ROBBING THE FED: DIRECTLY OBTAINING PRIVATE DATA IN FEDERATED LEARNING WITH MODIFIED MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning has quickly gained popularity with its promises of increased user privacy and efficiency. Previous works have shown that federated gradient updates contain information that can be used to approximately recover user data in some situations. These previous attacks on user privacy have been limited in scope and do not scale to gradient updates aggregated over even a handful of data points, leaving some to conclude that data privacy is still intact for realistic training regimes. In this work, we introduce a new threat model based on minimal but malicious modifications of the shared model architecture which enable the server to directly obtain a verbatim copy of user data from gradient updates without solving difficult inverse problems. Even user data aggregated over large batches – where previous methods fail to extract meaningful content – can be reconstructed by these minimally modified models.

1 INTRODUCTION

Federated learning (Konečný et al., 2015), also known as collaborative learning (Shokri & Shmatikov, 2015), is a mechanism for training machine learning models in a distributed fashion on multiple user devices. In the simplest setting, a central *server* sends out model states to a group of *users*, who compute an update to the model based on their local data. These updates are then returned to the server, aggregated, and used to train the model. Over multiple rounds, this protocol can train a machine learning model, distributed over all users, without exchanging local data – only model updates are exchanged. Two central goals of federated learning are to improve training efficiency by decreasing communication overhead and to side-step issues of user-level privacy and data access rights that have become a focus of public attention in recent years (Veale et al., 2018).

Accordingly, many organizations, ranging from large tech companies (McMahan & Ramage, 2017) to medical institutions with especially strict privacy laws, such as hospitals (Jochems et al., 2016), have utilized federated learning to train machine learning models. However, in practice, data privacy is not guaranteed in general, but is dependent on a large number of interdependent settings and design choices specific to each federated learning system. In this work, we focus on the user perspective of privacy, and we study federated learning systems in which the central server is not able to directly view user data.

The key privacy concern for users is whether model updates reveal too much about the data on which they were calculated. Although Kairouz et al. (2021) discuss that "model updates are more focused on the learning task at hand than is the raw data (i.e. they contain strictly no additional information about the user, and typically significantly less, compared to the raw data)", scenarios can be constructed in which the model updates themselves can be inverted to recover their input user information (Wang et al., 2018; Melis et al., 2018). Simple knowledge of the shared model state and model update can be sufficient for such an attack (Zhu et al., 2019; Geiping et al., 2020). These inversion attacks are particularly fruitful if a user's model update is based on a single data point or only a small batch. Accordingly, a strong defense against these attacks is aggregation. The user only reports model updates aggregated over a significant number of local data points, and data from multiple users can be combined with secure aggregation protocols (Bonawitz et al., 2017) before being passed to the server. This ability to aggregate user updates while maintaining their utility

thought to be the main source of security in federated learning. Averaging raw local data in similar amounts would make it unusable for training, but model updates can be effectively aggregated.

Previous inversion attacks typically focus on a threat model in which the server (server here is a stand-in for any party with root access to the server or its incoming and outgoing communication) is interested in uncovering user information by examining updates, but without modifying the federated learning protocol, a behavior also referred to as *honest-but-curious* or *semi-honest* (Goldreich, 2009). In our case, where the party intending to recover user data is the server, this "honest" scenario appears contrived, as the server can modify its behavior to obtain private information. In this work, we are thus interested in explicitly *malicious servers* that may modify the model architecture and model parameters sent to the user. We focus on scenarios in which an agent obtains data without making suspicious changes to the client code or learning behavior. One scenario which enables this threat model involves recently introduced APIs that allow organizations to train their own models using established federated learning protocols (Cason, 2020). In this environment, a malicious API participant can change their model's architecture and parameters but cannot force unsuspecting edge devices to send user data directly.

We introduce minimal changes to model architectures that enable servers to breach user privacy, even in the face of large aggregations that have been previously deemed secure. These changes induce a structured pattern in the model update, where parts of the update contain information only about a fixed subset of data points. The constituent data points can then be recovered exactly, while evading existing aggregation defenses.

2 Limitations of Existing Attack Strategies

A range of possible attacks against privacy in federated learning have been proposed in recent literature. In the simplest case of *analytic attacks*, Phong et al. (2017b) were among the first to discuss that the input to a learnable affine function can be directly computed from the gradient of its weights and bias, and additional analysis of this case can be found in Qian & Hansen (2020); Fan et al. (2020), and in Section 3.2. However, analytic recovery of this kind only succeeds for a single data point. For multiple data points, only the average of their inputs can be recovered, leading the attack to fail in most realistic scenarios.

Recursive attacks as proposed in Zhu & Blaschko (2021) can extend analytic attacks to models with more than only linear layers - a construction also mentioned in Fan et al. (2020). However, these attacks still recover only the average of inputs in the best case. Improvements in Pan et al. (2020) transform linear layers with ReLU activations into systems of linear equations that allow for a degree of recovery for batched inputs to these linear layers, although preceding convolutional layers still have to be deconvolved by recursion or numerical inversion techniques.

Surprisingly, *optimization-based attacks* turn out to be highly effective in inverting model updates. Wang et al. (2018) propose the direct recovery of input information in a setting where the users' model update is the model parameter gradient averaged over local data. In a supervised learning setting, we define this update by g and the loss function over this model by $\mathcal{L}$ with model parameters θ and data points $(x, y) \in [0, 1]^n \times \mathbb{R}^m$. The server can then attempt recovery by solving the gradient matching problem of

$$\min_{x \in [0,1]^n} ||\nabla_\theta \mathcal{L}(x, y, \theta) - g||^2 \tag{1}$$

and solve this optimization objective using first-order methods or any nonlinear equation solver. Subsequent work in Zhu et al. (2019); Zhao et al. (2020) and Wainakh et al. (2021) proposes solutions that also handle recovery of targets y and variants of this objective are solved for example in Geiping et al. (2020) with cosine similarity and improved optimization and in Jeon et al. (2021) with additional generative image priors. Reconstruction of input images can be further boosted by additional regularizers as in Yin et al. (2021) and Qian et al. (2021).

Most attacks in the literature focus on the described *fedSGD* setting (Konečný et al., 2015) in which the users return gradient information to the server, but numerical attacks can also be performed against local updates with multiple local steps Geiping et al. (2020), for example against *fedAVG* (McMahan et al., 2017). In this work, we will discuss both update schemes, noting that gradient aggregation in "time", with multiple local update steps, is not fundamentally more secure than aggregation over multiple data points. Previous attacks also focus significantly on learning scenarios

where the user data is comprised of images. This is an advantage to the attacker, given that image data is highly structured, and a multitude of image priors are known and can be employed to improve reconstruction. In contrast, data types with weaker structure, such as tabular data, do not lend themselves to regularization based on strong priors, and we will show that our approach, on the other hand, does not rely on such tricks, and is therefore more data-agnostic.

The central limitation of these attack mechanisms is the degradation of attack success when user data is aggregated over even moderately large batches of data (either by the user themselves or by secure aggregation). State-of-the-art attacks such as Yin et al. (2021) recover only 28% of the user data (given a charitable measure of recovery) on a batch size of 48 for a ResNet-50 (He et al., 2015) model on ImageNet (ILSVRC2012 (Russakovsky et al., 2015)) with unlikely label collisions. The rate of images that can be successfully recovered drops drastically with increased batch sizes. Even without label collisions, large networks such as a ResNet-32-10 (Zagoruyko & Komodakis, 2016) leak only a few samples for a batch size of 128 in Geiping et al. (2020). These attacks further reconstruct only approximations to the actual user data which can fail to recover parts of the user data or replace it with likely but unrelated information in the case of strong image priors.

