QBI: Quantile-based Bias Initialization for Efficient Private Data Reconstruction in Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated learning enables the training of machine learning models on distributed data without compromising user privacy, as data remains on personal devices and only model updates, such as gradients, are shared with a central coordinator. However, recent research has shown that the central entity can perfectly reconstruct private data from shared model updates by maliciously initializing the model's parameters. In this paper, we propose QBI, a novel bias initialization method that significantly enhances reconstruction capabilities. This is accomplished by directly solving for bias values yielding sparse activation patterns. Further, we propose PAIRS, an algorithm that builds on QBI. PAIRS can be deployed when a separate dataset from the target domain is available to further increase the percentage of data that can be fully recovered. Measured by the percentage of samples that can be perfectly reconstructed from batches of various sizes, our approach achieves significant improvements over previous methods with gains of up to 50% on ImageNet and up to 60% on the IMDB sentiment analysis text dataset. Furthermore, we establish theoretical limits for attacks leveraging stochastic gradient sparsity, providing a foundation for understanding the fundamental constraints of these attacks. We empirically assess these limits using synthetic datasets. Finally, we propose and evaluate AGGP, a defensive framework designed to prevent gradient sparsity attacks, contributing to the development of more secure and private federated learning systems. Our code will be made open-source.

1 Introduction

The proliferation of mobile devices and the Internet of Things has led to an unprecedented amount of data being generated at the edge of the network [1]. This data, often in the form of user-generated content, sensor readings, or other types of user interactions, holds immense value for training machine learning (ML) models [2]. Due to the sensitive and often private nature of this data, traditional ML approaches that rely on centralized data collection and processing are often inadequate from a legal or ethical perspective [3]. Furthermore, regulations such as data sovereignty laws and cross-border data transfer restrictions (e.g., GDPR, CCPA) can hinder the movement of data between jurisdictions [4, 5].

Federated learning (FL) was proposed as a solution to these challenges [6]. It enables the collaborative training of ML models while preventing the need for users' data to leave their devices – each device computes model updates locally, which are then sent to a central entity for aggregation into a shared model. FL, in theory, should preserve users' privacy and adhere to data transfer restrictions, as the computed gradient updates should not expose any user data.

However, a large body of prior work has demonstrated that the FL protocol is vulnerable to multiple forms of data reconstruction attacks, which can be roughly classified into two major categories.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Passive gradient leakage attacks In this scenario, the attacker is assumed to have no control over the model's architecture, its parameters, or the specific FL protocol that is being used. The gradients created through a standard FL protocol could either be obtained by an *honest but curious* central entity, or an external actor via a man-in-the-middle (MITM) attack. A long line of work has demonstrated, that simply by obtaining these gradients, an attacker could learn about properties of the data [7, 8], data membership [8], and even partially reconstruct user data [9–11].

Malicious model modifications The second major threat model assumes a malicious central entity, that has the ability to modify the model's architecture or parameters in an attempt to break the user's privacy. A broad range of research has outlined a multitude of possible active attacks, that often leverage induced gradient sparsity [12, 9, 13] or sample disaggregation [14, 15] to recover user data.

Each of these approaches strikes a different balance between computational overhead, quality of the reconstructed data, client-side detectability and the requirement for knowledge about the user data, including certain features and their distributions.

Efficient perfect user data reconstruction with bias tuning In this work, we propose a novel method of maliciously initializing a model, by only modifying the bias values of a fully connected layer, while leaving the weights completely randomly initialized. This reduces client-side detectability in parameter space compared to methods that either maliciously train the full model to compromise privacy [16] or introduce a shift in the magnitude of all weight values [12]. Additionally our method removes the need for experimentally determined hyperparameters [12] or handpicked target features or classes [17]. We provide two variants of our approach: *Quantile-Based Bias Initialization* (QBI), which directly determines the optimal bias with near-zero computational cost, and *Pattern-Aware Iterative Random Search* (PAIRS) which builds on QBI, but further enhances reconstruction success by incorporating auxiliary data and incurring marginally higher computational overhead. Additionally, we derive boundaries for the expected success of attacks leveraging stochastic gradient sparsity, which enables us to identify the inherent constraints of such attacks. Using the findings described in this paper, we propose a novel defensive framework – AGGP – aimed at preventing gradient sparsity attacks, contributing to the development of more secure and private FL systems.

Contributions

- We establish theoretical limits for attacks leveraging stochastic gradient sparsity and empirically assess these limits using synthetic datasets.
- We propose a novel, compute efficient method of maliciously initializing fully-connected layers in a server-side attack on the FL protocol, which achieves state-of-the-art results in the domain of perfect user data reconstruction.
- We provide two variants of our approach: QBI which can be deployed with near-zero computational cost, and PAIRS which can be used to achieve increased performance if auxiliary data and increased compute is available. We plan to release an open-source implementation of our method.
- We extensively evaluate both QBI and PAIRS and find that we achieve improvements above the previous state-of-the-art in perfect rectonstruction with gains of up to 50% on ImageNet and up to 60% on the IMDB sentiment analysis text dataset.
- We propose and evaluate AGGP, a novel and compute efficient *defensive framework* that prevents data leakage from both active and passive attacks that leverage gradient sparsity in fully connected layers.

2 Background

Federated learning FL is a decentralized approach to machine learning that enables multiple parties to collaboratively train a shared model on their local data without sharing the data itself. Each client has a local dataset and computes an update to the model parameters based on their local data. The update is typically computed as the gradient of the local loss function with respect to the model parameters. The local updates are then aggregated to update the global model parameters. One common aggregation method is federated averaging, which computes the weighted average of the local updates. This process is repeated over multiple rounds to train the global model [6].

Gradient sparsity attacks As demonstrated by Geiping et al. [18], it is feasible to extract a single input from the gradients of a fully-connected layer, given that the layer is preceded only by fully connected layers, has a bias b, uses the ReLU activation function, and the gradient of the loss with respect to the layer's output contains at least one non-zero entry (see Proposition D.1 in Geiping et al. [18] for a full proof). If this layer is placed at the beginning of the network, this corresponds to reconstructing the original input data point x. As shown in Section 5.1 of Boenisch et al. [12], for any non-zero output y_i, the gradient of the corresponding weight row directly contains the input scaled by the gradient of the loss with respect to the bias. In practice, gradients are typically computed as the average over an entire batch of samples. Since a single neuron is often activated by multiple samples, the gradients of its weight row contain the average of multiple input data points, effectively obscuring them and preventing individual extraction. To extract an input sample x, there has to exist a neuron n_i such that $L(x)_i > 0$ and $L(x')_i < 0$ for all $x' \neq x$, where $L(x)_i$ denotes the activation of the i-th neuron for sample x. Since the samples that are to be extracted are unknown, initializing a layer to achieve this specific activation pattern is challenging. Therefore, the goal of stochastic gradient sparsity attacks is to increase the probability that a neuron is activated only by a single sample, while producing a variety of neurons with diverse activation patterns, to capture as much data from the target domain as possible.

Adaptability to CNN-based architectures Boenisch et al. [12] extend the gradient sparsity attack to Convolutional Neural Networks (CNNs), where one or more convolutional layers precede the first linear layer. By utilizing zero-padding, a stride of one, and maliciously initialized filters, convolutional layers can effectively be transformed into identity functions, which perfectly transmit the inputs deeper into the network, allowing them to be extracted once they pass through a fully connected layer. For a detailed description, including visualizations, see Appendix B of Boenisch et al. [12].

