Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification

Yuxin Wen [*1] Jonas Geiping [*1] Liam Fowl [*1] Micah Goldblum [2] Tom Goldstein [1]

Abstract

Federated learning (FL) has rapidly risen in popularity due to its promise of privacy and efficiency. Previous works have exposed privacy vulnerabilities in the FL pipeline by recovering user data from gradient updates. However, existing attacks fail to address realistic settings because they either 1) require toy settings with very small batch sizes, or 2) require unrealistic and conspicuous architecture modifications. We introduce a new strategy that dramatically elevates existing attacks to operate on batches of arbitrarily large size, and without architectural modifications. Our model-agnostic strategy only requires modifications to the model parameters sent to the user, which is a realistic threat model in many scenarios. We demonstrate the strategy in challenging large-scale settings, obtaining high-fidelity data extraction in both cross-device and cross-silo federated learning. Code is available at https://github.com/JonasGeiping/breaching.

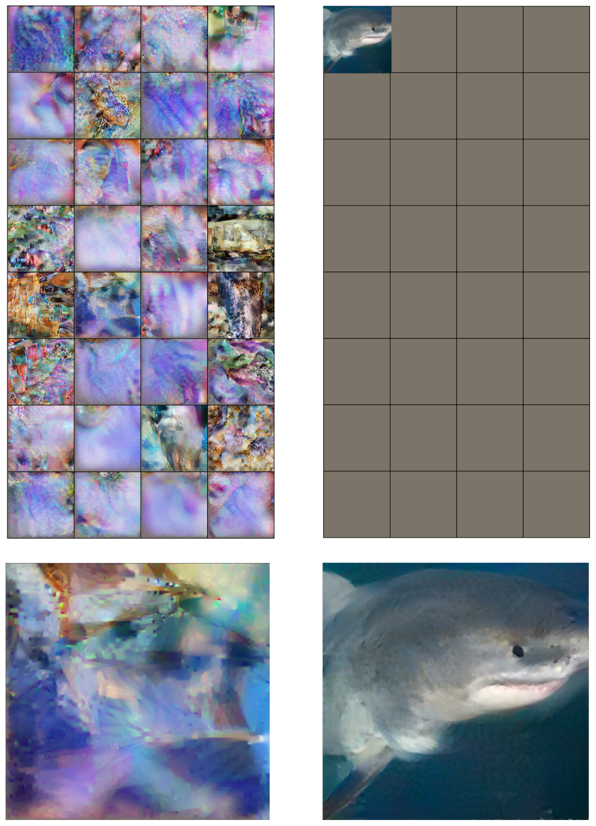


Figure 1. Previous attacks (left) for a batch size of 32 in the "honest-but-curious" threat model, vs. a fishing attack (right) under the "malicious-parameters" threat model. Malicious modifications of the server update lead to the complete compromise of a single example which is *fished* out of an arbitrarily large batch of aggregated data.

1. Introduction

Is it possible to train machine learning models on massive amounts of private data without compromising the data and feeding it to a central server? Federated learning is poised to be one step towards a solution to this question.

In federated learning (FL) and other collaborative learning frameworks, machine learning models are trained in a decentralized manner. A central server sends only the current state of the model to all participating users, the users compute model updates based on their own private data, and only their update is returned to the server, optionally after being securely aggregated across multiple users (Kairouz et al., 2021; Bonawitz et al., 2017). This is often referred to as the principle of data minimization - the server never sees user data directly, and model updates sent by the user are considered to only contain the minimal information necessary to improve the model (Bonawitz et al., 2021). However, federated learning is only secure as long as the model updates sent by users cannot be inverted to retrieve their original contents. Such attacks breach privacy in federated learning and directly reveal user data. Previously, such attacks have been demonstrated mostly on either small batches of data,

[*]Equal contribution [1]University of Maryland [2]New York University. Correspondence to: Yuxin Wen <ywen@umd.edu>, Jonas Geiping <jgeiping@umd.edu>, Liam Fowl <lfowl@umd.edu>, Tom Goldstein <tomg@umd.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

e.g. in Wang et al. (2018); Zhu et al. (2019) under the threat model of an "honest-but-curious" server or in settings with "malicious model" threat models with modified architectures (Fowl et al., 2021; Boenisch et al., 2021). For the threat model of an unmodified architecture, existing work seems to suggest that even a moderate batch size of 32 could be sufficient to deter an attack (Huang et al., 2021).

In this work, we discuss a third category of threat model, "malicious parameters". In this setting, the server cannot be trusted to send benign model updates to users. From a user perspective, this is a realistic scenario; the server update is outside of their control, computed on central machines by a company with potentially limited intent to uphold privacy. Such a company could publicly support federated learning, for example to comply with regulations such as GDPR (Truong et al., 2021), but still obtain private user information through attacks such as the one we discuss hereafter. As such, we argue that a user should regard server updates as untrusted.

We introduce a family of attacks that are able to fish for single data points from a large aggregate, magnifying the gradient contribution of a target data point and reducing the gradient contribution of other data through maliciously chosen parameter vectors. We discuss this attack for both common federated learning scenarios of cross-device and cross-silo learning. We then verify in a range of experiments for the task of image classification that this attack allows us to leverage existing optimization-based (Geiping et al., 2020) and analytic attacks (Lu et al., 2021), which currently only work well for inverting an updated calculated on a single or few data points.

2. Background and Related Work

We are interested in two main settings of federated learning (Kairouz et al., 2021). In the *cross-device* scenario, a shared model is trained on a massive number of users, e.g. mobile devices, which each hold a small amount of private data, but are highly unreliable and on average are not expected to participate more than once in training given their number (Bonawitz et al., 2019). In particular, these users cannot be queried or addressed separately, and are selected at random. To increase privacy, contributions of groups of users can be aggregated securely (McMahan et al., 2017). In contrast, in the cross-silo scenario, only a small group of users (e.g. < 100) exists, who are reliable and identifiable and can be queried in every round of federated learning. An often-quoted example here is that consortia of hospitals train models without permission to share sensitive patient data directly (Jochems et al., 2016). These users each own large amounts of private data, which they aggregate themselves and protect in this way without having to rely on aggregation with other users.

Gradient inversion attacks against federated learning systems have been demonstrated for some time now and in several scenarios. Most work so far though has focused on the threat model of an "honest-but-curious" server, in which the server participates in the federated learning protocol as designed, but records and processes updates sent by users. Phong et al. (2017); Geiping et al. (2020) discuss how analytical methods can recover the input data to a linear layer anywhere in the network. Analytical attacks can also directly recover input data for several vision transformer architectures (Lu et al., 2021). For CNNs, analytical attacks can be extended recursively (Zhu & Blaschko, 2021) to resolve input data from convolutional layers. However, all of these attacks can only invert the gradients of single data points, recovering a mixture of all aggregated data, when the gradient is aggregated over many examples as in realistic settings as discussed before.