Further, although all of the previous works nominally operate under an *honest-but-curious* server model, they do often contain model adaptations on which reconstruction works especially well, such as large vision models with large gradient vectors, models with many features (Wang et al., 2018; Zhu & Blaschko, 2021), special activation functions (Zhu et al., 2019; Zhu & Blaschko, 2021), wide models (Geiping et al., 2020), or models trained with representation learning (Yin et al., 2021; Chen et al., 2020). These may be seen as *malicious* models with architectural choices that breach user privacy. In the same vein, we ask, what is the worst-case (but small) modification that can be applied to a neural network to break privacy?

3 Model Modifications

In this section, we detail an example of a small model modification that has a major effect on user privacy, even allowing for the direct recovery of verbatim user data from model updates.

3.1 Threat Model

We define two parties: the server $\mathcal{S}$ and the users $\mathcal{U}$. The server could be a tech company, a third party app using a federated learning framework on a mobile platform, or an organization like a hospital. The server S defines a model architecture and distributes parameters θ for this architecture to the users, who compute local updates and return them to the server. The server cannot deviate from standard federated learning protocol in ways beyond changes to model architecture (within limits imposed by common ML frameworks) and model parameters. We measure the strength of a malicious modification of the architecture using the number of additional parameters inserted into the model. While it is clear that models with more parameters can leak more information, we will see that clever attacks can have a disproportionate effect on attack success, compared to more benign increases in parameter count, such as when model width is increased.

3.2 A Simple Example

To motivate the introduction of malicious modifications, we start with the simple case of a fully connected layer, followed by a ReLU activation – a simple construction that is highly insecure. Note the following calculation holds in general when the layer is positioned at some depth l within a network. A forward pass on this layer is written as $a_l = f(W_l a_{l-1} + b_l)$, where f is the ReLU activation, W_l is a weight matrix, b_l is a bias, and a_{l-1} is the layer's input. As seen in Phong et al. (2017a); Qian & Hansen (2020); Fan et al. (2020), when the parameters of the network are updated according to some objective $\mathcal{L}$, the ith row of the update to W_l (as long as $W_l^i a_{l-1} + b_l^i > 0$) is:

$$\nabla_{W_l^i} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial a_l^i} \cdot \nabla_{W_l^i} a_l^i = a_{l-1},$$

where we use the shorthand $\mathcal{L} = \mathcal{L}(a_{l-1}; W_l, b_l)$. Similarly,

$$\frac{\partial \mathcal{L}}{\partial b_l^i} = \frac{\partial \mathcal{L}}{\partial a_l^i} \frac{\partial a_l^i}{\partial b_l^i} = \frac{\partial \mathcal{L}}{\partial a_l^i}.$$

Figure 1: **Left:** Analytic reconstruction for a linear model of 64 ImageNet images with separate classes (PSNR: 36.45 versus true user data). **Right:** Same recovery algorithm but for 64 images from the same class (white shark), (PSNR: 13.84 versus true user data.)

So as long as there exists some index i with $\frac{\partial \mathcal{L}}{\partial b_l^i} \neq 0$, a single input is recovered perfectly as:

$$a_{l-1} = \nabla_{W_l^i} \mathcal{L} \oslash \frac{\partial \mathcal{L}}{\partial b_l^i} \tag{2}$$

where $\oslash$ denotes entry-wise division.

3.3 Imprinting User Information into Model Updates

The linear reconstruction described above suffers from significant problems.

- **Averaging**: Perfect reconstruction assumes a *single* user datum. When the update is averaged of multiple training instances from the same class, the server reconstructs the *average* of the instances. In Fig. 1, we see that in the worst-case scenario where all data points fall into the same class, each piece of user data contributes to the gradient equally, resulting in a mashup reconstruction that leaks little private information.
- **Integration**: The architecture is only guaranteed to work as a standalone model, and not within a larger network. If the naive linear model was placed before another network, like a ResNet-18, then gradient entries for the linear layer contain inputs averaged over all labels, as the combined network ostensibly depends on each output of the linear layer.
- **Scalability**: on an industrial scale dataset like ImageNet, the naive fully-connected model would require $> 150M$ parameters to retrieve an image from each label.

Nonetheless, the perfect reconstruction afforded by the linear model remains an attractive feature. To this end, we introduce the *imprint module* class of modifications which overcome the previously described issues, while maintaining the superior reconstruction abilities of the naive linear model. Our imprint module is a combination of commonly used architectural features with maliciously modified parameters that can create unique and separable gradient entries for large volumes of data.

The imprint module can be constructed with a single linear layer (with bias), together with a ReLU activation. Formally, let $\{x_i\}_{i=1}^n = X \in \mathbb{R}^{n \times m}$ be a batch of size n of user data, then a malicious server can define an imprint module whose forward pass (on a single datapoint, x) looks like

$$M(x) = f(W_* x + b_*),$$

where f is a standard ReLU nonlinearity. The crux of the imprint module lies in the construction of $W_* \in \mathbb{R}^{k \times m}$ and $b_* \in \mathbb{R}^k$. We denote the ith row (or channel) of W_* and the ith entry of b_* as W_*^i

and b_*^i, respectively. We then construct $W_*^{(i)}$ so that

$$\langle W_*^i, x\rangle = h(x),$$

where h is some linear function of the data. For example, if the user data are images, h could be average brightness, in which case W_*^i is simply the row vector with entries identically equal to $\frac{1}{m}$.

In order to define the entries of the bias vector, we assume that the server knows the cumulative density function (CDF), assumed to be continuous for the quantity measured by h. For brightness this is straightforward as image data is often normalized. For other quantities, e.g. red-channel intensity, it would not be difficult for a malicious server to estimate this distribution. We stress that the choice of h here is not important to our method. For the purpose of explanation, we will assume that the quantity measured by h is distributed normally, with $\mu = 0$, $\sigma = 1$. Then, the biases of the imprint module are determined by

$$b_*^i = -\Phi^{-1}(\frac{i}{k}) = -c_i,$$

where Φ^{-1} is the inverse of the standard Gaussian CDF. In plain language, we first measure some quantity, like brightness, with the matrix W_*. We duplicate this measurement along the k channels (rows) of W_*. In the meantime, we create k "bins" for the data corresponding to intervals of equal mass according to the CDF of h. Then, the measurement for a given datapoint will land somewhere in the distribution of h.

For example, consider the case when the brightness of some image x_t lands between two values: $c_l \leq h(x_t) \leq c_{l+1}$, and no other image in the same batch has brightness in this range. In this situation, we say that x_t alone activates bin l. Then, if the image x_t is passed through the imprint module, we have

$$\left(\nabla_{W_*^l}\mathcal{L} - \nabla_{W_*^{l+1}}\mathcal{L}\right) \oslash \left(\frac{\partial \mathcal{L}}{\partial b_*^l} - \frac{\partial \mathcal{L}}{\partial b_*^{l+1}}\right) = x_t + \sum_{s=1}^{p} x_{i_s} - \sum_{s=1}^{p} x_{i_s} = x_t, \tag{3}$$

where images $\{x_{i_s}\}$ are images from the batch with brightness $> c_l$. That is, the difference in successive rows $l, l+1$ of the gradient entry for W_* correspond to all elements with brightness $c_l \leq h(x) \leq c_{l+1}$ (in this case, assumed to be just x_t). This is because all of $x_t \cup \{x_{i_s}\}$ activate the non-linearity for layer l, since all these images have brightness $\geq c_l$, however, only images $\{x_s\}$ have brightness $\geq c_{l+1}$, so only these images activate the non-linearity for layer $l+1$. An interesting biproduct of this setup is that it would be difficult even for hand inspection of the parameters to reveal the inclusion of this module as the server could easily permute the bins, and add random rows to W_* which do not correspond to actual bins and only contribute to model performance.