3 Related Work

In FL, a common threat model is the passive gradient leakage scenario, where a malicious entity obtains a set of gradients to recover the original data points. Previous research has demonstrated that these gradients can be used to infer data properties [7, 8] or data membership [8]. Several optimization-based attacks have been proposed [19, 11, 20], which optimize a batch of random noise to generate gradients similar to those observed and thereby achieve partial reconstruction of user data. Although applicable to various architectures, these methods often require significant computational resources and are unable to achieve perfect reconstructions. The second threat model involves a malicious server (MS) that can manipulate the model's architecture or parameters to compromise user privacy. Research in this area has identified various active attacks, which exploit induced gradient sparsity [12, 9, 13] or sample disaggregation techniques [14, 15] to recover user data. The SEER framework [14] is a notable example of an MS that avoids client-side detectability in gradient space by disaggregating samples in an embedding space that is unknown by the client. Although it achieves a high percentage of well-reconstructed images, it does so at a high computational cost (14 GPU days to train on an A100 with 80GB) and falls short of perfectly reconstructing data.

We mainly build on the work of Boenisch et al. [12] that introduced the concept of *trap weights*, a computationally efficient way to initialize a model to induce gradient sparsity. By adding a slight negative shift to the weight values, their approach achieves perfect reconstruction on multiple datasets. However, their method relies on an experimentally determined scaling factor. In contrast, we automate the model's initialization process and achieve substantially higher rates of perfect reconstruction.

4 Method: Adversarial Bias Tuning

Intuition of bias tuning Given a linear layer L of shape $N \times M$, that uses the ReLU activation function, and a batch $\vec{X}$ of B samples $x_1, \cdots, x_B$, the objective is that for every x there exists one and only one neuron n_i such that $L(x)_i > 0$ and $L(x')_i < 0$ for all $x' \neq x$, where $L(x)_i$ denotes the activation of the i-th neuron for sample x. This ensures that neuron n_i allows for perfect reconstruction of x from the gradients of its weight row, while producing zero gradients for all other samples. Therefore, for every neuron n_i the desired probability to activate for any sample in the batch is $1/B$. We take the model's weights and our approximated normalized features to be independently

and identically distributed (i.i.d.) random variables sampled from a normal distribution:

$$\begin{pmatrix} \mathbf{w_i} \\ \mathbf{x_i} \end{pmatrix} \sim \mathcal{N}(0, I_M) \tag{1}$$

Using this assumption, the probability of a single neuron, with corresponding weight row w_i and bias b_i, activating for a sample $x_i \in \mathbb{R}^M$ can be expanded as:

$$P\big(L(x_i) > 0\big) = P\big(w_i^T x_i + b_i > 0\big) = P\big(w_{i1}x_{i1} + \ldots + w_{iM}x_{iM} + b_i > 0\big) \tag{2}$$

Now the left-hand side is a sum of i.i.d. random variables, each of which has characteristic function:

$$\big(1 + t^2\big)^{1/2} \tag{3}$$

In turn, the characteristic function of the entire sum is:

$$(1 + 2it)^{-M/2} = (1 - 2it/2)^{-M/2} \cdot (1 + 2it/2)^{-M/2} \tag{4}$$

This immediately identifies it as distributed like:

$$P\big(w_i^T x_i + b_i > 0\big) = P\big(Q_1/2 - Q_2/2 \leq b_i\big) \tag{5}$$

where Q_1 and Q_2 are independent and both of the chi-square distribution with M degrees of freedom. Note that the variance of each addend is $V(w_{ij}x_{ij}) = 1$ (derived by evaluating the second derivative of Equation (3) at 0). We can drop the assumption that our features are normally-distributed and instead assume that the data has been normalized, as we do in our experiments. Even with this change, the variance of each addend still evaluates to 1 as we maintain the assumption that the weights are independently- and normally-distributed. This enables us to apply the Central Limit Theorem to the original sum in Equation (2). As a sum of i.i.d. random variables with existing second moments, we can make a statement about convergence in distribution:

$$P\big(w_{i1}x_{i1} + \ldots + w_{iM}x_{iM} + b_i > 0\big) \xrightarrow{d} \Phi\left(\frac{b_i}{\sqrt{M}}\right) \tag{6}$$

where Φ is the cumulative distribution function of a standard normal distribution. Meaning we can solve for the asymptotically optimal bias b_* by using the inverse cdf or quantile function:

$$b_* = \Phi^{-1}(\frac{1}{B}) \cdot \sqrt{M} \tag{7}$$

It can be helpful to conceptualize the relationship between this linear layer and a set of samples as a bipartite random graph. The nodes of one partition class are neurons from the neuron set $\tilde{N} = \{n_1, \ldots, n_n\}$, and those in the other partition are from the sample set $\tilde{X} = \{x_1, \ldots, x_B\}$. A vertex $(n_i, x_j) \in \tilde{N} \times \tilde{X}$ is said to exist if n_i is activated by x_j. If the malicious actor wants to reconstruct a sample x_j, that sample needs to have a neighbor n_i with degree $\delta(n_i) = 1$.

Extraction metrics We closely follow the extraction metrics proposed by Boenisch et al. [12]. Given a linear layer with N output neurons and a batch $\tilde{X}$ of B samples $x_1, \cdots, x_B$, we define the following metrics:

1. **Active neurons** (A): This metric represents the percentage of neurons in layer L that activate for at least one of the B samples. Formally, let N_A be the number of neurons n_i that satisfy the condition that there exists at least one input x in $\tilde{X}$ such that $L(x)_i > 0$, where $L(x)_i$ denotes the activation of the i-th neuron in L for sample x. For the neuron conceptualized as a graph node, this condition is equivalent to $\delta(n_i) \geq 1$. The active neurons metric A is therefore defined as the ratio of N_A to the total number of neurons N:

$$A = \frac{N_A}{N} \tag{8}$$

2. **Extraction-Precision** (P): This metric measures the percentage of neurons that allow for the extraction of individual data points. Specifically, let N_u be the number of neurons n_i in layer L with unique activations, i.e. those that satisfy the following condition: $L(x)_i > 0$ for one input x in $\tilde{X}$, and $L(x')_i < 0$ for all other inputs $x' \neq x$. This condition is equivalent to $\delta(n_i) = 1$, which effectively counts neuron nodes that are leaves of their graphs. The extraction-precision P is defined as the following ratio:

$$P = \frac{N_u}{N} \tag{9}$$

3. **Extraction-Recall** (R): The extraction-recall measures the percentage of input data points that can be perfectly reconstructed from any gradient row. Let B_0 be the number of data points that can be extracted with an l_2-error of zero, then R is denoted as:

$$R = \frac{B_0}{B} \tag{10}$$

Notably, R is the most significant metric, as neither A nor P can be used in isolation to meaningfully assess the effectiveness of an attack. A high A value could lead to overlapping activations that prevent individual extraction, while a high P value could be observed in a scenario where all neurons activated for the same sample. If the true probability of a neuron activating is $1/B$, we can derive the explicit probabilities $p_{A;B}$ and $p_{u;B}$ for a neuron to be counted as a success in the context of the A and P metrics, respectively. Since the success of one neuron does not influence the remaining neurons, the entire batch follows a binomial distribution: $N_A \sim \mathcal{B}(N, p_{A;B})$ and $N_u \sim \mathcal{B}(N, p_{u;B})$. Consequently, the expected activation share A and the expected extraction-precision P are:

$$p_{A;B} = \mathbb{E}[A_B] = 1 - \left(\frac{B-1}{B}\right)^B \qquad \text{and} \qquad p_{u;B} = \mathbb{E}[P_B] = \left(\frac{B-1}{B}\right)^{B-1} \tag{11}$$

