ± 0.38	12.7 ± 0.77 45.4 ± 1.42
30	$(1.89 \pm 1.79) \times 10^{-1}$ 34.6 ± 1.15	1.02 ± 0.03 34.3 ± 0.71	1.71 ± 0.09 34.8 ± 0.50	2.38 ± 0.04 34.8 ± 0.50	12.5 ± 0.76 70.7 ± 3.20
40	$(7.46 \pm 3.76) \times 10^{-2}$ 47.5 ± 0.77	1.00 ± 0.03 46.7 ± 0.66	1.65 ± 0.10 47.1 ± 0.52	2.34 ± 0.03 46.7 ± 0.57	12.1 ± 0.73 96.6 ± 3.57
50	$(3.65 \pm 0.72) \times 10^{-2}$ 59.9 ± 0.59	$(9.82 \pm 0.34) \times 10^{-1}$ 58.6 ± 1.31	1.61 ± 0.10 59.6 ± 0.83	2.31 ± 0.02 58.8 ± 1.07	12.1 ± 0.73 99.9 ± 0.10
60	$(2.54 \pm 0.52) \times 10^{-2}$ 72.0 ± 0.76	$(9.63 \pm 0.38) \times 10^{-1}$ 71.1 ± 1.58	1.56 ± 0.09 71.9 ± 0.20	2.30 ± 0.02 71.1 ± 0.81	–
70	$(2.14 \pm 0.29) \times 10^{-2}$ 84.4 ± 0.59	$(9.43 \pm 0.41) \times 10^{-1}$ 84.0 ± 1.98	1.51 ± 0.07 84.5 ± 0.93	2.30 ± 0.02 83.9 ± 0.73	–
80	$(1.98 \pm 0.16) \times 10^{-2}$ 97.7 ± 1.26	$(9.21 \pm 0.45) \times 10^{-1}$ 96.8 ± 2.74	1.46 ± 0.06 98.0 ± 1.39	2.29 ± 0.02 96.5 ± 0.99	–
90	$(1.78 \pm 0.09) \times 10^{-2}$ 99.8 ± 0.03	$(9.24 \pm 0.36) \times 10^{-1}$ 99.6 ± 0.18	1.45 ± 0.05 99.8 ± 0.06	2.29 ± 0.02 99.8 ± 0.10	–

complete training trajectory, since a client must monitor all the models received during training. Finally, we consider a simple sanity check where a client just tests if the training loss is decreasing across training rounds.

The attacked weight matrix has a rank ratio equal to one for all tested perturbation levels and datasets. As shown in Fig. 8, reconstruction performance gradually decreases as the perturbation magnitude increases. In particular, for $p = 10^{-7}$, VGIA reconstructs more than 95% of the samples on every dataset while maintaining a full-rank weight matrix. Thus, even a very small direction

Table 2: Single-round detection results for weight variability and quantized-weight entropy. For each metric, we report the mean $\pm$ standard deviation for benign and attacked models, the central 95% interval of the empirical benign distribution, and the resulting false-negative rate (FNR). Results are shown for VGIA with one reference direction and with two opposite reference directions, using $N_a = 10$ attacked neurons and perturbation magnitude $p = 10^{-7}$.

Metric	No. of directions	Benign mean $\pm$ std	Benign 95% interval	Attacked mean $\pm$ std	FNR (%)
Weight variability	1	$8.16 \times 10^{-1} \pm 3.60 \times 10^{-4}$	$[8.16 \times 10^{-1}, 8.17 \times 10^{-1}]$	$8.16 \times 10^{-1} \pm 3.77 \times 10^{-4}$	88.16
Weight variability	2	$8.16 \times 10^{-1} \pm 3.60 \times 10^{-4}$	$[8.16 \times 10^{-1}, 8.17 \times 10^{-1}]$	$8.16 \times 10^{-1} \pm 3.76 \times 10^{-4}$	91.07
Weight entropy	1	$3.31 \pm 1.40 \times 10^{-2}$	$[3.27, 3.32]$	$3.31 \pm 1.39 \times 10^{-2}$	95.16
Weight entropy	2	$3.31 \pm 1.40 \times 10^{-2}$	$[3.27, 3.32]$	$3.31 \pm 1.39 \times 10^{-2}$	95.05

perturbation removes the rank deficiency of the parallel construction with only a limited impact on reconstruction performance.

In addition to the metrics of Carletti et al. (2026), we consider the mean weight value and the mean pairwise cosine similarity between neuron weight vectors. We evaluate cosine similarity both within each communication round and cumulatively across the training trajectory, since a client may retain and compare the sequence of models received throughout training. Finally, we consider a simple loss-based sanity check that tests whether the training loss decreases over successive communication rounds.

We evaluate these metrics using hypothesis tests. For each detection statistic, the null hypothesis corresponds to the behavior observed under benign training. We estimate its empirical null distribution by training 100 benign models on the Income dataset for 100 communication rounds using standard PyTorch Kaiming initialization and computing the corresponding detection statistics at each checkpoint. We define the acceptance region as the central 95% interval of the empirical benign distribution, yielding a target false-positive rate (FPR) of 5%, and report the resulting false-negative rate (FNR) on attacked models.

For the attacked models, we set $N_a = 10$ attacked neurons and evaluate each training checkpoint using the same detection procedure. The reference attack direction $\mathbf{w}$ is sampled using Kaiming initialization and perturbed with $p = 10^{-7}$. We additionally evaluate a variant of VGIA that uses two opposite reference directions, $\mathbf{w}$ and $-\mathbf{w}$. Specifically, the server constructs perturbed vectors around $\mathbf{w}$ for

$$N'_a \approx \frac{N_a + \sqrt{N_a}}{2}$$

attacked neurons and around $-\mathbf{w}$ for the remaining $N_a - N'_a$ neurons. This allocation approximately balances the positive and negative contributions of the attacked vectors to the average pairwise inner product, thereby reducing the signal exposed by pairwise-similarity statistics.

Table 2 reports the statistics obtained from the empirical benign distribution and the attacked models, together with the corresponding FNR. With a single attack direction, weight variability yields an FNR of 88.16%, while weight entropy yields an FNR of 95.16%. These results indicate that both statistics provide limited separation between benign and attacked models when evaluated on weights obtained from actual training. Using two independently perturbed opposite directions further increases the FNR of the weight-variability detector to 91.07%. Weight entropy remains essentially unaffected, with an FNR of 95.05%. In the latter case, the corresponding detection rate is approximately 5%, close to the nominal 5% FPR, indicating almost no separation between the benign and attacked distributions.

Table 3 reports the results for the mean pairwise cosine similarity. With a single attack direction, the detector achieves an FNR of 58.50%, indicating that the repeated directional structure remains partially detectable. In contrast, using two opposite directions increases the FNR to 96.00%. The corresponding detection rate of 4.00% is close to the nominal 5% FPR, showing that the two-direction construction almost completely removes the separation between benign and attacked models under this metric.