Optimization-based attacks as introduced in Wang et al. (2018) fare better, but can also only invert gradients for small batch sizes, such as 8 in (Zhu et al., 2019). Other attacks can recover a few samples from larger batches when attacking larger models Geiping et al. (2020); Yin et al. (2021), but overall, these methods also cannot invert gradients of large batches. This limits the attack vector of these attacks and allows for an effective defense through (secure) aggregation of user gradients.

Recently, other threat models have also come into focus, such as threat models focused on malicious parameter vectors. As discussed, this is a realistic threat for users who should view any updates sent by the server as unverified parameters that the users deploy directly on their private data. Attacks in these threat models can recover data from arbitrarily large batch sizes as shown in Fowl et al. (2021) and as such overcome secure aggregation (more formally, the notion that aggregation itself is a defense is overcome, reducing secure aggregation in these scenarios to a multi-party shuffle - the outcome of secure aggregation is attacked, but not the protocol itself) . However, this previous work often requires unrealistic model modifications. Attacks in Fowl et al. (2021) and Boenisch et al. (2021) can only recover data from linear layers and thus require the existence of large linear layers - even one of which could be larger than standard vision models - early in a network, to avoid downsampling and striding operations that degrade information. The attack in Pasquini et al. (2021) requires the ability of the server to send separate malicious parameters to individual users who each own only few data points - a threat that can in turn be overcome quickly if aggregation protocols are used in reverse to average server updates before processing them on the user side. The attack in Lam et al. (2021) can break secure aggregation without these previous assumptions, but requires some knowledge of side-channel information that is not strictly necessary for the FL protocol.

The attacks in Fowl et al. (2021); Boenisch et al. (2021) and Pasquini et al. (2021) can collectively be understood as attacks based on gradient sparsity. Although the aggregation protocol nominally runs as it was designed, all but one of the data points (Boenisch et al., 2021) or users (Pasquini et al., 2021) return a zero gradient for some parameters in the model, such that averaging still returns these entries directly. In Fowl et al. (2021), individual gradients are non-zero, but parts of the gradient obey a cumulative sum structure from which a sparse single update can be recovered. These attacks dense aggregation into a sparse aggregation and extract separate gradients per data point, which can then be inverted by the methods described above.

In this work, we show that this methodology can be extended much more generally that previously believed. We describe attack strategies that can fish for single gradients from arbitrarily large aggregated batches and, crucially, apply to arbitrary models, without requiring the model modifications of Fowl et al. (2021), and without requiring separate updates sent to all users Pasquini et al. (2021), or knowledge of side-channel information as in Lam et al. (2021). Even attacks such as Lu et al. (2021) that previously were not applicable to more than one data point, can now threaten arbitrary large aggregations.

3. Method

In this section, we describe the two straightforward mechanisms of our attack strategy. These two aspects allow an attacker to effectively reduce a aggregated gradient to an update calculated on a *single* data point. These combined strategies allow an attacker to breach privacy in both cross-device and cross-silo settings, even with large batches. In this work, we focus on tasks that can be posed as multi-class classification problems.

Threat Model: In this work we assume that users communicate with the server through established federated learning protocols (we focus mainly on *fedSGD*) which are orchestrated in secure compute environments (Frey, 2021) leaving the protocol itself as the only avenue of attack for the server. The server is intent on recovering private user information and is allowed to send modified updates to all users. In the cross-device setting, these updates can be sent to multiple groups of users, but each user is not expected to be queried more than once. In the cross-silo setting, modified updates can be sent successively as queries over multiple rounds of training, as all users participate in all rounds in this setting.

3.1. Class fishing strategy

We first introduce a class fishing strategy wherein the gradients for a single target class dominate the aggregated model update. The intuition behind this strategy is for the malicious server to artificially decrease the confidence of the network's predictions for the target class, thus significantly increasing the contribution of the gradient information calculated from data belonging to the selected class. In fact, the server can make the relative size of the selected class' gradient arbitrarily large in this manner.

More formally, given a target class c, and fully connected classification layer $W \in \mathbb{R}^{n\times m}$ with bias $b \in \mathbb{R}^n$, the class fishing strategy modifies the parameters

$$W_{i,j} = \begin{cases} W_{i,j}, & \text{if } i = c \\ 0, & \text{otherwise} \end{cases}$$

$$b_i = \begin{cases} b_i, & \text{if } i = c \\ \alpha, & \text{otherwise} \end{cases},$$

where α is a server-side hyperparameter which controls the relative magnitude of the selected class's gradient signal. Let x_i denote an image from target class c. If α is chosen so that $\langle W_j, x_i\rangle + b_j = \alpha,\ \forall j \neq i$, and $\langle W_i, x_i\rangle + b_i \ll \alpha$, then we have that the ratio $\frac{p_i}{p_o} \ll 1,\ \forall o \neq i$ where p_i is the softmax output for x_i corresponding to the target class c, and p_o denotes the softmax output of any other image in the batch corresponding to its ground-truth label ($\neq c$). That is to say, if the server chooses a large enough bias α for the non-target classes, the ratio of the ground-truth class' softmax entry for an image coming from the target class is much lower than the ground-truth class' softmax entry for any other class. This leads to a predictable model update that is dominated by the contributions of one class. We quantify this in Proposition 3.1 which decomposes a model update that is aggregated over images from many different classes as being almost exclusively from the target class.

Proposition 3.1. *Let ϕ denote the parameters of the feature extractor for the central model, and let W, b be the weight and bias vectors, respectively, of the last classification layer. Furthermore, let $\{x_i\}_{i=1}^N$ be the collection of user data on which model updates can be calculated. Let $\{x_i\}_{i=1}^{m\leq N}$ be any batch data containing k data points $\{x_i'\}_{i=1}^{k\leq m}$ from the target class. Then, for cross-entropy loss $\mathcal{L}$, if the gradient of $\mathcal{L}$ w.r.t. each data point in the batch is bounded in l_2 norm, then $\forall\varepsilon > 0$, $\exists\alpha$ so that a malicious server employing our class strategy on W, b recovers a feature extractor update g so that*

$$\|g - \frac{1}{m}\sum_{i=1}^{k} \nabla_\phi \mathcal{L}(x_i', c; \phi, W, b)\|_2 < \varepsilon.$$

Interestingly, the choice of parameters defined above already improves the performance of optimization-based attacks such as Geiping et al. (2020), even when operating on a single image.