For growing batch sizes, these converge to:

$$\lim_{B\to\infty} \mathbb{E}[A_B] = 1 - \frac{1}{e} \approx 63.2\% \quad \text{and} \quad \lim_{B\to\infty} \mathbb{E}[P_B] = \frac{1}{e} \approx 36.8\% \tag{12}$$

From a graph theory perspective, $R \cdot B = B_0$ denotes the size of the largest neuron subset $V \subset \tilde{N}$, such that the subgraph induced by the union of V and all neighbors of vertices in V forms a perfect matching. However in contrast to the previous metrics, the expected extraction recall R exhibits a crucial difference. Our assumptions state that existence of a particular edge (representing the activation of a neuron by a data point) is independent of the existence of any other edges. Due to this, and because neurons do not have edges in common with one another, the event of inclusion of any neuron in the count for N_A and N_u is also independent of the inclusion of any other neuron therein. This allows us to assert the success probabilities (see Equation (11)) and that N_A and N_u will both follow a binomial distribution. This binomial approach breaks down for R, however. We can and will still derive the non-zero probability (see Equation (13)) for one specific data point to be included in the count (for N_A and N_u we were counting neurons, for B_0 we count data points). This value will depend on the size of both partition classes of the graph, not just the one being counted. Namely, the expected share of data points that the malicious actor can perfectly reconstruct is:

$$p_{R;B;N} = \mathbb{E}[R] = 1 - \left(1 - \frac{1}{B}\left(\frac{B-1}{B}\right)^{B-1}\right)^N \tag{13}$$

See Appendix B.1 for a more detailed explanation. To check that this does not yield the exact distribution of R and that the binomial approach is no longer relevant, consider a graph with more data points than neurons, i.e. $B > N$. It is clear that $P(R = 100\%) = 0 \neq (p_{R;B;N})^B$, since there are not enough neurons to cover each data point. As Equation (13) assumes the optimal scenario, only obtainable with per-neuron activation probabilities of $1/B$ and truly normally distributed data, it provides an upper limit for the expected success of stochastic gradient sparsity attacks on real-world data, which will yield lower expected results, the further the data deviates from being normally distributed. In Appendix A.3 we assess these boundaries by evaluating the extraction success on synthetic truly random datasets.

Quantile-based Bias Initialization (QBI) QBI maliciously initializes a linear layer L before sending it to a client k targeted for extraction. The weight values w of L are initialized from a standard normal distribution. Given a batch size B used on the client side and the number of input features M, QBI determines the bias value b^* using Equation (7), which approximately leads to an activation probability of $1/B$ for each neuron in L. Even though the true distribution of features on the user side is unknown and the features are neither independent nor truly normally-distributed, our approximation is effective in practice.

Algorithm 1 Pattern-Aware Iterative Random Search (PAIRS)

Input: Linear layer L of shape $M \times N$, Number of retries T
K Batches $\tilde{X}_1, \cdots, \tilde{X}_K$ of size $B \leftarrow \lceil N/K \rceil$
Initialize: $L.bias \leftarrow \phi^{-1}(\frac{1}{B}) \cdot \sqrt{M}$ ▷ Fill bias values using quantile function
for all $\tilde{X}_k$ **do**
    Initialize: $F_D \leftarrow \emptyset$ ▷ Frozen data points
    for neuron $n = (k-1)B$ to $kB - 1$ **do**
        for $t = 1$ to T **do** ▷ Random resets for this neuron
            $A \leftarrow L(\tilde{X}_k)_n$ ▷ Get activations via forward pass for neuron n
            $I \leftarrow \{i \mid A[i] > 0\}$ ▷ Get indices of active samples
            $s \leftarrow I[0]$
            if $|I| \neq 1 \vee s \in F_D$ **then** ▷ Check if sample is not isolated, or already covered
                $L.W_i \sim \mathcal{N}$ ▷ Randomly re-initialize weight row i
                continue
            end if
            $F_D \leftarrow F_D \cup \{s\}$ ▷ Mark sample as frozen
        end for
    end for
end for

Pattern-Aware Iterative Random Search (PAIRS) We propose the PAIRS algorithm that further adapts the malicious QBI linear layer to the target domain when auxiliary data is available. By acknowledging that real-world data, such as images, rarely exhibits the assumed i.i.d. properties, PAIRS iteratively searches the weight space to better capture the underlying patterns. The procedure, outlined in Algorithm 1, begins by performing a forward pass with a batch of auxiliary data. It then identifies and re-initializes the weight rows of neurons that are either overactive or underactive, as well as those that exhibit redundant activation patterns. Through this process, PAIRS builds neuron-sample pairs, iteratively searching the weight space until all samples are covered or a fixed number of iterations is reached. Specifically, with an output shape of N and a batch size of B, PAIRS uses N/B batches of B samples for groups of B neurons each. By randomly re-initializing weight values, PAIRS avoids detectability in weight space while increasing the percentage of data that can be perfectly reconstructed, surpassing the performance of plain QBI.

5 Defence: Activation-based Greedy Gradient Pruning

To counter attacks that exploit gradient sparsity in fully connected layers, we propose Activation-based Greedy Gradient Pruning (AGGP). This is a novel approach that detects and mitigates both passive and active data leakage. Unlike previous works that suggest skipping entire training rounds when potential data leakage is detected [14], we adopt a more targeted strategy by selectively pruning gradients of suspect neurons, scaled by their activation pattern. We account for the fact that even benign networks may occasionally leak data points, and that skipping entire updates could withhold valuable training information.

A forward hook is registered at a potentially vulnerable linear layer L. The hook records and caches activation counts, i.e., the number of samples in the batch that lead to a positive activation in a particular neuron. As outlined in Algorithm 2, after the loss and gradients have been calculated, AGGP iterates over all neurons in L. We take a_n to be the number of samples that activate a specific neuron n. Take c to be an arbitrary cut-off sample count. Neurons that did not activate (i.e., with $a_n = 0$) or those that activated for more than c samples (i.e., with $a_n \geq c$) are skipped. For all other neurons with $0 < a_n < c$, the percentage $p_{keep,n}$ of gradient values to retain is calculated as:

$$p_{keep,n} = \frac{(a_n - 1)^2 \cdot (p_u - p_l)}{(c-2)^2} + p_l \tag{14}$$

where p_l and p_u represent the lower and upper bound for the percentage of gradient values that are retained for $a = 1$ and $a = c - 1$ respectively. In our experiments on the ImageNet dataset, we set $c = 16$, $p_l = 0.01$ and $p_u = 0.95$. See Appendix B.2 for a detailed explanation of how we arrived at these hyperparameters. For higher a_n values $p_{keep,n}$ increases as the overlapping samples lead

Algorithm 2 Activation-based Greedy Gradient Pruning (AGGP)