We next evaluate attack detectability across multiple communication rounds using the mean pairwise cosine similarity computed over a window of T rounds. In this setting, detection becomes more challenging because benign training naturally induces correlations across successive model states, making it difficult to distinguish these patterns from those introduced by the attack.

Table 3: Single-round detection results for the mean pairwise cosine similarity between neuron weight vectors. We report the mean $\pm$ standard deviation for benign and attacked models, the central 95% interval of the empirical benign distribution, and the resulting false-negative rate (FNR). Results are shown for VGIA with one reference direction and with two opposite reference directions, using $N_a = 10$ attacked neurons and perturbation magnitude $p = 10^{-7}$.

No. of directions	Benign mean $\pm$ std	Benign 95% interval	Attacked mean $\pm$ std	FNR (%)
1	$-3.73 \times 10^{-6} \pm 5.24 \times 10^{-5}$	$[-9.43 \times 10^{-5}, 9.02 \times 10^{-5}]$	$7.91 \times 10^{-5} \pm 5.35 \times 10^{-5}$	58.50
2	$-3.73 \times 10^{-6} \pm 5.24 \times 10^{-5}$	$[-9.43 \times 10^{-5}, 9.02 \times 10^{-5}]$	$-1.09 \times 10^{-6} \pm 5.08 \times 10^{-5}$	96.00

Table 4: Cross-round detection results for the cumulative mean pairwise cosine similarity using a single attack direction. A fixed attack budget of 100 attacked neurons is distributed over T communication rounds. We report the average number of attacked neurons per round, the mean $\pm$ standard deviation for benign and attacked trajectories, the central 95% interval of the empirical benign distribution, and the resulting false-negative rate (FNR).

T	N_a per round	Benign mean $\pm$ std	Benign cross-round 95% interval	Attacked mean $\pm$ std	FNR (%)
10	10.0	$8.96 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[8.07 \times 10^{-4}, 9.87 \times 10^{-4}]$	$9.79 \times 10^{-4} \pm 5.37 \times 10^{-5}$	55
15	6.7	$9.29 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[8.40 \times 10^{-4}, 1.02 \times 10^{-3}]$	$9.62 \times 10^{-4} \pm 5.22 \times 10^{-5}$	86
20	5.0	$9.46 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[8.57 \times 10^{-4}, 1.04 \times 10^{-3}]$	$9.62 \times 10^{-4} \pm 5.24 \times 10^{-5}$	90
25	4.0	$9.56 \times 10^{-4} \pm 5.25 \times 10^{-5}$	$[8.67 \times 10^{-4}, 1.05 \times 10^{-3}]$	$9.65 \times 10^{-4} \pm 5.25 \times 10^{-5}$	92
30	3.3	$9.63 \times 10^{-4} \pm 5.25 \times 10^{-5}$	$[8.73 \times 10^{-4}, 1.05 \times 10^{-3}]$	$9.68 \times 10^{-4} \pm 5.25 \times 10^{-5}$	93

Table 4 reports the detection results for the single-direction construction as the attack budget is distributed over an increasing number of communication rounds. When the budget is concentrated over $T = 10$ rounds, the cross-round cosine-similarity test yields an FNR of 55%. Distributing the same budget over a longer horizon substantially reduces detectability: the FNR increases to 86% for $T = 15$, reaches 90% for $T = 20$, and further increases to 93% for $T = 30$. These results show that spreading the attack over more communication rounds makes the attack-induced correlations increasingly difficult to distinguish from those arising during benign training.

The two-direction construction is substantially more effective at reducing detectability, as shown in Table 5. The FNR remains between 90% and 95% across all tested horizons. In particular, for $T = 10$, the FNR increases from 55% with a single direction to 95% with two opposite directions. The corresponding detection rate is therefore 5%, matching the nominal FPR of the test and indicating essentially no separation between benign and attacked trajectories under this metric.

H Defenses Evaluation

We conduct preliminary experiments to evaluate the robustness of our attack under common perturbation mechanisms, focusing on gradient clipping and additive Gaussian noise.

Gradient Clipping Gradient clipping aims to bound the magnitude of gradient updates, potentially limiting the information available for reconstruction. However, from Eq. (4), the effect of clipping can be written as

$$\frac{1}{B} \sum_{\substack{j \in \mathcal{A}_{i+1} \\ j \notin \mathcal{A}_i}} \gamma_j \frac{\partial \mathcal{L}_j}{\partial z^L} \mathbf{x}_j \quad \text{where} \quad \gamma_j = \min(1, \frac{C}{\|\nabla_\theta \mathcal{L}_j\|_2}). \tag{20}$$

Thus, clipping only rescales individual sample contributions. As long as all gradients are clipped, the algebraic structure of the signal is preserved, and the verification condition in Prop. 4.1 remains valid without modification. In practice, an attacker can enforce this regime by increasing the magnitude of the model parameters, ensuring that all per-sample gradients are clipped. To validate this observation, we evaluate VGIA under different clipping thresholds on the ACS Income dataset with batch size $B = 2048$. Results in Table 6 show that the attack performance remains essentially unchanged across clipping levels, achieving near-perfect recovery in all settings.

Gradient Noise We also evaluate the behavior of our attack under Gaussian noise perturbation. In this setting, the server no longer observes exact linear combinations of the inputs, but rather noisy

Table 5: Cross-round detection results for the cumulative mean pairwise cosine similarity using two opposite attack directions. A fixed attack budget of 100 attacked neurons is distributed over T communication rounds. We report the average number of attacked neurons per round, the mean $\pm$ standard deviation for benign and attacked trajectories, the central 95% interval of the empirical benign distribution, and the resulting false-negative rate (FNR).

T	N_a per round	Benign mean $\pm$ std	Benign cross-round 95% interval	Attacked mean $\pm$ std	FNR (%)
1	100	$-4.11 \times 10^{-6} \pm 5.26 \times 10^{-5}$	$[-9.34 \times 10^{-5}, 8.70 \times 10^{-5}]$	$-8.42 \times 10^{-6} \pm 4.90 \times 10^{-5}$	93
2	50	$4.96 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[4.07 \times 10^{-4}, 5.87 \times 10^{-4}]$	$4.69 \times 10^{-4} \pm 4.88 \times 10^{-5}$	90
3	33	$6.63 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[5.74 \times 10^{-4}, 7.54 \times 10^{-4}]$	$6.40 \times 10^{-4} \pm 4.82 \times 10^{-5}$	92
4	25	$7.46 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[6.57 \times 10^{-4}, 8.37 \times 10^{-4}]$	$7.26 \times 10^{-4} \pm 4.85 \times 10^{-5}$	92
5	20	$7.96 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[7.07 \times 10^{-4}, 8.87 \times 10^{-4}]$	$7.79 \times 10^{-4} \pm 4.93 \times 10^{-5}$	91
6	17	$8.29 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[7.40 \times 10^{-4}, 9.20 \times 10^{-4}]$	$8.14 \times 10^{-4} \pm 4.98 \times 10^{-5}$	93
7	14	$8.53 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[7.64 \times 10^{-4}, 9.44 \times 10^{-4}]$	$8.40 \times 10^{-4} \pm 5.04 \times 10^{-5}$	92
8	12	$8.71 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[7.82 \times 10^{-4}, 9.62 \times 10^{-4}]$	$8.59 \times 10^{-4} \pm 5.08 \times 10^{-5}$	93
9	11	$8.85 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[7.96 \times 10^{-4}, 9.76 \times 10^{-4}]$	$8.74 \times 10^{-4} \pm 5.04 \times 10^{-5}$	94
10	10	$8.96 \times 10^{-4} \pm 5.26 \times 10^{-5}$	$[8.07 \times 10^{-4}, 9.87 \times 10^{-4}]$	$8.86 \times 10^{-4} \pm 5.08 \times 10^{-5}$	95

Table 6: Reconstruction performance under different clipping norms on the ACS Income dataset for $B = 2048$.