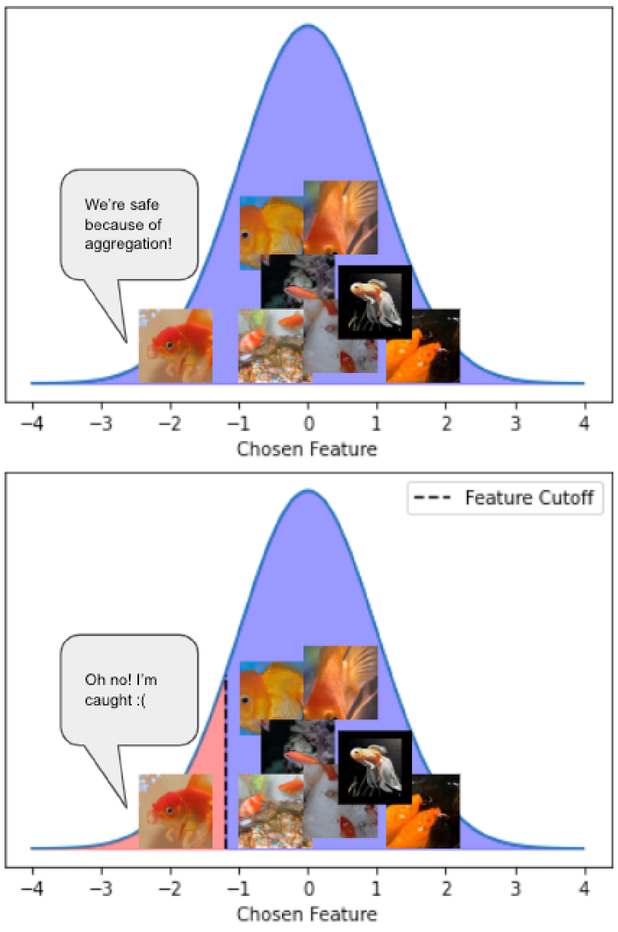


Figure 2. A high-level schematic of our one-shot binary attack. Within a class, "Goldfish", features can be suppressed to reveal a single user image.

3.2. Feature fishing strategy

Our previous class fishing strategy has a good chance of succeeding when the number of classes in the underlying label-space is large compared to the batch size on which updates are calculated. This is the case for the setting of previous attacks which have focused on inverting model updates for ImageNet (Yin et al., 2021). However, if there are intra-batch collisions in label space, or other users also have data from the target class, another strategy is needed as solely applying the class fishing strategy results in a "mixed" update on all images from the target class. To that end, we introduce a feature fishing strategy to selectively retrieve gradients for images with the same class label in the same batch. Similar to the class fishing attack, this strategy allows a malicious server to magnify the gradients for a target image by appropriately shifting the decision boundary to decrease the confidence of the network's predictions for said target images. Like the class fishing strategy, the malicious server only needs to modify the parameters of the classification layer.

While this strategy can succeed by adjusting the decision boundary in many ways, we opt to focus on selecting a *single* feature along which to move the boundary. In this case, all other features are ignored by the network. Formally, given a target class c, a pre-selected feature entry index k, and fully connected classification layer $W \in \mathbb{R}^{n\times m}$ with bias $b \in \mathbb{R}^n$, the feature fishing strategy modifies the parameters

$$W_{i,j} = \begin{cases} \beta, & \text{if } i = c \text{ and } j = k \\ 0, & \text{otherwise} \end{cases}$$

$$b_i = \begin{cases} -\beta * \theta, & \text{if } i = c \\ 0, & \text{otherwise} \end{cases},$$

where θ and β are two server-side hyperparameters for choosing the target images and controlling the relative magnitude of the target images' gradient signal.

To understand this construction, we consider a simple example case of two images x_{i_1}, x_{i_2} where the average of feature of the two images at entry j is known to be $\overline{f}_j = \frac{f_{i_1,j}+f_{i_2,j}}{2}$, and for simplicity, we assume $f_{i_1,j} < f_{i_2,j}$. A malicious server can choose θ equal to $\overline{f}_j$ so that with this feature fishing strategy, $\langle W_c, f_{i_1}\rangle + b_c = -l < 0$ and $\langle W_c, f_{i_2}\rangle + b_c = l > 0$, where $l = (f_{i_2,j} - \overline{f}_j) * \beta$. In this case, if the malicious server chooses a large enough β, the ratio of the ground-truth class' softmax entry for x_{i_1} coming from the target class is much lower than the ground-truth class' softmax entry for x_{i_2}. Such cutoff also can be applied to more than two images in a straightforward way. We theoretically motivate this strategy in Proposition 3.2 by showing that, if implemented properly, the server can often diminish the impact of all but one image on the gradient contribution for the target class.

Proposition 3.2. *If the server knows the CDF (assumed to be continuous) for the distribution of any feature of interest for the target class, then for any batch of data $\{x_i\}_{i=1}^N$ with $\{x_{i_j}\}_{j=1}^{M\leq N}$ images coming from the target class, if the gradient of $\mathcal{L}$ w.r.t. each data point in the batch is bounded in l_2 norm (where $\mathcal{L}$ is cross entropy loss), then we have that for the proposed feature fishing strategy, with probability $p \geq \frac{1}{e}$, there exists a data point $x_* \in \{x_{i_j}\}$ so that*

$$\|g_c - \frac{1}{N}\nabla_\phi \mathcal{L}(x_*, c; \phi, W, b)\|_2 \leq \varepsilon,$$

where g_c is the component of the model update coming from data from the target class.

The missing ingredient for the server now is *how* to find the distribution of a feature of interest from the users, so that an optimal cutoff can be chosen as visualized in Figure 2. What helps the server here is that the mean feature value over a batch of data can always be recovered from the gradient of such data analytically (Phong et al., 2017; Geiping et al., 2020). This allows the server to measure the feature live, without the need for any auxiliary data. In the following sections we describe schemes to estimate the optimal cutoff in cross-device and cross-silo scenarios.