Input: Linear layer L of shape $M \times N$, Batch $\tilde{X}$ of size B, cut-off threshold c, lower and upper pruning bounds p_l and p_u
$\mathbf{A} \leftarrow L(\tilde{X}) > 0$ ▷ Store layer activations during forward pass
$loss = \cdots$
$\mathbf{G} \leftarrow loss.backward()$ ▷ Compute gradients
for all neuron n in L **do**
  $\mathbf{a}_n \leftarrow \sum_{i=1}^{M} \mathbb{I}(A_{n,i} > 0)$ ▷ Number of samples that activate neuron n
  if $a = 0 \vee a > c$ **then** ▷ Neurons with no activation or $>$ cut-off are skipped
    continue
  end if
  $\mathbf{p}_{keep} \leftarrow \frac{(a_n - 1)^2 \cdot (p_u - p_l)}{(c-2)^2} + p_l$ ▷ Determine percentage of gradient values to retain
  $k \leftarrow \lfloor \mathbf{p}_{keep} \cdot N \rfloor$ ▷ Number of elements to fully prune
  $\mathbf{s} \leftarrow \text{argsort}(|\mathbf{G_n}|)$ ▷ Get sorted indices of absolute gradient values of n-th weight row
  $\mathbf{I} \leftarrow \mathbf{s}[:k]$ ▷ Retrieve indices of lowest k and random 75% of top $N - k$ values
  $\mathbf{I} \leftarrow \mathbf{I} \cup \mathbf{s}[k:][\text{randperm}(N-k)[:\lfloor 0.75 \cdot (N-k) \rfloor]]$
  $\mathbf{G_n}[\mathbf{I}] \leftarrow 0$ ▷ Zero out gradients
end for

to more diffuse representations, where individual samples are increasingly obscured. AGGP then continues by sorting the gradients of the corresponding weight row by their absolute magnitude. Among the top $p_{keep,n}$ percent of values, 25% are randomly selected to be retained, while all other values are set to zero, to further obscure potentially connected features. This effectively prevents the perfect reconstruction of individual samples, while 25% of the gradient values corresponding to the $p_{keep,n}$ percent largest magnitudes are maintained, allowing the propagation of valuable training information.

6 Experimental Evaluation

We closely follow the experimental setup of Boenisch et al. [12] for both image and text data, to allow a direct comparison to be made. See Appendix C.3 for further implementation details.

Image Data Extraction We evaluated our method on two benchmark vision datasets: ImageNet [21] at a resolution of 224×224, and CIFAR-10 [22] at a resolution of 32×32. As our method requires normalized data, we used publically available normalization parameters for both datasets. To verify that this kind of domain knowledge is **not** required for our method to work, we ran experiments with unnormalized data, placing a batch normalization layer before our maliciously initialized layer instead. The results listed in 6 show no significant difference in performance. The model we used consisted of convolutional layers maliciously initialized to transfer the input further into the network, followed by a linear layer initialized with either QBI or PAIRS. The rate of perfectly reconstructed images R (see Equation (10)) was evaluated using batch sizes of 20, 50, 100, and 200, and layer sizes of 200, 500, and 1000. Our results, presented in Table 1, surpass those reported by Boenisch et al. [12] using their *trap weights* approach. Our method yields consistent improvements for both the CIFAR-10 and ImageNet datasets across various layer size and batch size combinations. Notably, the most significant gains are observed on ImageNet with smaller batch sizes, where our method achieves up to 50% higher reconstruction rates. Figure 2 in Appendix A.1 presents an example batch of 20 images and the perfectly reconstructed subset of 16 images, obtained from a layer size of 200.

Text Data Extraction The IMDB sentiment analysis dataset [23] was used to evaluate the extraction of text data. Our model operates on 250-token sentences with an embedding dimension of 250, where the embedded tokens are directly fed to a fully connected layer of size 1000. We re-trained the `bert-base-uncased` tokenizer [24] on the IMDB dataset, resulting in a vocabulary size of $10,000$. The extraction was evaluated on batch sizes of 20, 50, 100, and 200. The results, presented in Table 2, are compared to those reported by Boenisch et al. [12]. Our approach performs similarly across batch sizes 20 and 50, but achieves performance gains of 25% and 47% on batch sizes 100 and 200, respectively.

Table 1: Comparing the percentage of perfectly reconstructed images (R, see Equation (13)) with the results reported in [12] using their *trap weights*, across various combinations of neuron counts N and batch sizes B. The values represent the mean across 10 random initializations, with each initialization evaluated on 10 random batches. The error margins indicate the 95% confidence interval.

	ImageNet			**CIFAR-10**		
(N, B)	*[12]*	*QBI (Ours)*	*PAIRS (Ours)*	*[12]*	*QBI (Ours)*	*PAIRS (Ours)*
(200, 20)	35.5	82.5 ± 2.43	$\mathbf{85.5 \pm 1.25}$	69.5	75.7 ± 1.85	$\mathbf{77.1 \pm 1.19}$
(200, 50)	30.4	52.0 ± 1.46	$\mathbf{56.0 \pm 1.13}$	45.2	46.5 ± 1.06	$\mathbf{48.4 \pm 0.43}$
(200, 100)	24.0	29.0 ± 0.92	$\mathbf{34.6 \pm 0.67}$	26.9	28.4 ± 0.60	$\mathbf{31.5 \pm 0.73}$
(200, 200)	11.3	15.1 ± 0.62	$\mathbf{19.5 \pm 0.60}$	9.60	15.8 ± 0.49	$\mathbf{18.6 \pm 0.32}$
(500, 20)	49.0	93.6 ± 1.10	$\mathbf{94.5 \pm 0.84}$	87.0	87.6 ± 1.18	$\mathbf{87.8 \pm 1.36}$
(500, 50)	42.6	73.5 ± 1.50	$\mathbf{76.8 \pm 1.27}$	61.4	63.8 ± 1.09	$\mathbf{67.3 \pm 1.28}$
(500, 100)	35.8	49.0 ± 1.09	$\mathbf{55.8 \pm 0.87}$	42.2	45.1 ± 0.74	$\mathbf{48.1 \pm 0.80}$
(500, 200)	19.9	29.2 ± 0.49	$\mathbf{35.9 \pm 0.35}$	17.7	28.4 ± 0.43	$\mathbf{32.6 \pm 0.60}$
(1000, 20)	59.5	96.6 ± 0.78	$\mathbf{96.7 \pm 0.66}$	91.5	91.3 ± 1.40	$\mathbf{91.8 \pm 1.49}$
(1000, 50)	51.6	84.3 ± 0.59	$\mathbf{86.6 \pm 0.67}$	72.4	74.3 ± 0.93	$\mathbf{77.7 \pm 1.17}$
(1000, 100)	45.7	64.8 ± 0.57	$\mathbf{68.8 \pm 1.00}$	55.6	57.2 ± 0.76	$\mathbf{59.4 \pm 0.52}$
(1000, 200)	28.8	42.7 ± 0.72	$\mathbf{49.4 \pm 3.00}$	25.6	39.2 ± 0.49	$\mathbf{42.6 \pm 0.60}$

Table 2: Comparing the percentage of perfectly reconstructed text samples R from the IMDB dataset [23], with the results reported in [12], across various batch sizes B using a layer size of 1000. The values represent the mean across 10 random initializations, with each initialization evaluated on 10 random batches. The error margins indicate the 95% confidence interval.

B	*Trap weights[12]*	*QBI (Ours)*	*PAIRS (Ours)*
20	100.0	100 ± 0.00	99.9 ± 0.20
50	96.2	$\mathbf{98.9 \pm 0.46}$	98.6 ± 0.43
100	65.4	90.5 ± 0.33	$\mathbf{90.8 \pm 0.82}$
200	25.5	72.8 ± 0.64	$\mathbf{73.3 \pm 0.78}$

Secondary Metrics The advantage of our method can be explained by examining the secondary metrics precision P and activation value A (see Equation (11)). As established in Equation (12), the theoretical optimum for these values lies at $A = 1 - 1/e \approx 63.2\%$ and $P = 1/e \approx 36.8\%$. Although these can only be achieved if the target data is truly normally distributed (see Table 5 in the Appendix), values that lie closer to this optimum will lead to better reconstruction rates. Table 4 in the Appendix compares the A and P values achieved using QBI and PAIRS to those obtained using *trap weights* [12] on image data. Specifically, on the ImageNet dataset, our A values range from 72% to 87%, whereas those reported in Boenisch et al. [12] show a stronger variance across batch sizes, ranging from 9% to 89%. Similarly, our precision values range from 26% to 34% on ImageNet, while those reported in [12] vary from 23% to 94%.