Clipping Norm	% Reconstructions	L2 Error
10	99.90%	1.10×10^{-25}
5	99.90%	1.10×10^{-25}
2	99.80%	2.90×10^{-26}
1	99.80%	2.90×10^{-26}

versions of them. In particular, (4) becomes:

$$\tilde{\mathbf{s}}_k = \frac{1}{B} \sum_{\substack{j \in \mathcal{A}_{i+1} \\ j \notin \mathcal{A}_i}} \frac{\partial \mathcal{L}_j}{\partial z^L} \mathbf{x}_j + \frac{\eta_{i+1}}{g_{i+1}} - \frac{\eta_i}{g_i}, \qquad \eta_i, \eta_{i+1} \sim \mathcal{N}(0, \sigma^2 I). \tag{21}$$

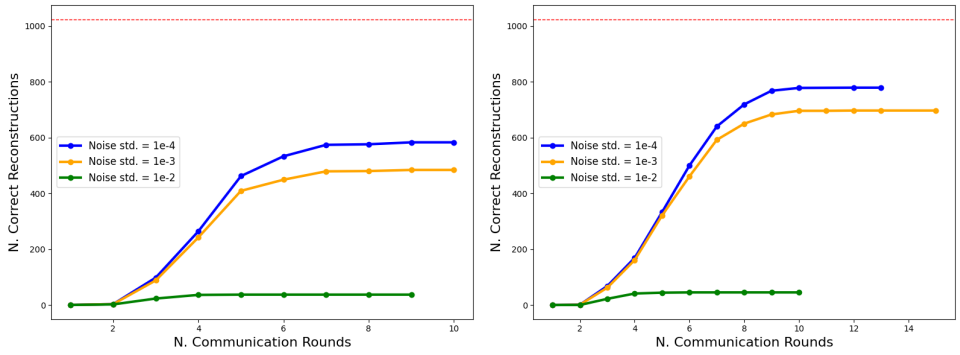
Due to the presence of noise, the span verification mechanism in Prop. 4.1 can no longer be applied exactly, as it is no longer possible to determine whether a slice at round t lies precisely in the span of the observations at round $t + 1$. To address this, we adopt a relaxed verification criterion, similar to the approach used in the FedAvg setting (App. I). Specifically, after projecting a slice onto the span of the candidate subspace (via orthogonalization), we check whether the norm of the residual component exceeds the expected magnitude of the noise. This noise level can be estimated directly by the attacker. In the first communication round, the attacker queries hyperplanes in a region of the input space that is guaranteed to contain no data points, by setting the bias values of the attack layer outside the known bounds of the projected inputs. The resulting slice vectors therefore consist purely of noise, allowing the server to estimate the per-coordinate noise variance σ^2. Assuming isotropic Gaussian noise, the norm of a noise vector in $\mathbb{R}^d$ concentrates around $\sigma\sqrt{d}$, with standard deviation approximately $\sigma/\sqrt{2}$. Accordingly, we define a detection threshold

$$\tau = \sigma\sqrt{d} + c \cdot \frac{\sigma}{\sqrt{2}}, \tag{22}$$

where c is a small constant (e.g., $c = 1$ in our experiments). A slice is deemed to contain a signal component if the norm of its residual after projection exceeds τ, enabling a robust approximate span test in the presence of Gaussian perturbations. We evaluate our attack on the HARUS and CIFAR10 datasets under noise with different standard deviations. Results in Figure 9 show that the attack remains robust across different noise levels. In particular, it successfully recovers up to 584 and 779 samples, respectively, out of a batch of 1024 inputs when attacking a three-layer fully connected neural network with $N = 1000$ neurons.

I Extension to FedAvg

We conduct additional experiments to evaluate the accuracy of our attack under FedAvg. In this setting, the client performs several local optimization steps, and each step evaluates the gradients



(a) HARUS (b) CIFAR10

Figure 9: Number of reconstructed samples under increasing levels of Gaussian noise with variance σ^2. (a) Results on HARUS. (b) Results on CIFAR10.

at different parameter values. The resulting model update is therefore a sum of gradients computed along the local optimization trajectory, with the slice equation (4) no longer being exact. Consequently, an empty slice does not necessarily yield a zero slice vecto, and the span relation used in Proposition 4.1 does not hold exactly.

We account for the first effect by replacing the exact zero-slice test with an absolute norm test. Given the slice $\mathbf{s}_k$, the search considers the corresponding interval as nonempty only if

$$\|\mathbf{s}_k\|_2 > \tau_{\text{empty}}.$$

Small differences are treated as residuals of the Gram-Schmidt orthogonalization process caused by the local updates.

The search accepts the approximate span relation if the norm of the residual is smaller than a fixed threshold τ_{span}.

For multiclass classification, repeated local updates can alter the one-carrier construction described in App. G.2. Although the server initially removes the connections from the attacked neurons to the non-carrier neurons, these connections can become nonzero if the non-carrier neurons receive a gradient during local training. We therefore use the isolated construction

$$W_{r,i}^{a+1} = 0, \qquad r \neq \kappa, \quad i \in \mathcal{P}, \tag{23}$$

and modify the output layer to suppress this gradient during the first local step. In the experiments, we focus on a three-layer fully connected neural network where, for every non-carrier neuron $r \neq \kappa$, the server replaces its output-layer weights with their mean:

$$W_{c,r}^{L} = \frac{1}{C} \sum_{c'=1}^{C} W_{c',r}^{L}, \qquad c = 1, \ldots, C, \quad r \neq \kappa. \tag{24}$$

With this modification, each non-carrier neuron adds the same value to all logits without affecting the softmax probabilities. In particular, if $p_{j,c}$ is the predicted probability of class c for sample j, then

$$\frac{\partial \mathcal{L}_j}{\partial z_{j,r}^{a+1}} = \sum_{c=1}^{C} W_{c,r}^{L} \left(p_{j,c} - \mathbb{1}_{\{c=y_j\}} \right) = \overline{W}_r \left(\sum_{c=1}^{C} p_{j,c} - 1 \right) = 0, \qquad r \neq \kappa. \tag{25}$$

The gradients of the connections entering the non-carrier neurons are therefore zero during the first local step, and the connections $W_{r,i}^{a+1}$ remain zero after this step. However, the output-layer weights can change during the same update. Their values are then no longer guaranteed to be identical across classes, and the zero connections can become nonzero during subsequent local steps. Thus, the FedAvg extension replaces the exact algebraic tests with approximate tests that tolerate projection residuals introduced by local training.