Remark 3.3. In order to successfully implement this feature fishing strategy, the attacker needs to be able to magnify the logits of a chosen feature and class over other classes. For example, to breach privacy for a fish image (as in Figure 2), the attacker must magnify the logits corresponding to the "fish" class for all images except the one that is ultimately compromised. In the scenario when there are other classes present in a user update, the attacker must accomplish this without inadvertently increasing the loss for non-targeted classes. A sufficient condition for an attacker to do this is to find a feature along which some cutoff can be made so that all data from a single class alone lies on one side of this cutoff. This allows the attacker to freely manipulate the logit for this class without increasing the loss for non-targeted data. Empirically, we find that this happens naturally in later rounds of training, especially when choosing to attack a feature with a large average value. However, we believe that future adjustments to our strategy are possible so that the feature fishing can occur at any stage of training.

3.3. Cross-device

In this subsection, we will illustrate how the strategies of class fishing and feature fishing enable a malicious server to successfully breach privacy in cross-device federated learning (Kairouz et al., 2021), where a server has an opportunity to train a model on the data from a massive number of users, yet the server might never visit the same user twice.

Because of the large number of users in cross-device FL, we propose an attack that first estimates the feature distribution from some group of users, and then applies the feature fishing strategy to catch only one data point from each of the remaining users. We call this method one-shot feature attack.

With the class fishing strategy, a malicious server can easily get the average feature of the images with the target class c from a batch: Given the gradient updates for weight and bias in the classification layer, $\nabla_W \mathcal{L}$ and $\nabla_b \mathcal{L}$, the average feature can be recovered as (Phong et al., 2017):

$$\overline{f} = \frac{\nabla_{W_c}\mathcal{L}}{\nabla_{b_c}\mathcal{L}}.$$

A malicious server first applies the class attack to recover collections of average features $\{\overline{f}_i, s_i\}_{i=1}^N$, where s_i is the number of images with target class c for user i. Out of the

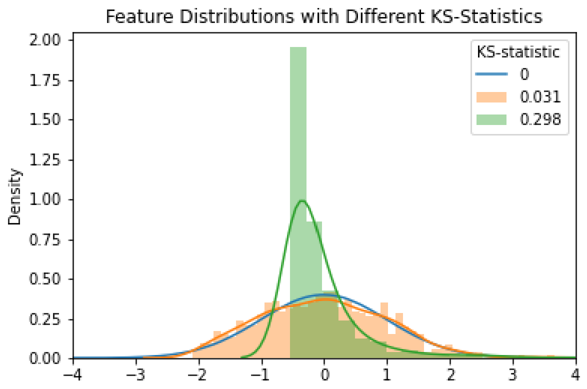


Figure 3. The comparison between normalized feature distributions with different KS-statistics. In the feature space of classification models, features exist which are abnormally distributed, i.e. the feature with KS-statistics of 0.298, but other features are close to normal, i.e. with KS-statistics of 0.031, and can be well estimated by an attacker.

m-dimensional feature vectors recorded in this manner, the attacker needs to choose a single feature for the attack. We argue that a strong choice for the attacker is to choose this feature j via:

$$j = \arg\min_k \mathrm{KS}(\mathrm{Norm}(\overline{f}_{i,k}), \mathcal{N}(0, 1)),$$

where $\mathrm{Norm}()$ evaluates the CDF of normalized data and $\mathrm{KS}()$ returns the statistic of the Kolmogorov-Smirnov test on two CDFs. Then, the CDF of feature entry j is estimated as a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\mu = \frac{1}{N}\sum_{i=1}^N \overline{f}_{i,j}$ and $\sigma = \sqrt{\frac{\sum_{i=1}^N (\overline{f}_{i,j}\sqrt{s_i} - \mu')^2}{N}}$, where $\mu' = \frac{1}{N}\sum_{i=1}^N (\overline{f}_{i,j}\sqrt{s_i})$.

The attack employs the Kolmogorov-Smirnov test to quantify the "normality" of the distribution of the chosen feature. Overall, the attacker has limited capacity to accurately model a complicated feature distribution, which might occur for a variety of reasons. For example, for many classification models like ResNets (He et al., 2015), the ReLU before the classification can lead to a bi-modal distribution for some features, with a large mode at 0. However, the attacker can circumvent this by simply choosing the most normal feature via the KS-Test. For example, in Figure 3, the CDF of the feature with KS-statistic 0.298 is hard to estimate using the sampling means, but the feature with KS-statistic 0.031 is almost perfectly normally distributed and can be estimated effectively from limited measurements $\overline{f}_{i,k}$.

3.4. Cross-silo

In cross-silo federated learning (Kairouz et al., 2021), a server is generally querying all or most users in each round.

Each user is identifiable by the server. In this setting, there might not exist enough users to reliably estimate the feature distribution from their updates, and feature distributions might be significantly different between participants. However, in this scenario, an attacker can abuse the ability of the server to query in every round to iteratively update their attack. Such a *binary attack* can eventually retrieve all gradients of each individual data point on the user side. The overall idea of this strategy is that if a malicious server moves the decision boundary to the average feature value $\overline{f}_j$, then they can wait for the next user update and retrieve the average feature of data points with feature value less than $\overline{f}_j$ and the average feature of data points with feature value more than $\overline{f}_j$, as well as their averaged gradient update. Iteratively, the server is thus able to construct a binary tree to find all cutoffs, retrieve all cumulative averaged gradients, and calculate the single gradient for each data point. Detailed implementation can be found in Algorithm 1. Instead of a full binary attack over a large number queries, the server can also aim to recover one data point, which we will call the one-shot binary attack and visualize in Figure 2.

Under the assumption that attacked user holds n data points with the target class c, and the feature is uniformly distributed at selected feature entry j, the binary attack requires $\mathcal{O}(n \log n)$ queries on average to retrieve all separate per-example gradients, and the one-shot binary attack requires $\mathcal{O}(\log n)$ queries on average. In the worst case where each cut of the binary tree only removes a single data point, $\mathcal{O}(n^2)$ queries are necessary for the full binary attack and $\mathcal{O}(n)$ for the one-shot binary attack. However, we find this scenario empirically less likely as, often, the feature distribution appears normal.