AGGP We find that our proposed defense framework reduces the percentage of perfectly reconstructable samples obtained via gradient sparsity in linear layers to **zero**. This occurs as any neuron that meets the condition for perfect extraction ($a_n = 1$), triggers the pruning of $1 - 0.25 \cdot p_l$ percent of gradient values of its corresponding weight row (see Equation (14)). The effect of our defense is best presented visually – the left-hand side of Figure 1, displays the data that is leaked passively from the first 20 neurons of a benign network after a single training step with a batch of 20 samples from the ImageNet dataset. The right-hand side depicts the impact of AGGP on the same scenario, where gradients of neurons with low activation counts are aggressively pruned, while those with high activation counts remain unaffected. Further visualizations of AGGP's impact on a maliciously initialized model, along with preliminary observations of its effect on training performance, are provided in Appendix A.4.

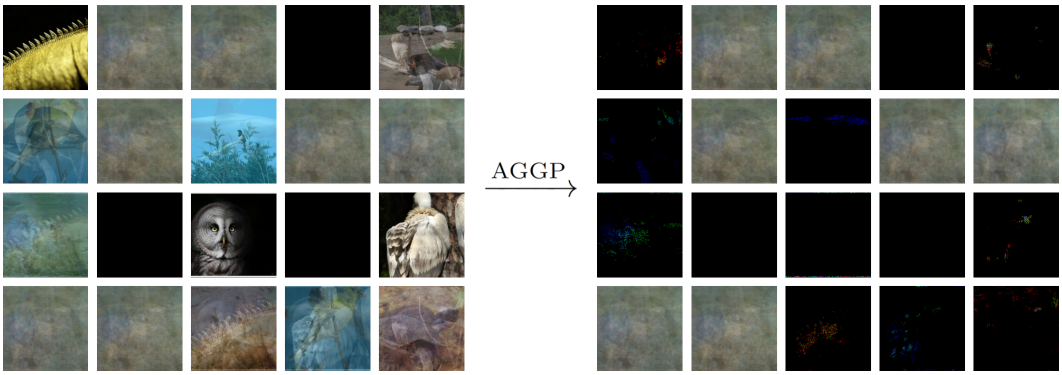

Figure 1: Visualization of the passive data leakage of the first 20 neurons of a linear layer of size 200 (left) and the impact of AGGP (right). Sparsely activated neurons are aggressively pruned, while the gradients of neurons with activation counts exceeding the cut-off threshold remain unaffected.

7 Discussion

Impact on secure aggregation and distributed differential privacy Secure aggregation (SA) [25] and distributed differential privacy (DDP) [26] are proposed modifications to the traditional FL protocol, designed to decrease the amount of trust users have to place in the central entity. SA employs a multiparty computation protocol to perform decentralized gradient aggregation, while in DDP users locally add a small amount of noise to their gradient updates. Boenisch et al. [27] demonstrated that attacks leveraging gradient sparsity can undermine these protocols if the server is able to introduce malicious nodes in the form of so-called sybil devices. Since their work uses the previously described *trap weights* [12], replacing the model initialization with our QBI or PAIRS approach could significantly improve the performance of this attack vector.

Detectability Our method leaves all weight values completely randomly initialized, making it virtually undetectable in weight space. However, it introduces a negative shift in the bias values, which could potentially be detected by the client. Additionally, as our method relies on gradient sparsity, it would be feasible to detect it in gradient space, e.g., by leveraging measures like the disaggregation signal-to-noise ratio [14].

Limitations Boenisch et al. [12] and our attack rely on the existence of a fully connected layer, either at the beginning of the network or positioned such that preceding layers can be maliciously initialized to perfectly transmit the input deeper into the network. Preceding layers that reduce dimensionality, for example, through pooling operations, will diminish fidelity and thereby prevent perfect data extraction. We conduct preliminary tests on the impact of AGGP on training performance that indicate little-to-no adverse effects, however, a comprehensive evaluation across a wider range of architectures, datasets, and pruning functions is needed to achieve generalizable insights.

8 Conclusion

In this work, we introduce a novel bias tuning method designed to enhance attacks targeting private data reconstruction in federated learning systems. Our approach, encompassing the QBI and PAIRS variants, outperforms comparable gradient sparsity methods across various datasets and batch sizes, achieving superior rates of perfect user data reconstruction. By establishing theoretical limits for stochastic gradient sparsity attacks, our work provides a crucial step towards a more comprehensive understanding of the fundamental constraints imposed by the probabilistic nature of these attacks. Additionally, we introduce AGGP as a defense mechanism against gradient sparsity attacks, effectively mitigating data leakage in linear layers. While our attack method poses privacy risks, we believe that by sharing the details of our approach, we can encourage systematic exploration of privacy safeguards and enable practitioners to better assess and mitigate privacy risks in FL deployments, ultimately promoting safer machine learning practices.

References

[1] Tuo Zhang, Chaoyang He, Tianhao Ma, Lei Gao, Mark Ma, and Salman Avestimehr. Federated learning for internet of things. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 413–419, 2021.

[2] Yi Wang, Imane Lahmam Bennani, Xiufeng Liu, Mingyang Sun, and Yao Zhou. Electricity consumer characteristics identification: A federated learning approach. *IEEE Transactions on Smart Grid*, 12(4):3637–3647, 2021.

[3] Treena Basu, Sebastian Engel-Wolf, and Olaf Menzer. The ethics of machine learning in medical sciences: Where do we stand today? *Indian Journal of Dermatology*, 65(5):358–364, 2020.

[4] General data protection regulation (gdpr). `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016.

[5] California consumer privacy act (ccpa). `https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=CIV&division=3.&title=1.81.&part=4.&chapter=&article=`, 2018. California Civil Code, Title 1.81.5, Section 1798.100 et seq.

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

[7] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[8] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 691–706. IEEE, 2019.

[9] Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

[10] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[11] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

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

[13] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Secure aggregation in federated learning is not private: Leaking user data at large scale through model modification. *arXiv preprint arXiv:2303.12233*, 2023.

[14] Kostadin Garov, Dimitar I Dimitrov, Nikola Jovanović, and Martin Vechev. Hiding in plain sight: Disguising data stealing attacks in federated learning. *arXiv preprint arXiv:2306.03013*, 2023.

[15] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2429–2443, 2022.

[16] Shuaishuai Zhang, Jie Huang, Zeping Zhang, and Chunyang Qi. Compromise privacy in large-batch federated learning via malicious model parameters. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 63–80. Springer, 2022.

[17] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022.

[18] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

[19] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.

[20] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[25] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[26] Muah Kim, Onur Günlü, and Rafael F Schaefer. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2650–2654. IEEE, 2021.