Recall the parameter k defined in the construction of the imprint module. This corresponds to the number of bins the malicious server can create to reconstruct user data. If a *batch* of data is passed through the imprint module, depending on the batch size n used to calculate the update sent to the server, and number of bins, k, the server can expect several bins to activate for *only* one datapoint. And the corresponding entries of the gradient vector can be appropriately combined, and inverted easily. The following result quantifies the user vulnerability in terms of the number of imprint bins, k, and the amount of data, n, averaged in a given update.

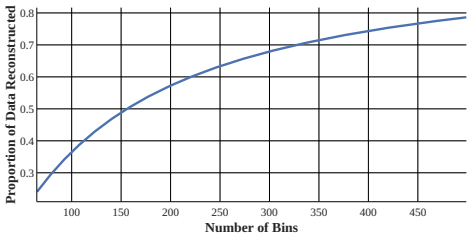
Proposition 1. *If the server knows the CDF (assumed to be continuous) of some quantity associated with user data that can be measured with a linear function $h : \mathbb{R}^m \to \mathbb{R}$, then for a batch of size n and a number of imprint bins $k > n > 2$, by using an appropriate combination of linear layer and ReLU activation, the server can expect to exactly recover*

$$\frac{1}{\binom{k+n-1}{k-1}}\left[\sum_{i=1}^{n-2} i \cdot \binom{k}{i} \cdot \left(\sum_{j=1}^{\lfloor\frac{n-i}{2}\rfloor} \binom{k-i}{j}\binom{n-i-j-1}{j-1}\right)\right] + r(n,k)$$

samples of user data (where the data is in $\mathbb{R}^m$) perfectly. Note: $r(n,k)$ is a correction term (see proof for full expansion).

Proof. See Appendix A.1. □

This can be thought of as a *lower* bound on privacy breaches since, often, identifiable information can be extracted from a mixture of two images. Note that increasing the expected number of perfectly reconstructed images requires increasing the number of imprint bins, which in turn requires increasing the number of channels of the matrix W_*. Thus, to visualize the result above in terms of the server-side hyperparameter, k, we plot the expected proportion of data recovered as a function of number of bins in Fig. 2(a). Note that this inversion is analytic, and significantly more efficient and realistic than optimization based methods which often require tens of thousands of update steps.



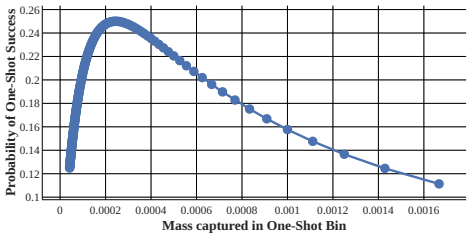


Figure 2: **Left (a)**: Expected proportion of a batch of 64 images *perfectly* recovered as a function of number of bins added via an imprint block in front of a ResNet-18 on ImageNet. With only 156 bins, an attacker can expect to recover over 50% of a batch of user images perfectly. **Right (b)**: Probability of a successful "one-shot" attack on a batch of 4096 images as a function of mass captured in the one-shot bin. An attacker can optimize their bin size given an expected batch size.

The imprint module as described can be inserted in any position in any neural network which receives a non-zero gradient signal. To recover the input feature dimension of the module, a second linear layer can be appended, or – if no additional parameters are of interest – the sum of the outputs of the imprint module can be added to the next layer in the network. The binning behavior of the imprint module is independent of the structure of succeeding layers as long as any gradient signal is propagated. This flexibility allows for wide trade-offs between inconspicuousness and effectiveness of the imprint module. The module can be placed in later stages of model whereas an early linear layer might be suspicious to observers (now that they have seen this trick), but depending on the data modality, early linear layers can be a feature of an architecture anyway, in which case there is even no model modification necessary, only parameter changes. The parameter alterations necessary to trigger this vulnerability can furthermore be hidden from inspection until use. The layer can lay "dormant", functioning and training as a normal linear layer initialized with random activations, as long as the server desires. At any point, a party with access to the server can send out parameter updates containing W^*, b^* and trigger the attack.

4 Experiments

In the following section, we provide empirical examples of imprint modules. In all experiments we evaluate ImageNet examples to relate to previous work (Yin et al., 2021), but stress that the approach is entirely data agnostic. Given an aggregated gradient update, we always reconstruct as discussed in Section 3. When, in the case of imprecision due to noise, more candidate data points are extracted than the expected batch size, only the candidates with highest gradient mean per row in W^* are selected and the rest discarded. For analysis, we then use ground-truth information to order all data points in their original order (as much as possible) and measure PSNR scores as well as Image Identifiability Precision (IIP) scores as described in Yin et al. (2021). For IIP we search for nearest-neighbors in pixel space to evaluate a model-independent distance - a more strict metric compared to IIP as used in Yin et al. (2021). All computations run in single floating point precision.

4.1 Full batch recovery

We begin with a straightforward and realistic case as a selling point for our method - user gradient updates aggregated over a batch of 64 ImageNet images. We modify a ResNet-18 to include an imprint module with 128 bins in front. This relatively vanilla setup presents is a major stumbling block for optimization based techniques. For example, for a much smaller batch size of 8, the prior art in optimization based batched reconstruction achieves only 12.93 average PSNR (Yin et al.,

2021). We do of course operate in different threat models (although Yin et al. (2021) also uses non-obvious parameter modifications). However, our imprint method is successfully able to recover almost perfect reconstructions of a majority of user data (see Fig. 3), and achieves an average PSNR of 70.94. We further stress that a batch size of 64 is by no means a limitation of the method. If bins, and hence additional rows in W^* are added proportionally to the expected batch size, then recovery of significant proportions (see Fig. 2) of batches of arbitrary size – albeit with the incurred cost in additional parameters for each row – is possible.

Figure 3: **Left:** Ground truth batch of 64 user images. **Right:** Analytic reconstruction for an imprint model with 128 bins in front of a ResNet-18. Gray reconstructions occur when there is a collision in a bin and no input is recovered from this row.)

4.2 Privacy breaches in industrial-sized batches – One-shot Attacks

A breach in privacy can occur if even a *single* piece of user data is compromised and massively increasing the number of parameters might not be desirable for the server and could raise suspicion under inspection. However, there is a threatening modification of the imprint module in this case. If a server has access to enough users, then it becomes feasible for the server to start *fishing* for private data among all updates, and attempt to recover a single data point from each incoming batch of data. Attacks of this nature require only as many additional parameters as the size of a single piece of targeted user data, as only a single well-chosen bin is needed. For this, k bins are constructed initially, and then all but 1 bin are "fused" to create 3 final bins: a one-shot bin containing mass $1/k$, and the other two (one on each side of the one-shot bin) containing the remaining mass. For perspective, for a ResNet-18 on ImageNet, this would require only an additional 1% of parameters. And, based on Proposition 1, it is always possible to select an optimal bin size, so that a data point is leaked on average once every four batches of incoming data, *no matter how large*. We demonstrate this statistical property by recovering a single image from an aggregated batch of $2^{14} = 16,384$ ImageNet images (see Fig. 4). Even though the gradient updates are averaged over a vast number of data points, there can be no perfect privacy, and one data point is leaked in its entirety by only a minor model modification.