Table 7: Comparison of VGIA and CTP$_=$ attack under FedAvg with different E local epochs and batch sizes on CIFAR-10. The client possesses $n = 1024$ samples in its local dataset. The attack layer of the model has $N = 1000$ neurons. All metrics are reported after 30 attack rounds. Each entry reports the number of unique recovered samples / spurious candidates as mean $\pm$ sample standard deviation across seeds.

E	Method	Batch 128	Batch 256	Batch 512
1	CTP	35.0 ± 21.7 / 9992.7 ± 3016.3	21.7 ± 4.5 / 7627.7 ± 2220.0	62.0 ± 25.9 / 6847.3 ± 3978.0
1	VGIA	385.3 ± 46.6 / 1109.0 ± 440.6	538.3 ± 15.9 / 288.0 ± 30.6	539.3 ± 63.9 / 321.7 ± 229.3
2	CTP	17.7 ± 3.1 / 12756.3 ± 1022.2	19.7 ± 7.5 / 11181.7 ± 3234.1	24.7 ± 16.2 / 7928.0 ± 2557.2
2	VGIA	244.0 ± 32.0 / 1168.7 ± 208.8	313.0 ± 20.0 / 1014.3 ± 170.1	557.7 ± 72.3 / 850.0 ± 183.9
3	CTP	15.0 ± 1.0 / 12165.0 ± 988.9	18.7 ± 9.0 / 10859.7 ± 1639.6	19.0 ± 6.2 / 11597.3 ± 3488.9
3	VGIA	179.0 ± 19.7 / 1021.7 ± 147.1	228.0 ± 10.4 / 929.0 ± 610.2	378.7 ± 31.9 / 786.0 ± 96.9
5	CTP	15.0 ± 2.6 / 11471.3 ± 392.1	12.0 ± 1.7 / 9677.0 ± 296.4	19.0 ± 5.0 / 9678.3 ± 2604.7
5	VGIA	117.3 ± 10.7 / 973.7 ± 316.9	167.3 ± 17.8 / 1329.0 ± 118.9	244.3 ± 15.0 / 640.0 ± 436.4

Table 8: Comparison of VGIA and CTP$_=$ attack under FedAvg with different E local epochs and batch sizes on HARUS. The client possesses $n = 1024$ samples in its local dataset. The attack layer of the model has $N = 1000$ neurons. All metrics are reported after 30 attack rounds. Each entry reports the number of unique recovered samples / spurious candidates as mean $\pm$ sample standard deviation across seeds.

E	Method	Batch 128	Batch 256	Batch 512
1	CTP	49.7 ± 46.3 / 22848.3 ± 4884.9	64.3 ± 62.5 / 19180.0 ± 6734.7	104.7 ± 69.9 / 11617.0 ± 7474.6
1	VGIA	393.3 ± 38.1 / 550.3 ± 106.1	488.7 ± 41.5 / 568.0 ± 266.3	485.0 ± 59.4 / 447.0 ± 471.7
2	CTP	24.0 ± 16.5 / 24413.3 ± 3534.5	45.7 ± 42.3 / 21784.3 ± 4378.5	35.7 ± 30.9 / 18663.7 ± 6602.7
2	VGIA	405.3 ± 48.6 / 984.3 ± 1051.1	481.7 ± 20.6 / 841.7 ± 530.1	487.3 ± 39.6 / 372.7 ± 86.3
3	CTP	16.7 ± 19.4 / 25393.0 ± 4148.3	19.7 ± 14.2 / 24434.3 ± 4071.3	41.7 ± 34.5 / 20794.0 ± 3736.9
3	VGIA	448.3 ± 41.2 / 822.0 ± 614.2	487.3 ± 20.1 / 1106.7 ± 367.2	465.0 ± 49.3 / 799.3 ± 866.7
5	CTP	13.0 ± 12.1 / 25516.3 ± 2331.2	18.7 ± 21.4 / 23645.3 ± 4943.2	31.0 ± 23.6 / 21890.7 ± 4285.7
5	VGIA	477.3 ± 24.7 / 1161.3 ± 477.2	479.7 ± 25.2 / 936.7 ± 348.5	472.3 ± 49.0 / 693.7 ± 740.6

We test the performance under FedAvg on CIFAR10, HARUS, ACS Income, and King County Housing. We set the learning rate to 10^{-4} and consider a local dataset of 1024 samples. We use scaled RMSE to account for differences in feature scale across datasets and consider reconstructions with scaled RMSE below 0.1 as successful, since empirically they showed no perceptible difference from the corresponding original samples. Results in Tables 7, 8, 9, and 10 confirm that, as expected, increasing the amount of local computation degrades the attack's accuracy, as isolating individual samples becomes more challenging.

Nevertheless, VGIA remains effective under FedAvg. Across all tested configurations, it reconstructs between 11% and 72% of the client dataset. In particular, it recovers up to 54% of the samples on CIFAR10, 49% on HARUS, 72% on ACS Income, and 65% on King County Housing. As discussed earlier, multiple local updates introduce parameter-drift-induced signals that interfere with both isolation and verification. As a result, the attack may generate spurious reconstructions, and the same sample can be reconstructed multiple times due to local drift in the search process.

We compare VGIA with CTP's specific adaptation to FedAvg, designed to mitigate hyperplane drift (Diana et al., 2025, Appendix F.1). CTP is substantially less effective under FedAvg, recovering only about 1–10% of the client dataset across the tested configurations. VGIA consistently outperforms CTP, reconstructing at least $7.8\times$, $4.6\times$, $9.6\times$, and $2.4\times$ more correct samples on CIFAR10, HARUS, ACS Income, and King County Housing, respectively. At the same time, CTP produces at least $7.3\times$, $22.0\times$, $286\times$, and $6.1\times$ more spurious candidates, respectively.

J Extension to SiLU and GELU

In this section, we show that, with minor modifications, our attack can be extended to smooth activation functions such as SiLU and GELU, which are commonly used in modern architectures,

Table 9: Comparison of VGIA and CTP$_=$ attack under FedAvg with different E local epochs and batch sizes on ACS Income. The client possesses $n = 1024$ samples in its local dataset. The attack layer of the model has $N = 1000$ neurons. All metrics are reported after 30 attack rounds. Each entry reports the number of unique recovered samples / spurious candidates as mean $\pm$ sample standard deviation across seeds.