Algorithm 1 Binary Attack

```
 1: Input: model $m$, $\theta$, target class $c$, feature index $j$, batch size $n$
 2: visited ← Empty Set
 3: $G$ ← Empty OrderedDict in descending order
 4: $C \leftarrow [+\infty]$
 5: while $C$ is non-empty do
 6:     $C_{\text{new}} = [\,]$
 7:     for $c_{\text{cut}}$ in $C$ do
 8:         $m$ = FeatureFishing$(m, \beta = c_{\text{cut}}, \theta = \theta)$
 9:         $g$ = gradient update from the user with model $m$
10:         $f = \nabla_{W_{c,j}} \mathcal{L} / \nabla_{b_c} \mathcal{L}$
11:         if $f$ not in visited then
12:             visited add node $f$
13:             $G[f] = g \times n$
14:             $C_{\text{new}}$ append $f$
15:             $C_{\text{new}}$ append $2 \times c_{\text{cut}} - f$
16:         end if
17:         $C = C_{\text{new}}$
18:     end for
19: end while
20: $G$ ← values of $G$
21: $g_{\text{single}} = [G[0]]$
22: for $i$ in 1...len$(G)$ do
23:     $g_{\text{single}}$ append $G[i] - G[i-1]$
24: end for
25: return $g_{\text{single}}$
```

4. Experiments

In this section, we evaluate the empirical threat of these attacks. We show that fishing attacks can breach the privacy even for user aggregates with large batch sizes. We also provide quantitative comparisons with prior work.

4.1. Experimentation details

For our experiments, we focus on image classification as the central task. All images are from ImageNet ILSVRC 2012 (Russakovsky et al., 2015) with a size of 224×224 and include 1000 classes in total. We use $\alpha = 1000$ for the class fishing strategy and $\theta = 1000$ for the feature fishing strategy, and these hyperparameters are large enough to suppress the ε bound on the "unwanted" gradients to be negligibly small in practice. We apply both strategies to the last linear layer of a pre-trained ResNet-18 for all experiments except Section 4.4. We also use the optimization method of Geiping et al. (2020) to recover the image from the gradient update caught by the attack. In the optimization, we use Adam with step size 0.1 and 50 iterations of warmup over total 24K (Yin et al., 2021). The initialization is set to the pattern tiling of 4×4 random normal data introduced in (Wei et al., 2020). For regularization, we use a double-opponent, isotropic vectorial total variation (Åström & Schnörr, 2016).

4.2. One-shot feature attack

For the one-shot feature attack employed in the cross-device setting, we focus on investigating how the number of users a malicious server estimates influences the probability of retrieving a separate data point. We test our estimation at several levels of number of users ranging from 50 to 225 (this range is chosen for simplicity, as discussed in Section 3.2 it not an upper bound on the number of users). For each user, we consider 10 images in total, and 4 images from the target class. After the feature estimation, we then run the attack on 100 other, separate users also with batch size 10, with images all sampled from unseen images within the target class. For our experiments, we declare that the attacker recovers n data points $\{x_i\}_{i=1}^n$ if

$$\text{Sim}(g_c, \sum_{i=1}^{n} \nabla_\phi \mathcal{L}(x_i, c; \phi, W, b)) \geq 0.95,$$

Table 1. Probability to recover n data points for various method of choosing the attacked feature entry. Recovery at $n = 1$ is an immediate breach in privacy.

Method	# of Data Points Caught			
	0	1	2	≥ 3
KS-Test (Ours)	0.35	**0.39**	0.18	0.08
Highest KS-Stat	1.0	.0	.0	.0
Lowest Mean	1.0	.0	.0	.0
Highest Mean	0.26	0.37	0.25	0.12
Lowest STD	1.0	.0	.0	.0
Highest STD	0.39	0.31	0.22	0.08
Random	0.7	0.19	0.09	0.02

where g_c is the gradient update and Sim is the cosine similarity. Predictably, we find in Figure 4(a) that a server better estimates the feature distribution of interest with a larger number of users. As stated in Proposition 3.2, if the server accurately estimates the distribution of the chosen feature, they can expect to recover a user image $\frac{1}{e} \approx 37\%$ of the time!

Further, we include an ablation study regarding the importance of the KS-test in Tab. 1. We test out other heuristic methods to choose the feature entry, like the feature entry with highest standard deviation. Choosing the most normal feature is a helpful heuristic to improve attack success. However, the KS-test is not crucial for the attack to succeed, even attacking a random feature still recovers private data.

4.3. Binary attack

We also verify the stability of the one-shot binary attack employed in the cross-silo setting experimentally and evaluate how many queries a malicious server needs to breach the privacy of a user. For different batch sizes, we sample 50 different users who each own only data points with the same label (so that the maximal number of label collisions occurs) and observe the number of queries for the one-shot binary attack. We do indeed observe in Figure 5(a) that the number of necessary queries scales as $\log(n)$. For example, this allows an attacker to retrieve an individual data point from an aggregation of 256 images after, on average, 8 queries. Given that applications, e.g. in McMahan et al. (2018), run over 100 rounds of FL training, this is well within the range of possibility (Over 100 rounds, the potential aggregation size that could be breached is $2^{100} \sim 1e30$ data points).

In all cases, the underlying attack we employ is the optimization recovery method of Geiping et al. (2020). We do this for the update retrieved by the one-shot binary attack with different batch sizes on groups of users generated for ImageNet data by random partitions. We plot Max-RPSNR and Max-SSIM in Figure 5(b), which represent the maximum registered PSNR and maximum SSIM value in the

Table 2. The numerical results between previous work (Geiping et al., 2020) and the one-shot binary attack.

Metric	Method	Batch Size	
		32	50
Max-RPSNR	Prev.	14.029	14.568
	Ours	**17.415**	**17.610**

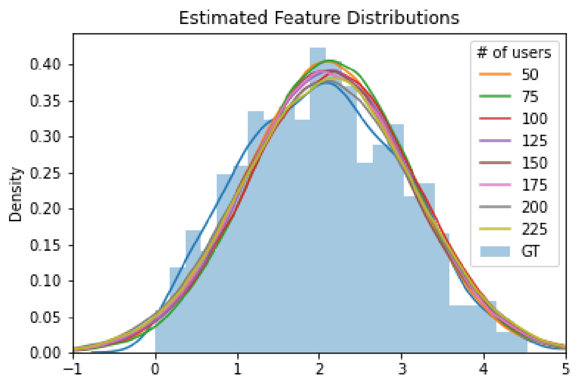
batch for each batch during the experiment. Table 2 also provides the results between the one-shot binary attack and original attack in Geiping et al. (2020). With the one-shot binary attack, we can see a noticeable improvement on the original attack. We note that with increasing batch size, this improvement actually increases further as the larger batches contain more vulnerable data points that are empirically easier to reconstruct.