4.3 Variants

Flexible placement The imprint module introduced in the previous section does not depend on its placement within a given model. No matter the position in a network, the incoming input features will be leaked to the server. Furthermore, the server can also change the parameters of preceding layers to represent (near)-identity mappings, allowing for the recovery of raw input data from inconspicuous positions deep in a network. For a convolutional network, we show examples of this strategy in Fig. 11 for a ResNet-18. Here, the model parameters are manipulated to contain identity maps up to the location of the imprint module, while the downsampling operations remain. Even these later layers leak enough information that their inputs can be upsampled to the breach the privacy of the full resolution inputs.

Figure 4: **Left (a)**: A true user image from class "minibus". **Right (b)**: The reconstructed image from class "minibus" captured via our one-shot attack from averages aggregated over **16,384 datapoints**. PSNR 161.36, i.e. a verbatim copy around machine precision. This user could potentially be identified via their recovered license plate, which was blanked out for preserve privacy.

Multiple local updates: In several federated learning protocols, such as fedAVG, users take several local update steps on data before sending model updates to the server Konečný et al. (2015). The imprint module is threatening when a large amount of user data is used for a *single* update step. However, when multiple steps are taken, inverting gradients from pairwise differences (as in Eq. (3)) becomes more difficult, as entries in W^* shift with local updates. However, an imprint variant that produces sparse gradients per data point is a threat to such federated averaging. Defining a forward pass in this new variant as $M'(x)$ as $M'(x) = g(W_* x + b_*)$ where the non-linearity g is a thresholding function:

$$g(t) = \begin{cases} 0 & t \leq 0 \\ t & 0 \leq t \leq 1 \\ 1 & 1 \leq t \end{cases}$$

Note this non-linearity can be simply constructed with a combination of two ReLUs, or with an implementation of a Hardtanh. We now define W_*^i as: $\langle W_*^i, x \rangle = \frac{h(x)}{\delta_i}$ where h is the a linear function of the data as described before, and the biases are defined as:

$$b_*^i = -\frac{c_i}{\delta_i} \quad \text{where} \quad \delta_i = \Phi^{-1}(\frac{i+1}{k}) - \Phi^{-1}(\frac{i}{k}).$$

This setup creates sparse bins where inversion is directly possible (without taking pairwise differences) in gradient entries, at the cost of an additional activation layer. Analyzing a local update step with $W_*^{i,j}$ as the ith row of W_* at update step j, reveals that

$$W_*^{i,j} = W_*^{i,j-1} - \alpha \frac{\partial \mathcal{L}}{\partial a^{i,j}} x_j$$

where x_j denotes the data from the previous batch that activated bin i, and $a^{i,j}$ denotes the ith activation at step j, and α is the local learning rate. Note that if either a linear layer, or a convolutional layer follows this imprint module, then $\frac{\partial \mathcal{L}}{\partial a^{i,j}}$ does not depend on the scale of $W_*^{i,j}$. Therefore, a simple way to increase the effectiveness of the new imprint module, M' in the fedAVG case is to scale the linear function associated to the rows of W_* - i.e. $h'(x) = c_0 \cdot h(x)$. This "flattens" the distribution of values, and increases the relative size of $b_*^{i,j}$ compared to the gradient update $\frac{\partial \mathcal{L}}{\partial a^{i,j}}$, which prevents the bins from shifting too significantly during local updates.

With this modification, reconstruction quality is similar to the gradient update setting. For example, splitting the batch of 64 ImageNet images up into 8 local updates with learning rate $\tau = 1e-4$ yields an IIP score of 59.38%, mostly due to image duplications where images hit multiple shifted bins. We visualize results in Appendix Fig. 16.

Other data modalities: Yet another advantage to our imprint module over existing optimization based gradient inversion techniques is the flexibility in data domain. Other techniques have demonstrated some success in the image domain by leveraging strong regularizers including total variation

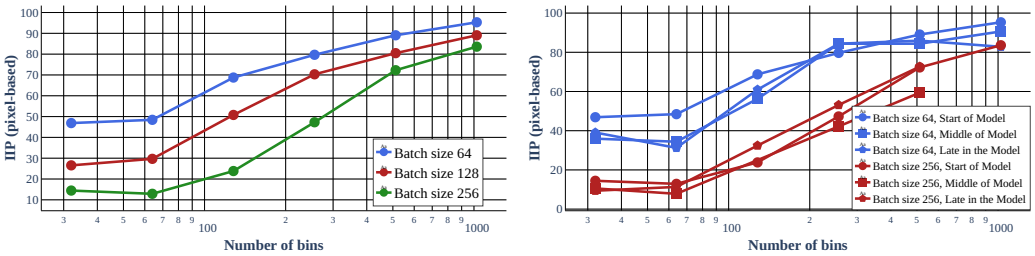


Figure 5: **Left:** Identification Success vs bin size. **Right:** Identification Success (via IIP score) vs. bin size and position in a ResNet-18 model. We find that the attack is stable over a range of batch sizes and positions in a model.

(TV), image registration, matching batch norm statistics, and DeepInversion priors (Geiping et al., 2020; Yin et al., 2021). As previously mentioned, such strong regularizers do not always exist in other domains of interest, such as text or tabular data. The discussed imprint module, however, is data-agnostic, and while we focus on the image domain for the sake of comparison to previous works, nowhere do we use any assumptions unique to vision. In fact, linear layers often appear in language models, and tabular data models - cases in which the attacker only needs to modify parameters of an existing model to breach user privacy (Vaswani et al., 2017; Somepalli et al., 2021) without architecture modifications.

5 Potential Defense and Mitigation Strategies

If aggregation is the only source of security in a FL system, then the proposed attack breaks it, uncovering samples of private data from arbitrarily large batches, especially via the One-shot mechanism. In light of this attack, the effectiveness of secure aggregation is reduced to only *secure shuffling* (Kairouz et al., 2021): When private data is uncovered via the imprint module, based on data that has been securely aggregated, then the data is breached, but is not directly connected to any specific user (aside from possible revealing information in the data itself). A mitigation strategy for users that does not require coordination (or consent) of a central server is to employ local differential privacy (Dwork & Roth, 2013). Adding sufficient gradient noise can be a defense against this attack as the division in Eq. (2) leads to potentially unbounded errors.Yet, in practice, the amount of noise that has to be added is significantly large, and even with $\sigma = 0.01$, private data is visibly leaked as can be seen in Fig. 6.