[27] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. Reconstructing individual data points in federated learning hardened with differential privacy and secure aggregation. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 241–257. IEEE, 2023.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

A Additional Experimental Results

A.1 Images

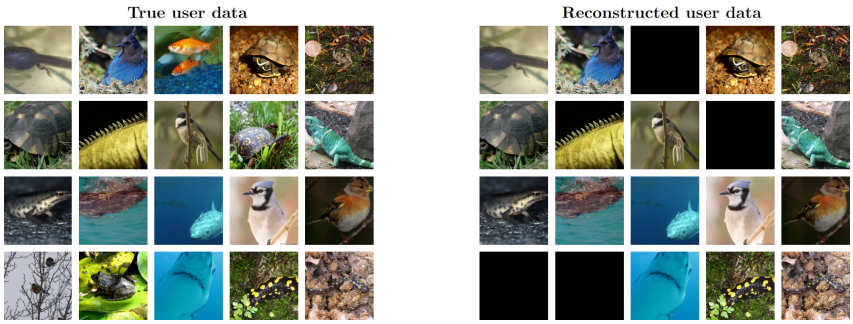

Figure 2: True user data (left), a batch of 20 images from the ImageNet dataset and reconstructed user data (right), using a linear layer of size 200 that was maliciously initialized with our QBI approach. Fully black images denote data points that could not be recovered. Despite the small layer size, in this particular setting, our method achieves perfect reconstruction of around 82.5% of the original data points, on average.

A.2 Activation values (A) and precision (P)

Table 3: Comparing A and P (see Equation (11)) with the results reported by Boenisch et al. [12], across various batch sizes B using a layer size of 1000. Results are averaged over 10 random initializations, each evaluated on 10 random batches.

	A			P		
B	[12]	*QBI*	*PAIRS*	[12]	*QBI*	*PAIRS*
20	51.9	58.7	59.2	61.0	33.3	33.6
50	77.6	57.1	57	37.6	32.6	32.5
100	91.0	55.1	55.5	19.2	31.8	31.5
200	97.8	53.4	53.8	0.07	30.6	30.7

Table 4: Comparing A and P (see Equation (11)) with the results reported by Boenisch et al. [12], across various batch sizes B and layer sizes N. Results are averaged over 10 random initializations, each evaluated on 10 random batches.

	ImageNet						**CIFAR-10**					
	A			P			A			P		
N, B	[12]	*QBI*	*PAIRS*	[12]	*QBI*	*PAIRS*	[12]	*QBI*	*PAIRS*	[12]	*QBI*	*PAIRS*
(200, 20)	0.09	75.6	72.6	94.8	31.4	33.9	45.4	55.7	56.2	67.0	31.2	31.6
(200, 50)	38.1	80.8	76.2	76.3	25.1	29.7	66.2	53.9	57.1	49.4	26.7	29.4
(200, 100)	65.3	84.6	79.0	50.0	21.4	27.8	84.6	54.6	57.6	28.0	24.9	28.7
(200, 200)	88.6	86.5	80.6	23.3	18.8	26.3	95.4	54.6	56.6	12.4	22.7	27.6
(500, 20)	0.09	75.8	72.7	93.9	31.4	33.0	45.2	56.3	56.9	68.9	30.7	32.7
(500, 50)	38.7	81.4	76.2	76.7	26.2	30.2	65.3	54.8	56.7	50.5	26.2	30.3
(500, 100)	64.6	84.6	78.6	50.8	21.6	27.7	84.5	55.7	57.9	29.0	24.4	29.0
(500, 200)	89.2	87.1	80.4	24.0	18.8	26.3	95.0	56.0	57.0	11.9	22.8	27.8
(1000, 20)	10.2	75.5	73.4	94.2	31.4	33.1	44.1	55.5	57.0	70.3	30.2	32.2
(1000, 50)	38.8	80.9	76.4	77.0	26.1	30.1	64.8	55.6	57.5	50.4	27.0	30.1
(1000, 100)	65.5	84.4	79.3	51.4	22.0	28.0	84.4	55.7	56.5	29.7	24.6	28.5
(1000, 200)	89.2	87.3	81.5	23.8	18.8	26.0	95.1	56.0	58.2	12.0	23.0	27.5

A.3 Synthetic Data

Table 5: Comparing the predicted values A_{pred}, P_{pred} and R_{pred}, obtained via Equations (11) and (13), to A_{exp}, P_{exp} and R_{exp} observed experimentally when using a synthetic, fully random dataset. All numbers are averaged over 300 random initalizations, tested with 10 batches of random data each. The experimental setup was idential to that when evaluating the extraction on the CIFAR-10 dataset using QBI, replacing the image data with normal random noise of shape $3 \times 32 \times 32$.

(N, B)	A_{pred}	A_{exp}	P_{pred}	P_{exp}	R_{pred}	R_{exp}
(200, 20)	64.2	64.1	37.7	37.5	97.8	97.7
(200, 50)	63.6	64.2	37.2	37.3	77.5	77.1
(200, 100)	63.4	63.0	37.0	36.7	52.3	52.1
(200, 200)	63.3	63.1	36.9	36.5	30.9	30.7
(500, 20)	64.2	64.2	37.7	38.0	100	100
(500, 50)	63.6	63.4	37.2	37.0	97.6	97.5
(500, 100)	63.4	63.3	37.0	36.8	84.3	83.9
(500, 200)	63.3	63.1	36.9	36.5	60.3	59.8
(1000, 20)	64.2	64.2	37.7	37.6	100	100
(1000, 50)	63.6	63.6	37.2	37.0	99.9	100
(1000, 100)	63.4	63.2	37.0	36.8	97.5	97.1
(1000, 200)	63.3	63.4	36.9	36.8	84.2	83.6

A.4 AGGP

Figure 3: Visualization of the active data leakage of the first 20 neurons of a linear layer of size 200 (left), that was maliciously initialized using QBI, and the impact of AGGP (right). The artificially induced sparsity leads to aggressive gradient pruning across the entire layer.

We evaluated the impact of AGGP on a CNN-based architecture (see Table 7), which could, in theory, be maliciously initialized by changing parameter values without modifying the architecture. Since model performance is not a concern when the central entity is malicious or compromised, the impact is assessed on a benign network. The model's validation accuracy on the CIFAR-10 dataset was evaluated across 10 random initializations, recorded across 25 epochs, using a batch size of 64. The results of these preliminary experiments, presented in Figure 4, show no significant impact on training performance.

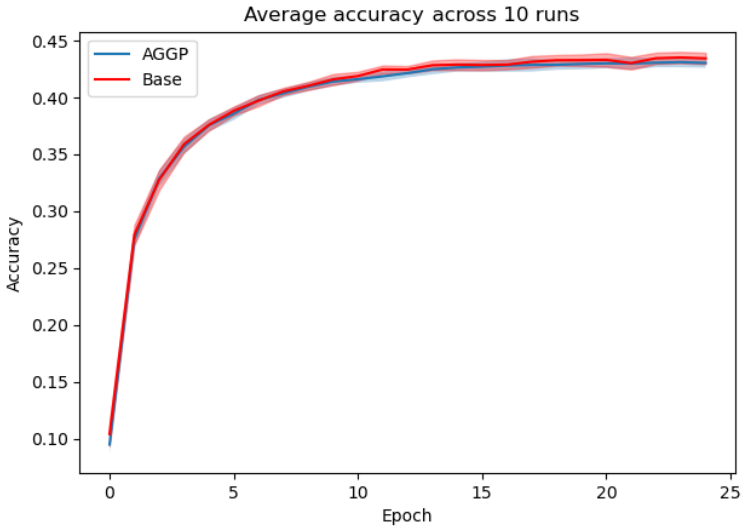

Figure 4: Performance of a benign CNN-based image model (Table 7) on the CIFAR-10 dataset, using a batch size of 64. Comparing the unmodified version (Base) to one protected using AGGP. The experiment used the Adam optimizer [28] with a learning rate of 1e-5 and optimized the cross-entropy loss. Results are averaged across 10 runs using different seeds. The shaded regions correspond to the 95% confidence interval.