4.4. Fishing with closed-form APRIL attacks

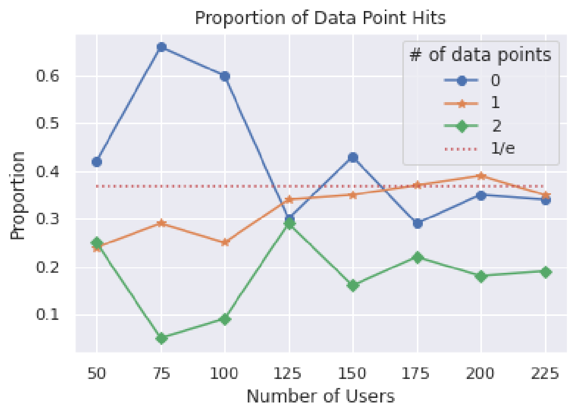
Because our proposed method is attack-agnostic (it can be combined with prior optimization or analytic attacks), we also reproduce the experiments from a recent closed-form attack, APRIL (Lu et al., 2021), which targets vision transformer models (Dosovitskiy et al., 2020). In APRIL, a malicious server is able to solve the closed-form solution of the self-attention layer to recover the patch embeddings, and then recover the original image by inverting the patch embedding layer. This method works exceedingly well for a single data point, but fails when the batch size is greater than 1, as then, patches from multiple images are mixed. We first modify a ViT-Small (Dosovitskiy et al., 2020) to build the setting described in Lu et al. (2021) by removing the Layer norm and the residual connection from the first block. Then, we run both attacks on 50 users for each batch size. Table 3 shows the results across different batch sizes on ImageNet. The original APRIL attack fails when the batch size reaches 2, further and further mixing patches until the recovered image is totally irrecoverable. However, the proposed fishing attack remains consistent with larger batch size, allowing this attack to scale to arbitrary batch sizes.

5. Defenses and Mitigation Strategies

Under the proposed threat model, users have few defenses against this attack, as further aggregation does not guarantee protection. However, other forms of protection, or at least identification, may be possible. The attack investigated in this work, and the attacks of Boenisch et al. (2021) and Pasquini et al. (2021) work by producing sparse gradients. This means that the update returned by the user contains parameters where their gradients are dominated (or entirely contributed) by single examples. In theory, a user could



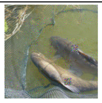
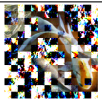
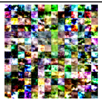
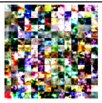
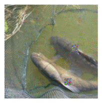
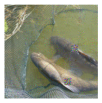
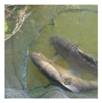
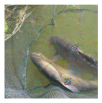
(a)



(b)

Figure 4. Influence of the number of users on the feature estimation. (a) The comparison of ground truth feature distribution and Gaussian fit to estimated feature distribution with different numbers of users. The more number of users a malicious server estimates the more estimated distribution closer to the ground true distribution. (b) The comparison of the proportions of getting n data points across various numbers of users. Here the server wants exactly 1 datapoint to fall across the selected cuts.

Table 3. Quantitative (Max-RPSNR) and qualitative comparison between original APRIL and APRIL + class fishing strategy on modified ViT-Small. The original APRIL attack fails when the batch size is barely 2, but fishing method with APRIL can still breach privacy at a batch size of 256.

Method	**Batch Size / Max-RPSNR**			
	1	2	64	256
APRIL	**23.257**	12.664	8.318	8.422
Ours	23.154	**21.047**	**21.187**	**21.481**
APRIL				
Fishing				

attempt to measure and detect this attack pattern (inspecting and storing all per-example gradients) and refuse a model update. However, this would also flag natural examples, as this pattern also appears in benign models (Boenisch et al., 2021). At first glance, a more promising direction could be to aggregate based on a per-parameter median instead of mean, yet it is still unclear if this is at all achievable in a secure aggregation framework where the median computation has to be online and facilitated without a central party. Another potential limitation of median aggregation is that there could be parameter modifications in which the targeted data point is returned as the median, breaching privacy again.

Overall, the best defense for a user of federated learning is local differential privacy (Bhowmick et al., 2019), which computes per-example gradient clipping and noise directly on the user side. A sufficient level of differential privacy breaks attacks against privacy, such as this one. However, model accuracy tend to diminish as privacy guarantees are increased (Jayaraman & Evans, 2019; Bagdasaryan et al., 2019).

6. Disparate Impact

We further want to highlight another concern that is unveiled by this attack. For the default threat model, privacy is non-uniform. The fishing attack generally targets outliers in feature space first. These outliers can correspond to underrepresented groups or different data domains in the dataset. For example, in the cross-silo setting, it will take on average $n\log(n)$ queries to recover the data point whose feature is at the median of all features in the update, whereas the outlier feature with the smallest value is recovered in only $\log(n)$ queries on average, as we also verify in Figure 5(a). Likewise the attack in the cross-device setting most easily fishes off the data point with the lowest feature value for each user/aggregation of users it attacks. To the best of our knowledge, this is the first time that disparate impact in privacy attacks has come up as a concern and want to highlight this for future investigations.

7. Conclusions

We investigate the threat model of malicious, or untrusted, server updates sent to user in federated learning, and find that this threat model allows attacker to leverage existing attacks to fish for separate data points in arbitrarily large batches. This requires no modification of the model, only

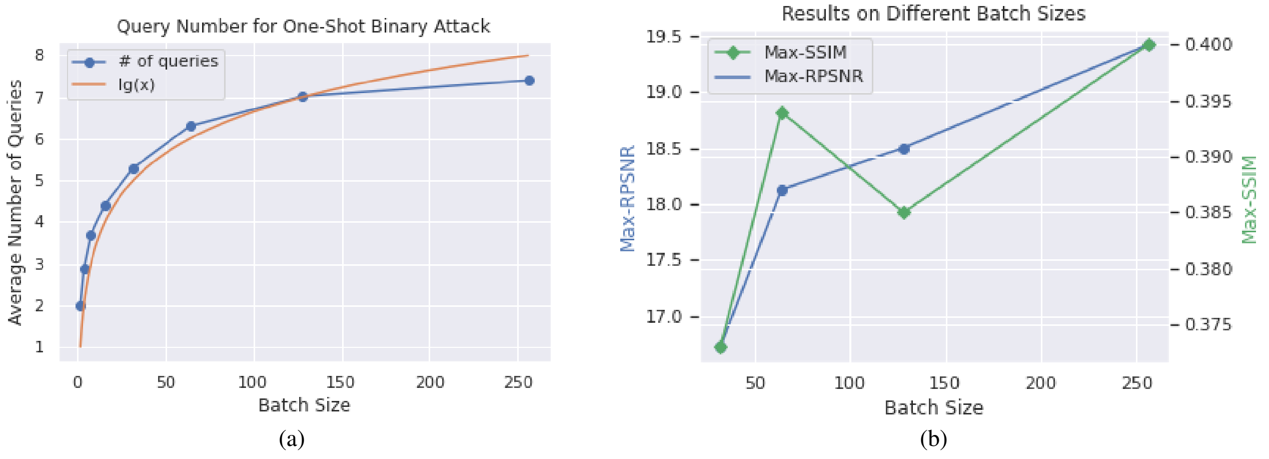


Figure 5. The performance of the one-shot binary attack. (a) The average number of queries required to breach one data point in various batch sizes for the one-shot binary attack. Each data point is the average over 50 users with separate labels. (b) Quantitative results leveraging the attack of Geiping et al. (2020) on large ImageNet batches.

of the feature of the last layer, and can break user privacy in both cross-device and cross-silo scenarios. A defense utilizing aggregation alone is not sufficient to protect user privacy against these attacks.