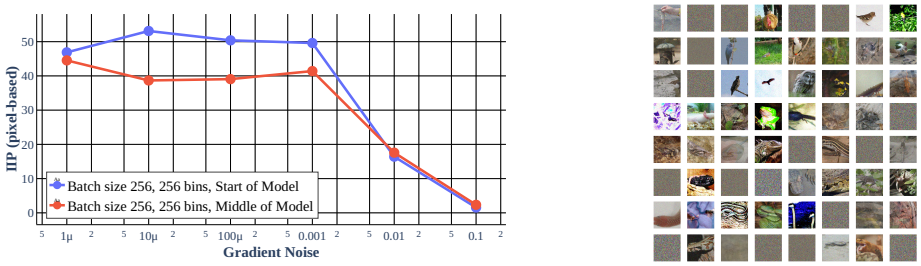


Figure 6: **Left:** IIP score vs Laplacian gradient noise. **Right:** Exemplary recovery for $\sigma = 0.01$. Recovery is stable for a large range of Laplacian gradient noise injections.

6 Conclusions

Federated learning offers a promising avenue for training models in a distributed fashion. However, the use of federated learning, even with large scale averaging, does not guarantee user privacy. Using common and inconspicuous machine learning modules, a malicious server can breach user privacy by sending minimally modified models and parameters in a federated setup. We hope that constructing these examples clarifies current limitations and informs discussions on upcoming applications, especially concerning API design.

ETHICS STATEMENT

In this work, we uncover an attack on federated learning that has the potential to compromise user privacy. While this method has the potential to be used for malicious purposes, the fundamental purpose of this research is to inform the community about the state of privacy in federated learning, and to help users and technical experts better understand the limitations of federated learning for the purpose of preserving user privacy.

REPRODUCIBILITY STATEMENT

We provide additional technical details and the proof Proposition 1 in the appendix. Further, we provide python code to replicate our experiments. The experiments in this work require no GPU resources and can be cheaply evaluated on almost any machine with sufficient RAM.

REFERENCES

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. Technical Report 281, 2017.

Patrick Cason. *Announcing 4 New Libraries for Federated Learning on Web and Mobile Devices*, 2020. `https://blog.openmined.org/announcing-new-libraries-for-fl-on-web-and-mobile/`.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning. *arXiv:2003.04297 [cs]*, March 2020.

Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013. ISSN 1551-305X, 1551-3068. doi: 10.1561/0400000042.

Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, Chang Liu, Chee Seng Chan, and Qiang Yang. Rethinking Privacy Preserving Deep Learning: How to Evaluate and Thwart Privacy Attacks. In *Federated Learning: Privacy and Incentive*, Lecture Notes in Computer Science, pp. 32–50. Springer International Publishing, Cham, 2020. ISBN 978-3-030-63076-8. doi: 10.1007/978-3-030-63076-8_3.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients - How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, December 2020.

Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, Cambridge, 1st edition edition, September 2009. ISBN 978-0-521-11991-7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015.

Jinwoo Jeon, Jaechang Kim, Kangwook Lee, Sewoong Oh, and Jungseul Ok. Gradient Inversion with Generative Image Prior. In *International Workshop on Federated Learning for User Privacy and Data Confidentiality*, pp. 12, 2021.

Arthur Jochems, Timo M. Deist, Johan van Soest, Michael Eble, Paul Bulens, Philippe Coucke, Wim Dries, Philippe Lambin, and Andre Dekker. Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital – A real life proof of concept. *Radiotherapy and Oncology*, 121(3):459–467, December 2016. ISSN 0167-8140, 1879-0887. doi: 10.1016/j.radonc.2016.10.002.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He,

Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]*, March 2021.

Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated Optimization:distributed Optimization Beyond the Datacenter. *arXiv:1511.03575 [cs, math]*, November 2015.

Brendan McMahan and Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data, April 2017.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv:1602.05629 [cs]*, February 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. *arXiv:1805.04049 [cs]*, November 2018.

Xudong Pan, Mi Zhang, Yifan Yan, Jiaming Zhu, and Min Yang. Theory-Oriented Deep Leakage from Gradients via Linear Equation Solver. *arXiv:2010.13356 [cs, stat]*, October 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*, Long Beach, CA, 2017.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. Technical Report 715, 2017a.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-Preserving Deep Learning: Revisited and Enhanced. In *Applications and Techniques in Information Security*, Communications in Computer and Information Science, pp. 100–110, Singapore, 2017b. Springer. ISBN 978-981-10-5421-1. doi: 10.1007/978-981-10-5421-1_9.

Jia Qian and Lars Kai Hansen. What can we learn from gradients? September 2020.

Jia Qian, Hiba Nassar, and Lars Kai Hansen. Minimal conditions analysis of gradient-based reconstruction in Federated Learning. *arXiv:2010.15718 [cs, eess]*, March 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y.

Reza Shokri and Vitaly Shmatikov. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pp. 1310–1321, Denver, Colorado, USA, 2015. ACM Press. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813687.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017.

Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, November 2018. doi: 10.1098/rsta.2018.0083.

Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User Label Leakage from Gradients in Federated Learning. *arXiv:2105.09369 [cs]*, June 2021.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. *arXiv:1812.00535 [cs]*, December 2018.

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*, pp. 16337–16346, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, May 2016.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved Deep Leakage from Gradients. *arXiv:2001.02610 [cs, stat]*, January 2020.

Junyi Zhu and Matthew Blaschko. R-GAP: Recursive Gradient Attack on Privacy. *arXiv:2010.07733 [cs]*, March 2021.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems 32*, pp. 14774–14784. Curran Associates, Inc., 2019.

A APPENDIX

A.1 PROOF

Proposition. *If the server knows the CDF (assumed to be continuous) of some quantity associated with user data that can be measured with a linear function $h : \mathbb{R}^m \to \mathbb{R}$, then for a batch size of n, and a number of imprint bins $k > n > 2$, by using an appropriate combination of linear layer and ReLU activation, the server can expect to recover*

$$\frac{1}{\binom{k+n-1}{k-1}}\left[\sum_{i=1}^{n-2} i \cdot \binom{k}{i} \cdot \left(\sum_{j=1}^{\lfloor \frac{n-i}{2} \rfloor} \binom{k-i}{j}\binom{n-i-j-1}{j-1}\right)\right] + r(n,k)$$

amount of user data (where the data is in $\mathbb{R}^m$) perfectly. Note: $r(n,k)$ is a correction term (see proof for expression).

Proof. By construction of the imprint module, given a random sample (batch) $X_1, \ldots, X_n$ (iid) the server perfectly recovers data whenever an imprint bin has exactly 1 element of the batch. Because we know the CDFs, we can create partitions of equal mass corresponding to imprint bins $\{b_j\} = \{[a_j, b_j]\}$ where $P(X_i \in [a_j, b_j]) = 1/k \; \forall i, j$.

We can then phrase the problem of expected number of perfectly recovered samples as a modified "stars and bars" problem. For a given batch of data, to calculate the amount of data recovered, we first calculate:

$$\sum_{i=1}^{n} i \cdot \binom{k}{i} \cdot N_i$$

Where $\binom{k}{i}$ is the number of ways to select the i bins that have exactly 1 element, and N_i is the number of orientations of the remaining data into the remaining bins so that no bin has exactly 1 element. Note that we can do this because the bins all have equal mass, and thus we can factor out the (uniform) probability of any configuration from the sum.

Simply put, we first take the configuration where there is only 1 bin with exactly 1 element, and weight it by 1, then we take the number of configurations with 2 bins with exactly 1 element, and weight it by 2, and so on.

In order to calculate N_i, we notice that in our construction, once the i bins with exactly 1 element are chosen, every other bin has either 0 or ≥ 2 elements. We focus on the bins that have ≥ 2 elements. By a simple "reverse" pigeon hole argument, we can now have at most $\lfloor \frac{n-i}{2} \rfloor$ of the remaining bins containing any elements, as otherwise, one bin would be guaranteed to contain exactly 1 element.