E	Method	Batch 128	Batch 256	Batch 512
1	CTP	35.3 ± 7.6 / 253.7 ± 266.1	42.0 ± 14.0 / 773.0 ± 965.2	31.7 ± 8.7 / 64.3 ± 58.7
1	VGIA	742.3 ± 32.0 / 0.0 ± 0.0	879.0 ± 8.7 / 2.3 ± 2.5	900.3 ± 10.0 / 0.0 ± 0.0
2	CTP	33.3 ± 10.0 / 208.7 ± 234.2	36.7 ± 10.5 / 2531.0 ± 3758.7	41.7 ± 17.6 / 200.3 ± 182.8
2	VGIA	523.3 ± 17.6 / 0.0 ± 0.0	801.3 ± 4.9 / 0.0 ± 0.0	888.0 ± 8.5 / 0.7 ± 0.6
3	CTP	33.7 ± 8.5 / 287.7 ± 175.8	38.7 ± 13.0 / 449.7 ± 477.5	36.7 ± 12.7 / 210.7 ± 224.8
3	VGIA	387.7 ± 11.8 / 0.0 ± 0.0	652.0 ± 18.7 / 0.0 ± 0.0	874.3 ± 19.1 / 0.3 ± 0.6
5	CTP	25.7 ± 20.5 / 5659.3 ± 9229.8	20.3 ± 17.2 / 8677.0 ± 12614.5	38.3 ± 15.2 / 308.0 ± 274.7
5	VGIA	248.0 ± 8.2 / 0.0 ± 0.0	448.3 ± 15.1 / 0.0 ± 0.0	743.0 ± 19.1 / 0.3 ± 0.6

Table 10: Comparison of VGIA and CTP$_=$ attack under FedAvg with different E local epochs and batch sizes on King County Housing. The client possesses $n = 1024$ samples in its local dataset. The attack layer of the model has $N = 1000$ neurons. All metrics are reported after 30 attack rounds. Each entry reports the number of unique recovered samples / spurious candidates as mean $\pm$ sample standard deviation across seeds.

E	Method	Batch 128	Batch 256	Batch 512
1	CTP	53.7 ± 15.5 / 8055.7 ± 3540.0	46.7 ± 21.9 / 3020.0 ± 1451.3	39.0 ± 12.1 / 1929.7 ± 331.3
1	VGIA	255.7 ± 36.0 / 684.0 ± 199.9	450.7 ± 17.7 / 498.3 ± 474.5	665.7 ± 32.0 / 0.3 ± 0.6
2	CTP	55.7 ± 10.7 / 7609.3 ± 3555.9	41.7 ± 18.0 / 4336.7 ± 3113.8	43.7 ± 15.4 / 3606.7 ± 2968.8
2	VGIA	188.7 ± 36.9 / 614.3 ± 534.9	360.3 ± 11.2 / 471.7 ± 48.8	640.0 ± 21.9 / 189.0 ± 192.5
3	CTP	50.0 ± 8.9 / 9932.0 ± 1635.6	39.7 ± 14.2 / 6640.3 ± 4444.7	52.3 ± 15.9 / 3695.0 ± 2823.9
3	VGIA	164.0 ± 39.6 / 433.3 ± 423.8	298.7 ± 58.3 / 545.0 ± 574.2	566.0 ± 6.6 / 218.7 ± 87.9
5	CTP	52.0 ± 2.6 / 11488.3 ± 2747.5	35.7 ± 25.4 / 9257.3 ± 4828.7	44.3 ± 22.0 / 3008.0 ± 2308.7
5	VGIA	126.0 ± 46.4 / 377.3 ± 535.3	205.0 ± 62.6 / 219.0 ± 219.5	384.0 ± 66.4 / 102.0 ± 116.9

including Transformer-based models. Our attack, as well as other isolation-based analytical attacks (Fowl et al., 2022; 2023; Boenisch et al., 2023; Shi et al., 2025; Diana et al., 2025), relies on the sparsity induced by the ReLU activation function. In particular, ReLU induces hard activation regions, which enable the exact cancellation of gradient contributions in Eq. (4). For smooth activation functions, this exact sparsity is lost. Samples with negative pre-activation can still contribute to the aggregated gradient, and Eq. (4) no longer holds in its original form. Let ϕ denote a generic differentiable activation function. We have

$$\mathbf{s}_k = \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_{i+1}}}{g_{i+1}} - \frac{\frac{\partial \mathcal{L}}{\partial \mathbf{W}^a_{i}}}{g_{i}} = \frac{1}{B} \sum_{j=1}^{B} \frac{\partial \mathcal{L}_j}{\partial z^L} \left[\phi'(h_{i+1,j}) - \phi'(h_{i,j})\right] \mathbf{x}_j, \tag{26}$$

where

$$h_{i,j} = \mathbf{W}^a_i \mathbf{x}_j + b^a_i. \tag{27}$$

For ReLU, the term $\phi'(h_{i+1,j}) - \phi'(h_{i,j})$ reduces to an indicator of whether $\mathbf{x}_j$ lies between the two hyperplanes. Hence, only samples inside the corresponding slice contribute to $\mathbf{s}_k$. For SiLU and GELU, instead, this difference can be nonzero for every sample in the batch, so even an empty slice may contain a residual contribution.

The attacker first mitigates this effect by increasing the norm of the attack direction $\mathbf{w}$. Since SiLU and GELU behave increasingly like ReLU away from the origin, this scaling moves most pre-activations farther from the smooth transition region and reduces contributions from samples outside the slice. However, residual leakage remains; therefore, we adapt both the search process and the verification mechanism to distinguish it from the signal generated by samples inside the slice.

In the ReLU case, an empty slice can be detected exactly by checking whether $\mathbf{s}_k = \mathbf{0}$. For SiLU and GELU, we instead initialize a leakage calibration set from the slice norms observed during the first attack round. We sort the positive norms in logarithmic scale and identify the largest gap separating the low- and high-norm populations. Exact zero norms and values below this gap are treated as initial empty-slice observations. The split is accepted only if both populations contain sufficiently many observations; otherwise, the corresponding decisions are deferred until additional calibration data become available.

For each empty slice k of width Δ_k, we store its width-normalized leakage

$$\eta_k = \frac{\|\mathbf{s}_k\|_2}{\Delta_k}. \tag{28}$$

The normalization accounts for the first-order dependence of the smooth-activation leakage on the slice width, since, for small Δ, $\phi'(h + \Delta) - \phi'(h) \approx \phi''(h)\Delta$. The normalized leakage can nevertheless vary across the search space because $\phi''(h)$ depends on the pre-activation and is most significant close to the smooth transition region. These calibration observations are retained across communication rounds. To keep the calibration representative as the search progresses, the attacker also reserves a small set of spatially distributed empty intervals as calibration probes. The motivation is that since any subinterval of a known-empty interval is also empty, the additional probing provides more precise estimations at the positions and widths explored throughout the search.