8. Acknowledgements

This work was support by DARPA's GARD program (HR00112020007), the AFOSR MURI program, and the National Science Foundation (DMS-1912866).

References

Åström, F. and Schnörr, C. Double-Opponent Vectorial Total Variation. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pp. 644–659, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6. doi: 10.1007/978-3-319-46475-6_40.

Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. Differential Privacy Has Disparate Impact on Model Accuracy. In *Advances in Neural Information Processing Systems 32*, pp. 15479–15488. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9681-differential-privacy-has-disparate-impact-on-model-accuracy.pdf.

Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., and Rogers, R. Protection Against Reconstruction and Its Applications in Private Federated Learning. *arXiv:1812.00984 [cs, stat]*, June 2019. URL http://arxiv.org/abs/1812.00984.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. When the Curious Abandon Honesty: Federated Learning Is Not Private. *arXiv:2112.02918 [cs]*, December 2021. URL http://arxiv.org/abs/2112.02918.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical Secure Aggregation for Privacy Preserving Machine Learning. Technical Report 281, 2017. URL http://eprint.iacr.org/2017/281.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, H. B., Van Overveldt, T., Petrou, D., Ramage, D., and Roselander, J. Towards Federated Learning at Scale: System Design. *arXiv:1902.01046 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1902.01046.

Bonawitz, K., Kairouz, P., McMahan, B., and Ramage, D. Federated Learning and Privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue*, 19(5):Pages 40:87–Pages 40:114, October 2021. ISSN 1542-7730. doi: 10.1145/3494834.3500240.

Cotter, F. *Uses of Complex Wavelets in Deep Convolutional Neural Networks*. Thesis, University of Cambridge, June 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fowl, L., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models. *arXiv:2110.13057 [cs]*, October 2021. URL http://arxiv.org/abs/2110.13057.

Frey, S. Introducing Android's Private Compute Services, September 2021. URL https://security.googleblog.com/2021/09/introducing-androids-private-compute.html.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting Gradients - How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, December 2020. URL https://proceedings.neurips.cc//paper_files/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. URL http://arxiv.org/abs/1512.03385.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. Evaluating Gradient Inversion Attacks and Defenses in Federated Learning. In *Advances in Neural Information Processing Systems*, May 2021. URL https://openreview.net/forum?id=0CDKgyYaxC8.

Jayaraman, B. and Evans, D. Evaluating Differentially Private Machine Learning in Practice. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1895–1912, 2019. ISBN 978-1-939133-06-9. URL https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman.

Jochems, A., Deist, T. M., van Soest, J., Eble, M., Bulens, P., Coucke, P., Dries, W., Lambin, P., and Dekker, A. 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.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]*, March 2021. URL http://arxiv.org/abs/1912.04977.

Kingsbury, N. Complex Wavelets for Shift Invariant Analysis and Filtering of Signals. *Applied and Computational Harmonic Analysis*, 10(3):234–253, May 2001. ISSN 1063-5203. doi: 10.1006/acha.2000.0343.

Lam, M., Wei, G.-Y., Brooks, D., Reddi, V., and Mitzenmacher, M. Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix. In *ICML*, 2021.

Lu, J., Zhang, X. S., Zhao, T., He, X., and Cheng, J. APRIL: Finding the Achilles' Heel on Privacy for Vision Transformers. *arXiv:2112.14087 [cs]*, December 2021. URL http://arxiv.org/abs/2112.14087.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv:1602.05629 [cs]*, February 2017. URL http://arxiv.org/abs/1602.05629.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning Differentially Private Recurrent Language Models. *arXiv:1710.06963 [cs]*, February 2018. URL http://arxiv.org/abs/1710.06963.

Pasquini, D., Francati, D., and Ateniese, G. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. *arXiv:2111.07380 [cs]*, December 2021. URL http://arxiv.org/abs/2111.07380.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*, Long Beach, CA, 2017. URL https://openreview.net/forum?id=BJJsrmfCZ.

Phong, L. T., Aono, Y., Hayashi, T., Wang, L., and Moriai, S. Privacy-Preserving Deep Learning: Revisited and Enhanced. In *Applications and Techniques in Information Security*, Communications in Computer and Information Science, pp. 100–110, Singapore, 2017. Springer. ISBN 978-981-10-5421-1. doi: 10.1007/978-981-10-5421-1_9.

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., and Bradski, G. Kornia: An Open Source Differentiable Computer Vision Library for PyTorch. *arXiv:1910.02190 [cs]*, October 2019. URL http://arxiv.org/abs/1910.02190.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. 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.

Truong, N., Sun, K., Wang, S., Guitton, F., and Guo, Y. Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Computers & Security*, 110:102402, November 2021. ISSN 0167-4048. doi: 10.1016/j.cose.2021.102402.

Wainakh, A., Ventola, F., Müßig, T., Keim, J., Cordero, C. G., Zimmer, E., Grube, T., Kersting, K., and Mühlhäuser, M. User Label Leakage from Gradients in Federated Learning. *arXiv:2105.09369 [cs]*, June 2021. URL http://arxiv.org/abs/2105.09369.

Wang, Z. and Simoncelli, E. Translation insensitive image similarity in complex wavelet domain. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 2, pp. ii/573–ii/576 Vol. 2, March 2005. doi: 10.1109/ICASSP.2005.1415469.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. *arXiv:1812.00535 [cs]*, December 2018. URL http://arxiv.org/abs/1812.00535.

Wei, W., Liu, L., Loper, M., Chow, K.-H., Gursoy, M. E., Truex, S., and Wu, Y. A Framework for Evaluating Gradient Leakage Attacks in Federated Learning. *arXiv:2004.10397 [cs, stat]*, April 2020. URL http://arxiv.org/abs/2004.10397.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. 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. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yin_See_Through_Gradients_Image_Batch_Recovery_via_GradInversion_CVPR_2021_paper.html.