So we further select any $1 \leq j \leq \lfloor \frac{n-i}{2} \rfloor$ number of the remaining $k - i$ bins all to contain at least 2 elements. Formally, this is equivalent to calculating the number of orientations of integers $\{x_l\}_{l=1}^{j}$

so that

$$x_1 + \cdots + x_j = n - i$$

constrained with $x_l \geq 2 \; \forall l$

Now, we make a change of variables to instead calculate the number of orientations of integers

$$\{p_l\}_{l=1}^{j}$$

so that

$$p_1 + \cdots + p_j = n - i - 2j$$

constrained with $p_l \geq 0 \; \forall l$. Now we just have a "stars and bars" problem with $k' = j$ bars and $n' = n - i - 2j$ stars. This reduces to:

$$\binom{n-i-j-1}{j-1}$$

orientations for the remaining bins with exactly j elements. Once these i bins with 1 element, and j bins with ≥ 2 elements are chosen, all the other bins are required to have 0 elements.

So adding these parts together, we have the expected amount data the server can expect to reconstruct perfectly becomes:

$$\frac{1}{\binom{k+n-1}{k-1}}\left[\sum_{i=1}^{n-2} i \cdot \binom{k}{i} \cdot \left(\sum_{j=1}^{\lfloor \frac{n-i}{2} \rfloor} \binom{k-i}{j}\binom{n-i-j-1}{j-1}\right)\right] + \overbrace{\frac{n}{\binom{k+n-1}{k-1}}\binom{k}{n} - \frac{n}{k}}^{r(n,k)}$$

We call the last two "residual" terms $r(n, k)$. The first of these terms corresponds to the term in the expectation where all elements of the batch end up in separate bins, and the second term is the expected number of elements that land in the tail of the CDF not covered in any bin. □

B Technical Details

All experiments were implemented in PyTorch (Paszke et al., 2017) and were run on several laptop and machine CPUs, as the reconstruction itself requires only a few tensor operations. Especially, compared to optimization-based reconstruction techniques, this makes the approach significantly faster and significantly more portable. For visualization purposes and to measure accurate PSNR and IIP scores all images (which are recovered in the order given by the chosen function h) are matched with possible correspondences in the ground truth batch. The matching is found based on LPIPS feature similarities scores (Zhang et al., 2018) which are matched using a linear sum assignment solver. No labels are recovered using the proposed approach which is entirely label-agnostic, but labels could be assigned a-posteriori using model predictions of the reconstructed data if required. For experiments where the imprint module is placed deeper into a network, the network (which is here a ResNet) is linearized by resetting batch normalization parameters and buffers to the identity map, setting all residual paths to zero and initializing the first convolution and the shortcut convolutions to identity maps. The nonlinearities can be bypassed by bias shifting.

PSNR scores are computed as average PSNR where we first compute PSNR scores per image and then average. This procedure is standard in computer vision, but does bias the score toward successful reconstructions, as the minimal PSNR score is bounded at 0, but its potential upside unbounded. For the image identifiability precision (IIP) score of Yin et al. (2021) we implement the score as proposed therein, but measure nearest neighbors not in the model feature space (which we consider biased, given that the model parameters are already used for reconstruction), but directly in image pixel space, where we check whether the given reconstruction is indeed closer to its true counterpart in euclidean distance than any other image from this class in the validation set.

C Additional Images

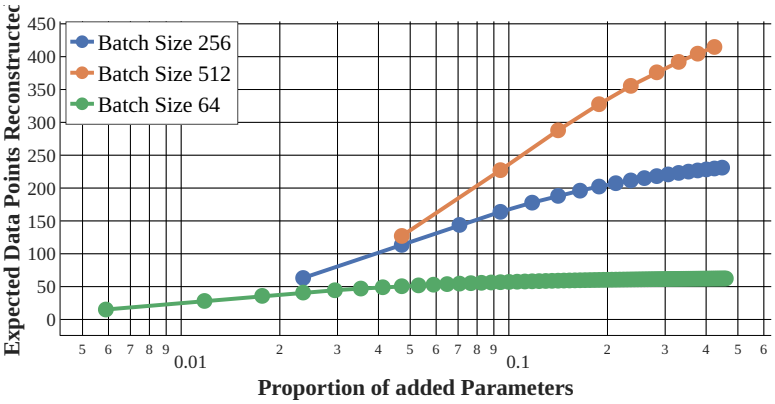


Figure 7: Expected number of data points recovered for several batch sizes and increased bins and corresponding parameter increase. Model: ResNet50 with ImageNet, targeting the input to the 3rd residual block.

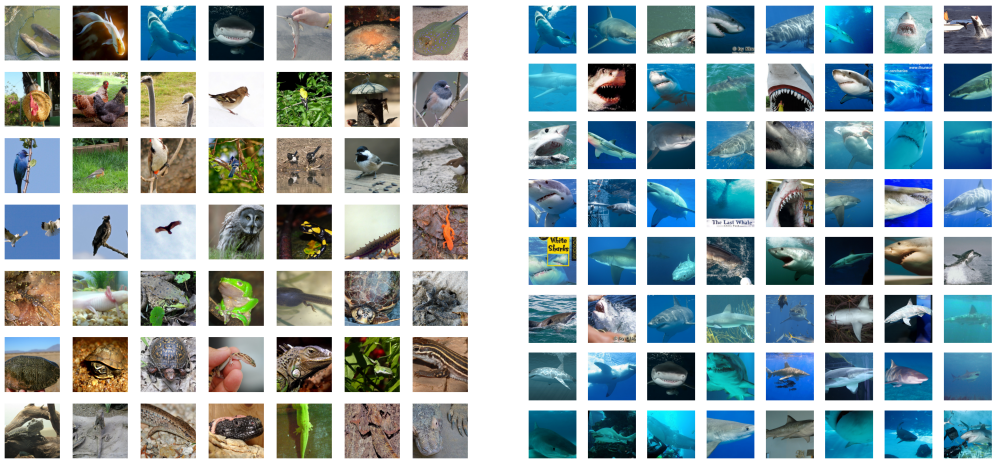


Figure 8: **Left:** Raw data for the 64 ImageNet images with separate classes. **Right:** Raw data for the 64 images from the white shark class.

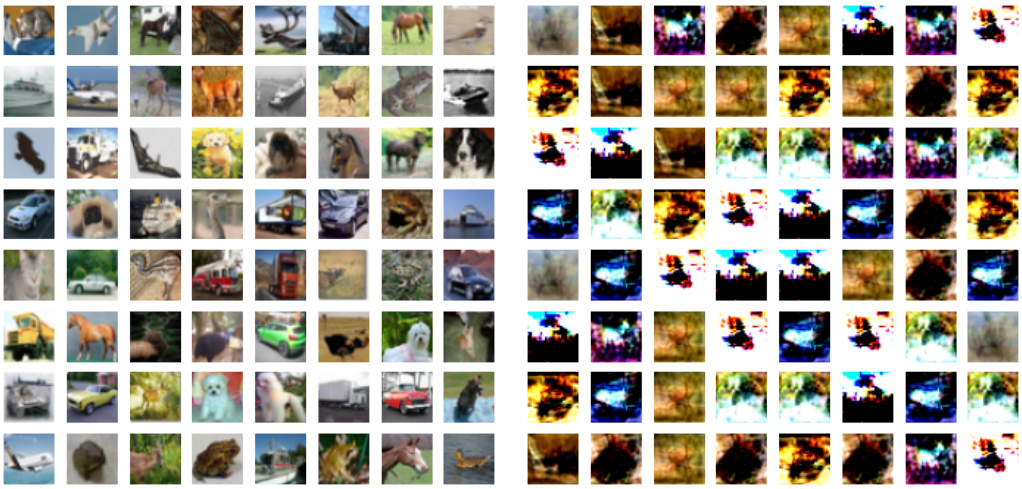


Figure 9: **Left (a)**: A batch of 64 CIFAR10 images. **Right (b)**: The same batch of images reconstructed naively using Eq. (2).