A.5 Batchnorm vs. Datanorm

To remove the reliance of our method on the knowledge about normalization parameters for the target domain, we tested our approach on unnormalized data, using a batch norm layer in front of our maliciously initialized layer. Results reported in Table 6 show performance very similar to that achieved using regular data normalization, proving that our method does not require any domain knowledge to achieve high rates of extraction.

Table 6: Comparing extraction recall R on image data achieved using QBI with batch normalization and regular data normalization, across ImageNet and CIFAR-10. The results show no significant impact of the normalization method, underlining that our approach does not rely on domain knowledge about the target data, such as normalization parameters.

	ImageNet		**CIFAR-10**	
N, B	*QBI + Batchnorm*	*QBI + Datanorm*	*QBI + Batchnorm*	*QBI + Datanorm*
(200, 20)	81.3 ± 1.10	82.5 ± 2.43	77.7 ± 1.28	75.7 ± 1.85
(200, 50)	51.9 ± 1.41	52.0 ± 1.46	47.6 ± 0.73	46.5 ± 1.06
(200, 100)	32.5 ± 0.89	29.0 ± 0.92	28.9 ± 0.68	28.4 ± 0.60
(200, 200)	18.7 ± 0.30	15.1 ± 0.62	16.1 ± 0.62	15.8 ± 0.49
(500, 20)	91.4 ± 0.93	93.6 ± 1.10	87.6 ± 1.52	87.6 ± 1.18
(500, 50)	71.3 ± 1.24	73.5 ± 1.50	63.9 ± 1.44	63.8 ± 1.09
(500, 100)	50.0 ± 1.13	49.0 ± 1.09	44.8 ± 0.82	45.1 ± 0.74
(500, 200)	32.6 ± 0.52	29.2 ± 0.49	28.0 ± 0.51	28.4 ± 0.43
(1000, 20)	94.9 ± 1.11	96.6 ± 0.78	91.6 ± 0.93	91.3 ± 1.40
(1000, 50)	81.4 ± 1.10	84.3 ± 0.59	74.5 ± 1.41	74.3 ± 0.93
(1000, 100)	63.4 ± 0.43	64.8 ± 0.57	56.2 ± 0.70	57.2 ± 0.76
(1000, 200)	44.1 ± 0.54	42.7 ± 0.72	38.7 ± 0.59	39.2 ± 0.49

B Background

B.1 Intuition of Equation (13)

Equation (13) estimates the expected percentage of reconstructed samples in the optimal case of normally distributed features. To do this, we calculate the probability of a single sample being successfully reconstructed, which is equal to the expected recall percentage. Let B be the number of samples in our batch, and N be the number of neurons in our linear layer. Additionally, we assume that we have achieved the optimal probability of activation of $1/B$ for every neuron-sample pair. Given a single neuron n_i and a single sample x, the probability that this neuron activates only for x and not for all other samples $x' \neq x$ is

$$\frac{1}{B}\left(\frac{B-1}{B}\right)^{B-1}, \tag{15}$$

which we also refer to as the probability that n_i isolates x. The complement of this event is the probability that the neuron does **not** isolate x:

$$1 - \frac{1}{B}\left(\frac{B-1}{B}\right)^{B-1}. \tag{16}$$

Raising this result to the power of N yields the probability that all neurons do **not** isolate x. Finally, the complement of this event is the probability that at least one neuron isolates x, which in turn results in x being reconstructed. This leads to the final formula:

$$p_{R;B;N} = \mathbb{E}[R] = 1 - \left(1 - \frac{1}{B}\left(\frac{B-1}{B}\right)^{B-1}\right)^{N} \tag{17}$$

B.2 AGGP Hyperparameters

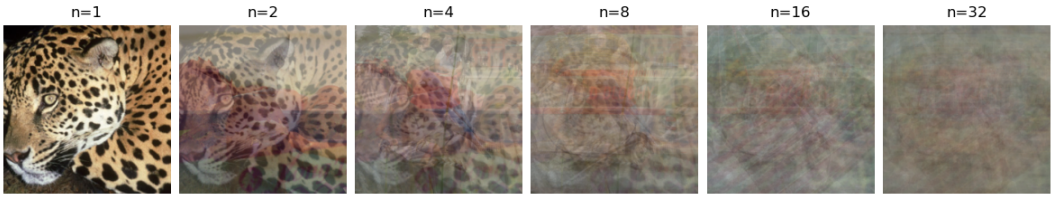

Figure 5: Effect of averaging an increasing number of images n from the ImageNet dataset, on the level of obfuscation.

To determine suitable hyperparameters for AGGP applied to ImageNet data, we investigated the level of obfuscation achieved when multiple images are averaged together in a gradient row. Figure 5 displays an example of this effect, where the average is computed over an increasing number of images n. To quantify this effect more objectively, we conducted a systematic evaluation. We randomly selected one image and averaged it with an increasing number of additional images (from 1 to 25), and then computed three commonly used metrics to quantify the similarity between the original image and the averaged image: the Structural Similarity Index Measure (SSIM, [29]), L1 Distance, and Peak Signal-to-Noise Ratio (PSNR). By repeating this process 100 times and averaging the metrics across all experiments, we obtained the results shown in Figure 6. The results clearly show that PSNR and SSIM decrease sharply in the range $[0, 15]$, while the L1 distance increases. After that, the values appear to converge slowly with little to no movement, which is why we decided to select 16 as our cut-off value c. As evident from Figure 5 and Figure 6, low activation counts provide little obscurity, while higher activation counts require minimal pruning to obscure data, which is why we set the bounds for p_{keep} to $p_l = 0.01$ and $p_u = 0.95$.

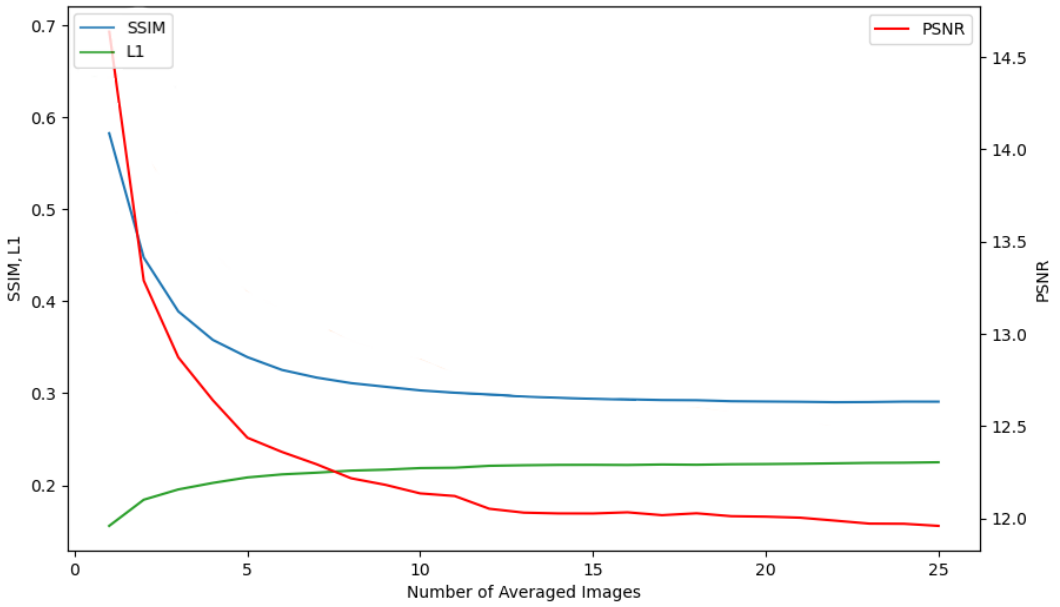

Figure 6:

Figure 7: Average similarity metrics (SSIM, L1 Distance, PSNR) between an original image from the ImageNet dataset and its average with 1 to 25 additional images, repeated 100 times. PSNR and SSIM drop significantly up to N=15, while L1 Distance increases.