For a candidate slice k, the attacker selects calibration observations with similar positions and widths. Let $\mathcal{N}_k$ denote this local calibration set. The corresponding empty-slice threshold is

$$\tau_k^{\text{empty}} = \Delta_k Q_q\left(\{\eta_\ell : \ell \in \mathcal{N}_k\}\right), \tag{29}$$

where Q_q denotes the empirical quantile of order q. The slice is classified as empty if

$$\|\mathbf{s}_k\|_2 \le \tau_k^{\text{empty}},$$

and as non-empty if

$$\|\mathbf{s}_k\|_2 > 3\tau_k^{\text{empty}}.$$

Slices whose norms fall between these two thresholds are marked as uncertain and remain in the search. This three-way decision prevents the attacker from prematurely discarding or reconstructing slices whose signal cannot be reliably distinguished from smooth-activation leakage.

The verification mechanism must also account for the leakage introduced by smooth activations. Consider a parent interval k at round t. After removing the children classified as empty, let n_k be the number of remaining non-empty child slices at round $t+1$. The attacker collects the non-empty child-slice vectors in

$$\mathbf{S}_k^{t+1} = \begin{bmatrix} \mathbf{s}_{k,1}^{t+1} & \cdots & \mathbf{s}_{k,n_k}^{t+1} \end{bmatrix}. \tag{30}$$

With ReLU, empty children are exactly zero and do not introduce additional directions into the column space of $\mathbf{S}_k^{t+1}$. With SiLU or GELU, instead, smooth leakage can produce small nonzero singular directions even when no sample lies in the corresponding intervals. Retaining these directions would enlarge the estimated child subspace and could make the span test accept an invalid parent interval.

We therefore calibrate the child-subspace threshold against an empirical reference distribution. For parent k, the attacker constructs a collection of null matrices $\mathbf{E}_{k,r}$ with the same dimensions as $\mathbf{S}_k^{t+1}$. Each null matrix contains one known-empty vector for each child column. The vectors are selected from nearby empty intervals and rescaled to the widths of the corresponding children. The attacker constructs at most 64 sibling matrices and 64 resampled matrices, and defers verification if fewer than 16 null matrices are available.

For each null matrix, the attacker records its largest singular value. The SVD threshold is

$$\tau_k^{\text{SVD}} = Q_{0.99}\left(\{\sigma_{\max}(\mathbf{E}_{k,r})\}_{r=1}^{R_k}\right), \tag{31}$$

where R_k is the number of available null matrices. This distribution represents the strongest singular direction that empty-slice leakage can produce. The attacker computes the SVD of $\mathbf{S}_k^{t+1}$ and retains only the left singular vectors whose singular values exceed τ_k^{SVD}. Let $\mathbf{Q}_k$ be an orthonormal basis of the retained directions.

The attacker then calibrates the parent residual against an empirical distribution obtained from locally matched empty-slice vectors. It first requires

$$\left\|\mathbf{s}_k^t\right\|_2 > \tau_k^{\text{empty}}, \tag{32}$$

which prevents a leakage-dominated parent from passing the test. It then computes the absolute parent residual

$$r_k = \left\|\left(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^\top\right)\mathbf{s}_k^t\right\|_2. \tag{33}$$

To calibrate this residual, the attacker selects nearby known-empty vectors, rescales them to the parent width, and projects them onto the same basis. If $\widetilde{\mathbf{e}}_{k,r}$ denotes one such vector, its null residual is

$$r_{k,r}^{\text{null}} = \left\|\left(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^\top\right)\widetilde{\mathbf{e}}_{k,r}\right\|_2. \tag{34}$$

The span threshold is therefore

$$\tau_k^{\text{span}} = Q_{0.99}\left(\left\{r_{k,r}^{\text{null}}\right\}\right). \tag{35}$$

The parent is verified if $r_k < \tau_k^{\text{span}}$, rejected if $r_k > 2\tau_k^{\text{span}}$, and marked as uncertain otherwise.

We evaluate the proposed extension on four datasets: ACS Income and King County Housing for regression, and HARUS and ImageNet for classification. We use the same three-layer fully connected network, replacing the activation function of the attacked layer with either SiLU or GELU, and compare VGIA against CTP. We use a batch size of $B = 2048$ for all datasets except ImageNet, for which $B = 1024$. As shown in Figure 10, VGIA successfully adapts to both smooth activations, recovering more than 97% of the samples in all settings while maintaining a reconstruction error below 1%. In contrast, CTP fails to reconstruct any input across all datasets and activation functions.

K Comparison with the Scale-MIA attack on ResNet and Transformer

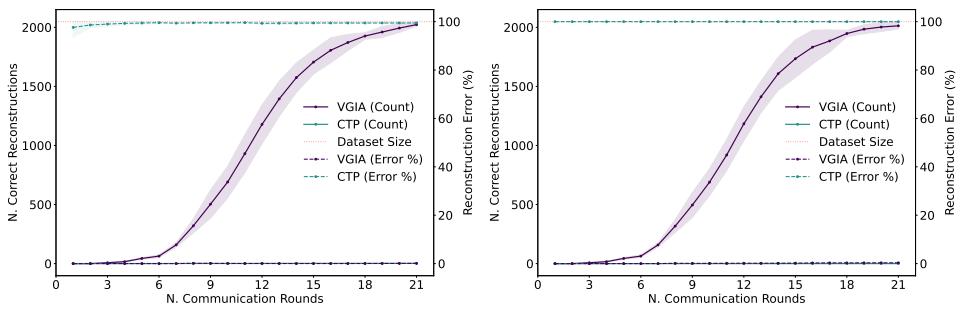
To demonstrate that our attack generalizes to different model architectures under assumptions comparable to prior malicious-server attacks, we compare it against the state-of-the-art method Scale-MIA (Shi et al., 2025) on Transformer and ResNet architectures. We adopt the same threat model as the baseline, assuming that the server has access to an auxiliary dataset $\mathcal{D}_{aux}$ drawn from a distribution similar to the training data, and that the attacker is allowed to modify the model architecture.

At a high level, Scale-MIA leverages the linear leakage block introduced by Fowl et al. (2022) to disentangle gradient contributions in a latent space, thereby recovering input embeddings. These embeddings are then passed through a decoder, trained on $\mathcal{D}_{aux}$, to reconstruct the corresponding inputs.

Using the authors' implementation[1] and the provided encoder and decoder weights, we evaluate our attack on both ViT and ResNet architectures and compare it with Scale-MIA on CIFAR10. In all experiments, the attack layer has $N = 512$ neurons.