This section contains additional image examples, such as using CIFAR-10 in Fig. 9 and Fig. 10 where an attack with the imprint module on this dataset shows that almost all of the user data is perfectly recovered. Furthermore, example panels of imprint modules inserted in later ResNet layers are visualized as well as the results of the sparse variant that is used to attack a federated averaging scheme.

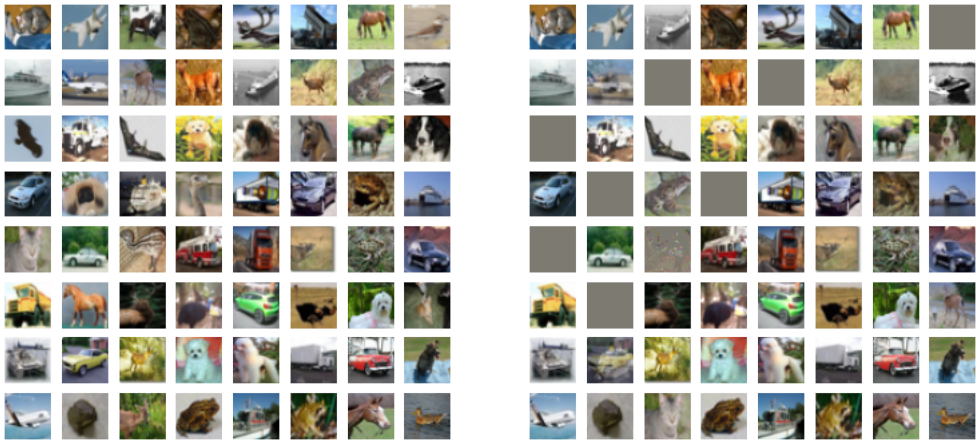


Figure 10: **Left (a)**: A batch of 64 CIFAR10 images. **Right (b)**: The same batch of images reconstructed using the imprint module with 300 bins. Gray images can result from collisions within a given bin.

Figure 11: Different placements of the imprint module in a ResNet-18. From left to right: Before the first block ($56x56$), before the third block ($28x28$), before the fourth block ($14x14$) and before the last average pooling ($7x7$). Compare to a placement before the first convolution ($224x224$) and raw input data in Fig. 3. Enlarged versions of each panel can be seen in Figs. 12 - 15

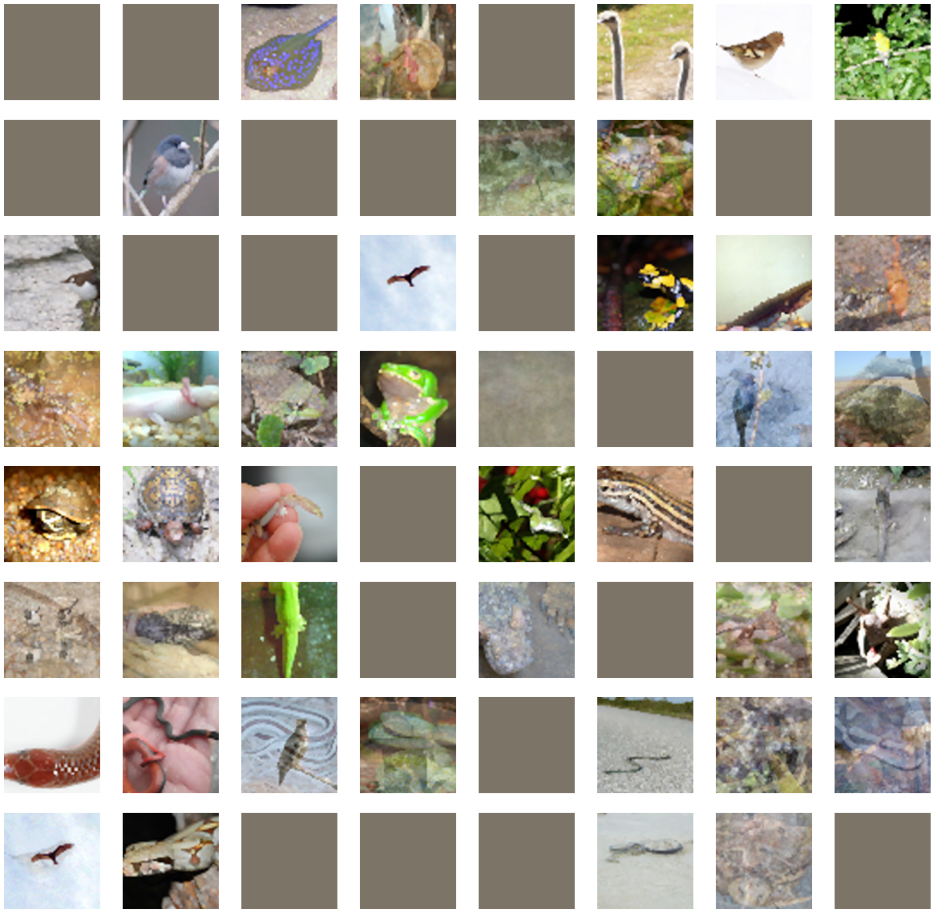


Figure 12: Different placements of the imprint module in a ResNet-18. Before the first block (56×56)