C Experiment details

All results that we report are averaged across 10 runs, each evaluated on 10 batches of unseen test data. For each individual run, the weights are randomly initialized, and the train / test split is shuffled randomly. In the case of malicious model initialization and data extraction, train images refer to those used by the PAIRS algorithm to tune the model's parameters, while test images refer to those used to evaluate the reconstruction percentage. For each (N,B) setting, the standard error across these 10 runs is calculated, and multiplied by 1.96 to determine the 95% CI. For every batch, a single forward pass is performed, and the metrics A, P and R (see Equations (11) and (13)) are obtained. Since during evaluation, we have access to the internals of the model, we don't have to examine the gradients to determine the data leakage. Rather we directly determine which samples will be leaked by observing the activation patterns in the maliciously initialized layer during the forward pass. Furthermore this direct access allows us to feed the input directly to our maliciously initialized layer, circumventing compute heavy convolutional layers that were initialized as identity functions.

C.1 Datasets

We used three datasets to evaluate our method: The ImageNet-1k [21] validation set (6GB, 50k images) [1], the CIFAR-10 [22] dataset [2] and the IMDB [23] binary sentiment classification dataset [3]. All these datasets can be used for non-commercial research purposes.

C.2 Compute Resources

All experiments were run on an RTX 2060 Super with 8GB VRAM. Data-extraction runs in near real-time, scaling linearly with batch size and data dimensionality, as it is simply done in one step, by dividing the weight gradients by the bias gradients. Time to maliciously initialize the model varied

[1] https://image-net.org/challenges/LSVRC/2012

[2] https://www.cs.toronto.edu/ kriz/cifar.html

[3] https://ai.stanford.edu/ amaas/data/sentiment

across methods: QBI required less than 1 second in all settings, while PAIRS initialization times ranged from 10 seconds for CIFAR-10 (N=200, B=20) to 12 minutes for ImageNet (N=1000, B=200). For even larger layer sizes N, initialization time for PAIRS will scale linearly. Obtaining all results, for both QBI and PAIRS, across all possible combinations of N and B and all three datasets, averaged across 10 runs with random seeds for model intialization and train / test split, required approximately 8 hours of compute. Evaluation of AGGP's impact on training performance across 20 runs (10 Base, 10 protected with AGGP) required about 2 hours of training time. Experiments evaluating QBI on synthetic data (see Table 5) took about 1 hour, averaging over 300 random runs per (N, B) setting.

C.3 Image models

Table 7 outlines the implementation of the model used for image data extraction. As Boenisch et al. [12] explain in their Appendix B, convolutional layers can be modified to pass the input to the next layer, effectively acting as an identity function. This can be achieved through various methods. Algorithm 3 provides a minimal working example using a 2D convolutional layer, suitable for RGB images. To preserve the image shape, we employ a kernel size of 3, padding of 1, and stride of 1. Since we have three channels to transmit, we initialize three filters, each acting as the identity function for its respective channel. This is achieved by setting the weight values of the i-th filter to zero and then setting the center value of its weight matrix for the i-th channel to one. Additionally, randomly initialized filters could be added to obscure the modifications made to the model.

Algorithm 3 Conv2D Identity Initialization Example

$num\_channels \leftarrow 3$
$conv2d \leftarrow \textsc{Conv2D}(in = 3, out = 3, k = 3, s = 1, p = 1)$
for $i \leftarrow 0$ to $num\_channels$ **do**
    $conv2d.weight.data[i, :, :, :] \leftarrow 0$
    $conv2d.weight.data[i, i, 1, 1] \leftarrow 1$
end for

Table 7: Architecture of models used in the experiments on image data. f: number of filters, k: kernel size, s: stride, p: padding act: activation function, n: number of neurons. The size of the second to last layer was varied across experiments.

CNN Architecture
Conv(f=128, k=(3, 3), s=1, p=1)
Conv(f=256, k=(3, 3), s=1, p=1)
Conv(f=3, k=(3, 3), s=1, p=1)
Flatten
<Optional> BatchNorm
Dense(n=1000, act=ReLU)
Dense(n=#classes, act=None)

Table 8: Architecture of models used in the experiments on the IMDB dataset. feat: vocabulary size, dim: embedding size, act: activation function, n: number of neurons.

IMDB-Model Architecture
Embedding(feat=10_000, dim=250)
<Optional> BatchNorm
Dense(n=1000, act=ReLU)
Dense(n=1, act=None)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 6 outlines our claimed performance gains on the stated datasets in detail, while our provided descriptions of the proposed methods (Section 4) in combination with the code provided by us allows their reproducibility. Section 4 further contains a detailed derivation of the proposed fundamental constraints, while Appendix A.3 lists our assessment using synthetic datasets. Finally section 5 introduces our defensive framework, which is further evaluated in section 6 and Appendix 2.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 outlines the limitations of our approach, e.g. the requirement for particular model architectures, and the existence of normalization parameters. Further, Appendix C.2 states the linear scaling behavior with increasing data size, batch size and layer size.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Our theoretical results derived in Section 4 list the full set of assumptions and a complete and correct proof. Supplementary material to further explain the intuition is provided in Appendix B.1.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: While we do provide the code to reproduce our results, Section 4 explains how to apply our algorithms in practice, while Section 6 and Appendix C provide further implementation details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide open access to dedicated scripts to reproduce all reported results, along with demo notebooks that enable a quick demonstration and visualization of our methods (3 second runtime to demonstrate a full experiment including QBI and extraction on the CIFAR-10 dataset).

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Implementation details are stated in Section 4, 6, Appendix C, and available via our shared code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For R, which is the main metric of success (percentage of perfectly reconstructed samples), the 95% CI is provided with every result, see Table 1 and Table 2. For the secondary metrics A and P, the standard error is omitted to improve readability (see Table 3, 4 and 5), as neither A nor P directly measure the success of an attack. Section C.3 explains the method for calculating the error margins.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Appendix C.2 states the hardware that was used, including memory and time of execution for all experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have reviewed the code of ethics and verified that the datasets we used conform to the guidelines. Furthermore in Section 7 and 8 we acknowledge the potential impact on security and privacy in FL attack details.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Section 7 and 8 highlight the potential impact on safety and privacy of FL systems (even under application of SA and DDP). Section 8 emphasizes our aim to facilitate the development of more robust FL systems by sharing our attack details, while our proposal of the defensive framework AGGP (Section 5) is a first step in this direction.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not provide datasets or pretrained models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Creators, URLs and terms of use are named in Appendix C.1.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We are releasing our code, together with detailed instructions on how to install, set up and run our experiments. The code contains documentation, further detailing the operational mechanisms.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.