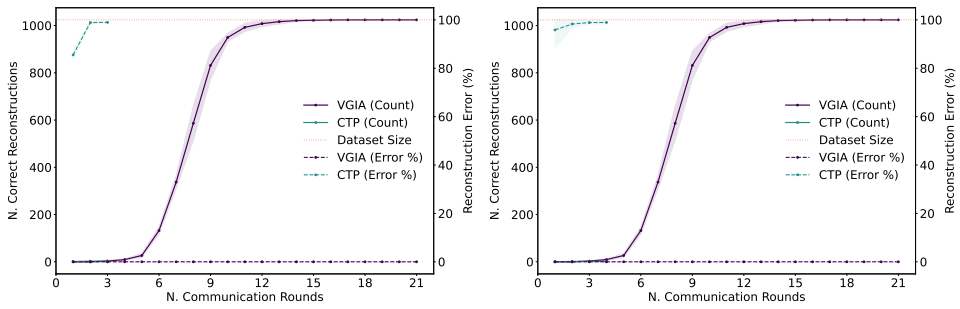
Results in Table 11 show that VGIA correctly isolates more than 99% of the original images in the embedding space for both architectures, achieving over a $6\times$ improvement compared to the baseline. In terms of reconstruction quality, VGIA improves SSIM by more than 160% on ResNet and over 20% on ViT. Similarly, PSNR increases by more than 20% on ResNet and over 70% on ViT, consistently outperforming the baseline. VGIA also increases the proportion of samples with PSNR > 18, the heuristic threshold used in (Shi et al., 2025) to determine reconstruction success. However, this criterion remains arbitrary and does not reliably reflect true recovery. To support this claim, Figure 11 presents examples of baseline reconstructions with PSNR > 18 that would be considered successful under this metric. In practice, these outputs often correspond to pure noise for ResNet and low-quality images for ViT, lacking meaningful semantic content. Such artifacts typically arise from linear combinations of multiple samples rather than accurate reconstructions. In contrast, VGIA provides a principled and automatic verification mechanism by certifying the isolation of individual embeddings. This yields a more meaningful measure of attack success, without relying on heuristic thresholds, human inspection, or access to the original private data.

[1] https://github.com/unknown123489/Scale-MIA



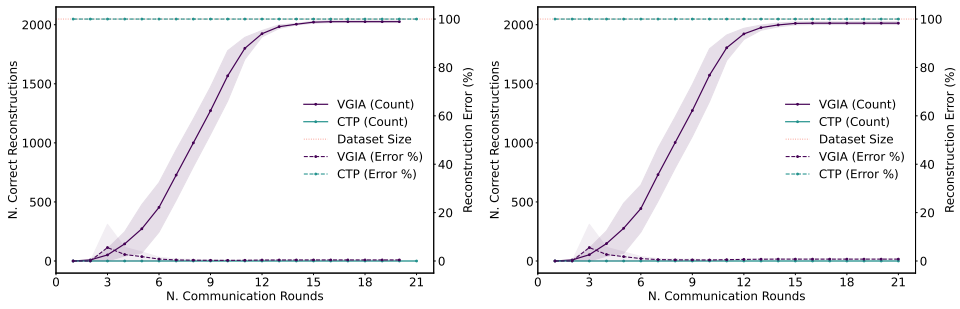
(a) Income-GELU

(b) Income-SiLU



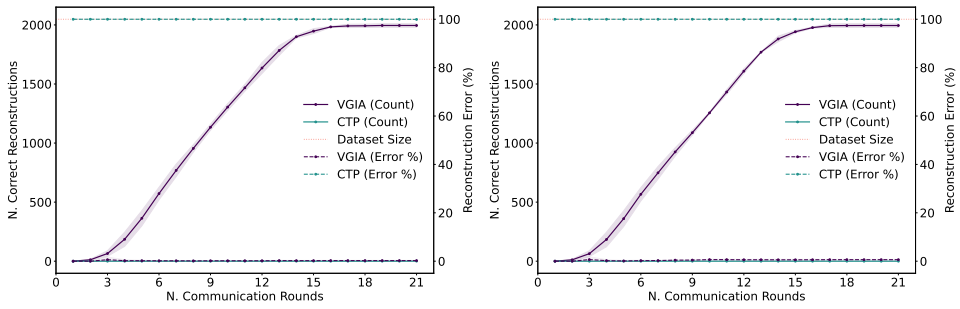
(c) ImageNet-GELU

(d) ImageNet-SiLU



(e) HARUS-GELU

(f) HARUS-SiLU



(g) King County-GELU

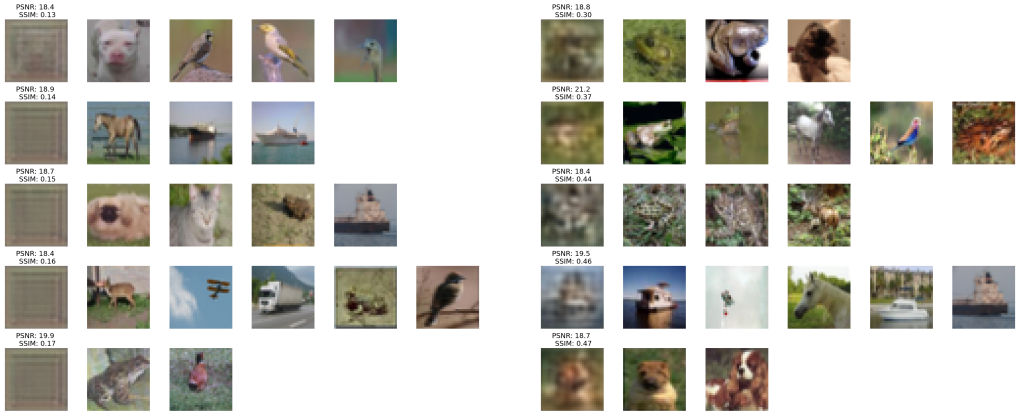
(h) King County-SiLU

Figure 10: Number of correct reconstructions and reconstruction error under varying attack-round budgets. (a-c-e-g) Attack on a fully connected layer with GELU activation. (b-d-f-h) Attack on a fully connected layer with SiLU activation.

Finally, Figure 12 reports performance as the batch size increases, for $B \in \{128, 256, 512, 1024, 2048\}$. VGIA remains stable and consistently outperforms the baseline across all batch sizes, indicating that its search procedure is largely independent of the number of

Table 11: Attack performance on ResNet and Transformer models on CIFAR10 for $B = 1024$.

	Isolated (%)		PSNR $\geq$ 18 (%)		Avg PSNR (↑)		Avg SSIM (↑)	
Model	VGIA	Scale-MIA	VGIA	Scale-MIA	VGIA	Scale-MIA	VGIA	Scale-MIA
ResNet	**99.21**	13.96	**28.13**	3.03	**16.07**	12.90	**0.80**	0.30
ViT	**99.71**	15.03	**86.23**	13.18	**20.16**	11.50	**0.85**	0.70



(a) ResNet (b) ViT

Figure 11: Reconstruction examples from the Scale-MIA (Shi et al., 2025) attack on a batch of $B = 1024$ images from CIFAR10. The left column shows the decoded image with the corresponding PSNR and SSIM values. The other columns show the corresponding true images isolated in the corresponding bins. In both figures, according to the proposed evaluation metric, the attack is claimed to be successful ($PSNR > 18$). (a) Examples of non-correct isolations from a ResNet model. (b) Examples of non-correct isolations from a ViT model.