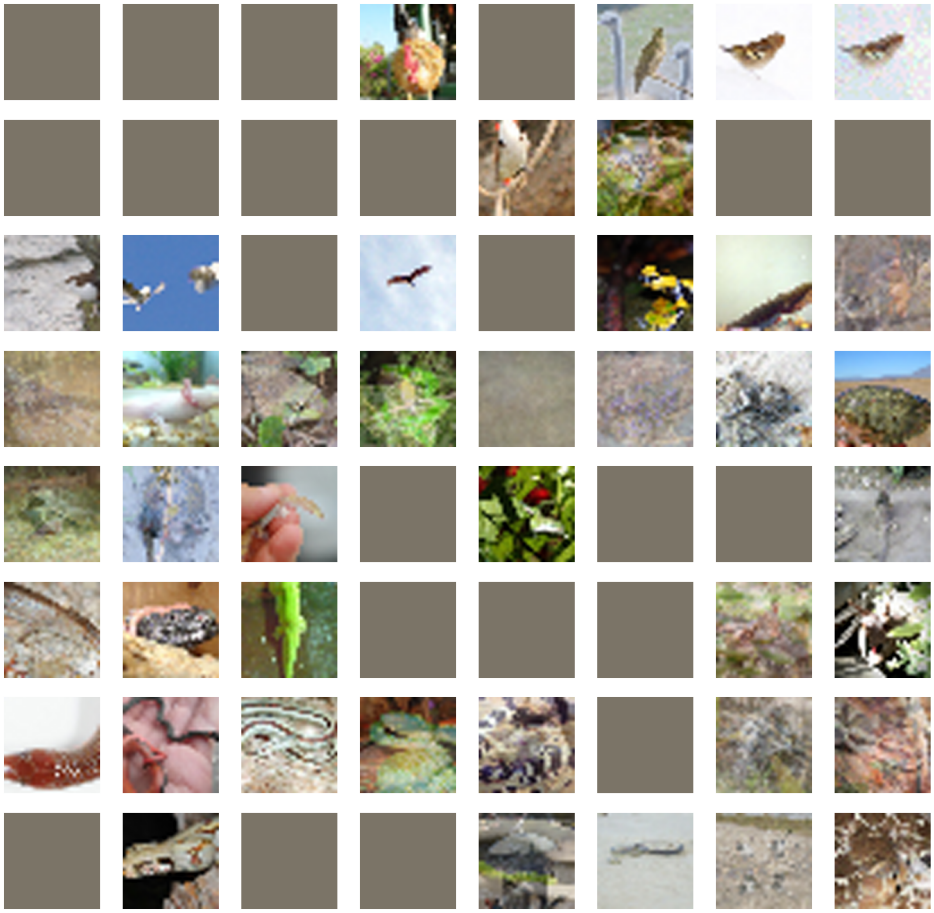


Figure 13: Different placements of the imprint module in a ResNet-18. Before the first block (28×28)

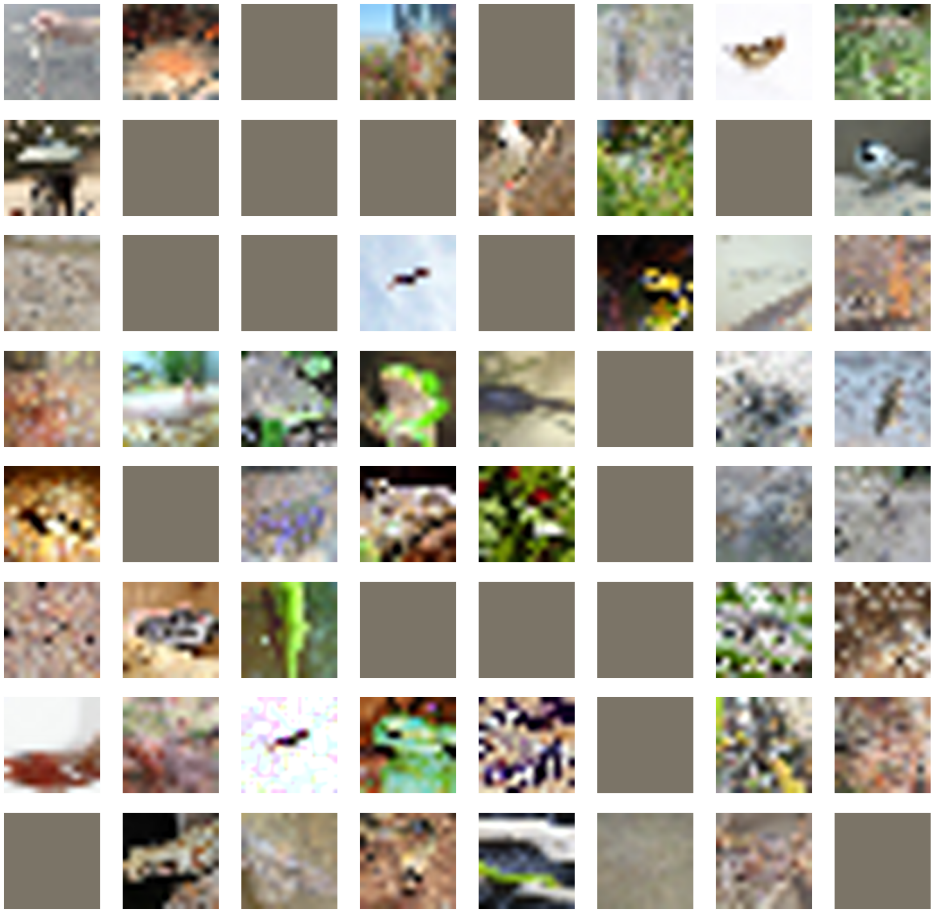


Figure 14: Different placements of the imprint module in a ResNet-18. Before the first block (14×14)

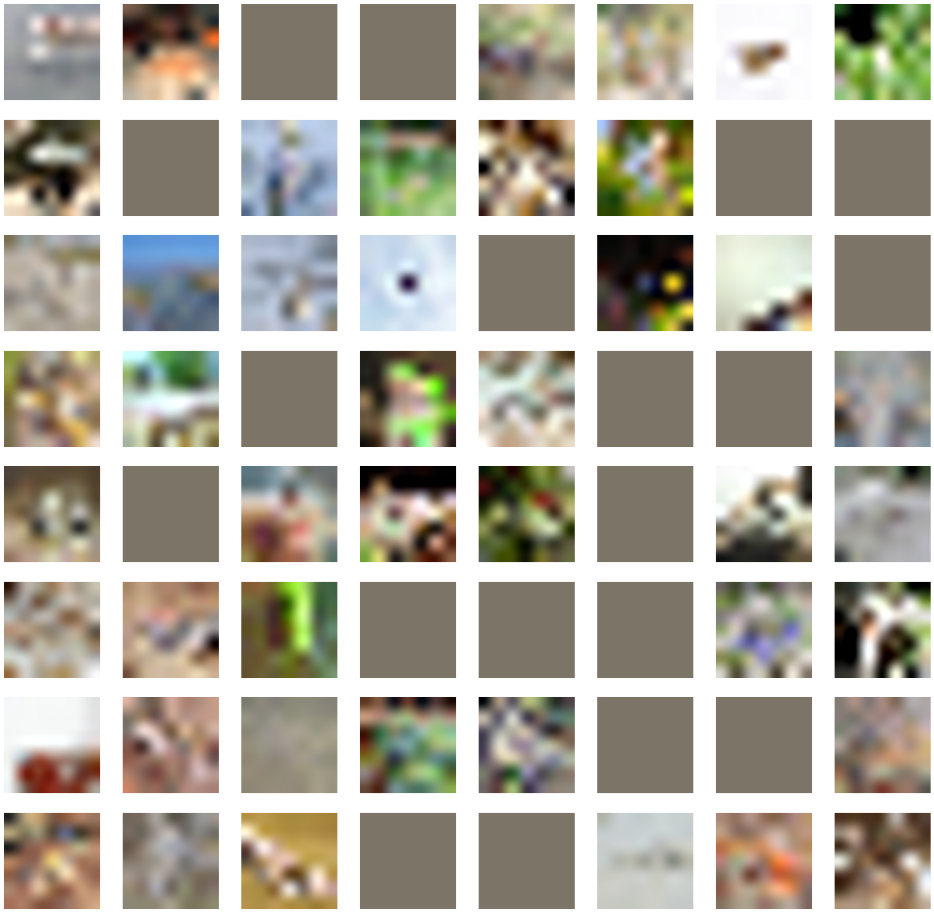


Figure 15: Different placements of the imprint module in a ResNet-18. Before the first block (7×7)

Figure 16: Results for federated averaging for 8 steps with 8 images each, i.e. 64 unique data points, and 128 bins. PSNR: 27.37. IIP 59.38%.