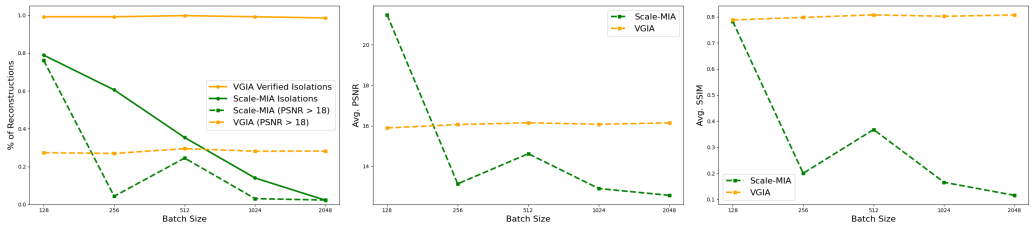
samples in the batch. In contrast, the performance of Scale-MIA degrades as the batch size grows. Figure 13 provides our representative reconstruction examples.

L Reconstruction Examples

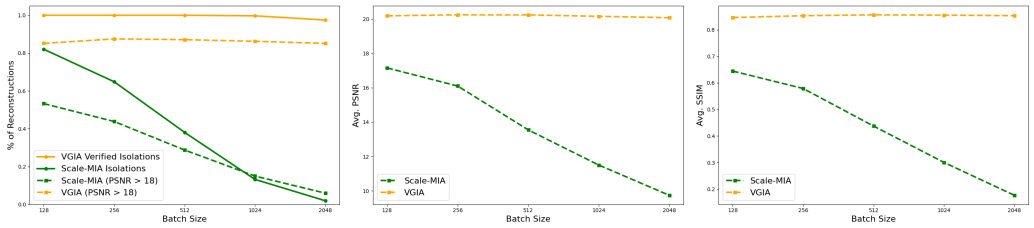
Here we provide reconstructed examples of VGIA on all the tested datasets. See Tables 12, 13, 14 for tabular datasets and Figures 14, 15 for image datasets.

Table 12: VGIA Reconstructions on the ACS Income dataset for a batch of 2048 samples.

Example	Sample	AGEP	SCHL	WKHP	SEX	COW	MAR	RELP	RAC1P	ST
1	Rec.	58	19	40	2	4	5	0	3	4
1	True	58	19	40	2	4	5	0	3	4
2	Rec.	58	11	45	1	1	5	0	8	4
2	True	58	11	45	1	1	5	0	8	4
3	Rec.	30	21	50	1	5	5	0	1	4
3	True	30	21	50	1	5	5	0	1	4
4	Rec.	26	22	45	2	4	5	0	1	4
4	True	26	22	45	2	4	5	0	1	4



(a) ResNet

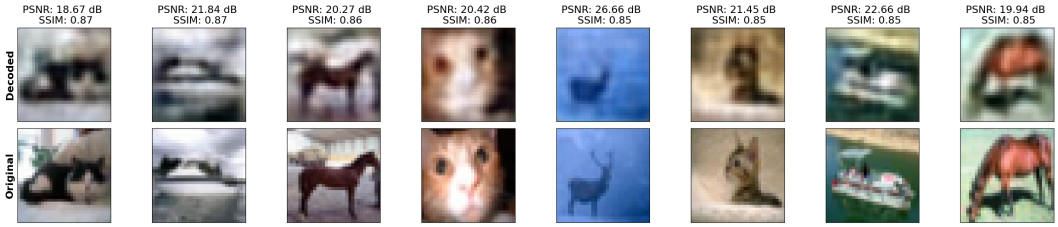


(b) ViT

Figure 12: Comparison of VGIA and Scale-MIA across different batch sizes. Each row corresponds to a model architecture, while columns report different evaluation metrics. From left to right: (i) percentage of correctly reconstructed samples (according to each method's criterion), (ii) average PSNR of the decoded inputs, and (iii) average SSIM of the reconstructed samples. (a) Results on ResNet. (b) Results on ViT.



(a) ResNet



(b) ViT

Figure 13: Examples of reconstructed images from VGIA on a batch of $B = 1024$ CIFAR10 samples.

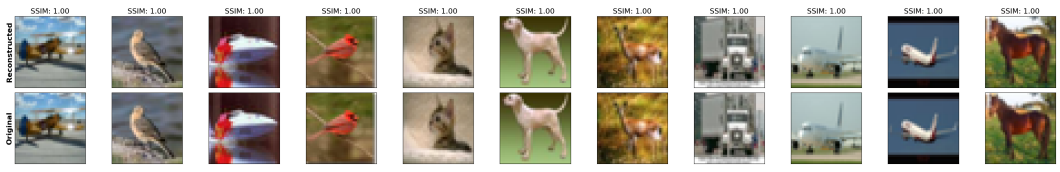


Figure 14: VGIA reconstruction examples on a batch of $B = 2048$ images from CIFAR10.

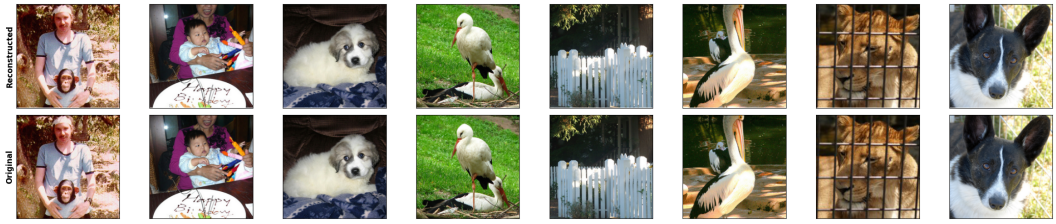


Figure 15: VGIA reconstruction examples on a batch of $B = 1024$ images from ImageNet.

Table 13: First 5 features from HARUS reconstructed samples from a batch of $B = 2048$ records.

Example	Sample	tBody-mean-X	tBody-mean-Y	tBody-mean-Z	tBody-std-X	tBody-std-Y
1	Rec.	0.62	-0.0081	-0.066	0.79	0.42
1	True	0.62	-0.0081	-0.066	0.79	0.42
2	Rec.	0.27	-0.032	-0.13	-0.28	-0.12
2	True	0.27	-0.032	-0.13	-0.28	-0.12
3	Rec.	0.33	-0.014	-0.075	0.65	0.53
3	True	0.33	-0.014	-0.075	0.65	0.53
4	Rec.	0.25	-0.026	-0.21	0.66	0.55
4	True	0.25	-0.026	-0.21	0.66	0.55

Table 14: Reconstruction examples from a batch of 2048 samples of the KCH dataset. The table shows 8 of the 18 features of the dataset.

Example	Sample	Bedrooms	Bathrooms	Sqft living	Condition	Grade	Sqft above	Sqft basement	Year built
1	Rec.	6	4.75	5770	3	12	4500	1270	1998
1	True	6	4.75	5770	3	12	4500	1270	1998
2	Rec.	4	2.50	3350	3	10	1760	1590	1995
2	True	4	2.50	3350	3	10	1760	1590	1995
3	Rec.	2	1.75	2260	3	9	1890	370	2005
3	True	2	1.75	2260	3	9	1890	370	2005
4	Rec.	5	2.50	3490	4	9	1840	1650	1958
4	True	5	2.50	3490	4	9	1840	1650	1958