Zhu, J. and Blaschko, M. R-GAP: Recursive Gradient Attack on Privacy. *arXiv:2010.07733 [cs]*, March 2021. URL http://arxiv.org/abs/2010.07733.

Zhu, L., Liu, Z., and Han, S. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems 32*, pp. 14774–14784. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9617-deep-leakage-from-gradients.pdf.

A. Proofs.

Proposition. **(Proposition 3.1)** Let ϕ denote the parameters of the feature extractor for the central model, and let W, b be the weight and bias vectors, respectively, of the last classification layer. Furthermore, let $\{x_i\}_{i=1}^{N}$ be the collection of user data on which model updates can be calculated. Let $\{x_i\}_{i=1}^{m \leq N}$ be any batch data containing k data points $\{x'_i\}_{i=1}^{k \leq m}$ from the target class. Then, for cross-entropy loss $\mathcal{L}$, if the gradient of $\mathcal{L}$ w.r.t. each data point in the batch is bounded in l_2 norm, then $\forall \varepsilon > 0$, $\exists \alpha$ so that a malicious server employing our class strategy on W, b recovers a feature extractor update g so that

$$\|g - \frac{1}{m}\sum_{i=1}^{k} \nabla_\phi \mathcal{L}(x'_i, c; \phi, W, b)\|_2 < \varepsilon.$$

Proof. Let $\{f_i\}_{i=1}^{m} = \{\phi(x_i)\}_{i=1}^{m}$ be the features of the batch of data used for update g. Similarly, let $\{p_i\}_{i=1}^{m}$ be the collection of softmax outputs for these features (i.e. $p_j = \sigma(W f_j + b) = \sigma(l_j)$ for σ the softmax function, l_j the corresponding logits). We want to show that the gradient contribution for the images coming from the non-target classes is minimal. To see this, WLOG, assume the first k images from the batch are images from the targeted class. Then, we separate the model update as:

$$g = \frac{1}{m}\left(\sum_{i=1}^{k} \nabla_\phi \mathcal{L}(x_i, c; \phi, W, b) + \underbrace{\sum_{j=k+1}^{m} \nabla_\phi \mathcal{L}(x_j, y_j; \phi, W, b)}_{(*)}\right)$$

We now equivalently have to show that $\frac{1}{m}\|(*)\|_2 \leq \varepsilon$. To do this, we take $x_* = \arg\max_{x \in \{x_j\}_{j=k+1}^{m}} \|\nabla_\phi \mathcal{L}(x_j, y_j; \phi, W, b)\|_2$. Now, it suffices to show that $\|\nabla_\phi \mathcal{L}(x_*, y_*; \phi, W, b)\|_2 < \varepsilon$. For cross-entropy loss, the only softmax prediction used in the loss corresponds to the ground-truth label. So for logits $\{l_i\}$, so label space of size s, we have:

$$L(x_*, y_*; \phi, W, b) = -\log(p_{*,y_*}) = -\log\left(\frac{e^{l_{*,y_*}}}{\sum_{i=1}^{s} e^{l_{*,i}}}\right)$$

for any index $s \neq c$ for the logits l_*, by construction of the matrix W, we have $\nabla_{\phi(x_*)} l_{*,s} = W_s = \vec{0}$. Therefore, when determining contributions to the gradient vector for ϕ, we need only account for gradient contributions from the target class' logit. From here, a straightforward calculation shows that.

$$\frac{\partial \mathcal{L}}{\partial l_{*,c}} = \frac{\partial \mathcal{L}}{\partial p_{*,y_*}} \frac{\partial p_{*,y_*}}{\partial l_{*,c}} = \frac{-1}{p_{*,y_*}} \cdot \frac{\partial p_{*,y_*}}{\partial l_{*,c}} = \underbrace{\frac{-1}{p_{*,y_*}} \cdot \frac{\partial}{\partial l_{*,c}}\left(\frac{e^{l_{*,y_*}}}{\sum_{i=1}^{s} e^{l_{*,i}}}\right)}_{(\#)}$$

Now there are two cases for $(\#)$:

Case 1: $y_* \neq c$:

$$\frac{-1}{p_{*,y_*}} \cdot \frac{\partial}{\partial l_{*,c}}\left(\frac{e^{l_{*,y_*}}}{\sum_{i=1}^{s} e^{l_{*,i}}}\right) = \frac{-1}{p_{*,y_*}} \cdot \frac{-e^{l_{*,y_*}} \cdot e^{l_{*,c}}}{\left(\sum_{i=1}^{s} e^{l_{*,i}}\right)^2} = \frac{1}{p_{*,y_*}} \cdot p_{*,y_*} \cdot p_{*,c} = p_{*,c}$$

Case 2: $y_* = c$:

$$\frac{-1}{p_{*,c}} \cdot \frac{\partial}{\partial l_{*,c}}\left(\frac{e^{l_{*,y_*}}}{\sum_{i=1}^{s} e^{l_{*,i}}}\right) = \frac{-1}{p_{*,c}} \cdot \frac{e^{l_{*,c}} \cdot \left(\sum_{i=1}^{s} e^{l_{*,i}}\right) - e^{2 \cdot l_{*,c}}}{\left(\sum_{i=1}^{s} e^{l_{*,i}}\right)^2} = \frac{-1}{p_{*,c}} \cdot p_{*,c} \cdot (1 - p_{*,c}) = p_{*,c} - 1$$

We focus on Case 1 since this determines the gradient magnitude for $(*)$. For this case, we notice that:

$$p_{*,c} = \frac{e^{l_{*,c}}}{\sum_{i=1}^{s} e^{l_{*,s}}} \leq \frac{e^{l_{*,c}}}{(s-1)\cdot e^{\alpha}}$$

so if the server sets $\alpha = \gamma \cdot \max_i \max_s f_{i,s}$ - i.e. the max logit over the entire batch - for some scale factor γ, this simplifies to $p_{*,c} \leq \frac{1}{(s-1)\cdot e^{(\gamma-1)l_{*,c}}}$. Putting this all together, we have:

$$\|\nabla_\phi \mathcal{L}(x_*, y_*; \phi, W, b)\|_2 = \|\frac{\partial \mathcal{L}}{\partial l_{*,c}} \cdot \nabla_\phi l_{*,c}\|_2 \leq \|\frac{1}{(s-1)\cdot e^{(\gamma-1)l_{*,c}}} \cdot \nabla_\phi l_{*,c}\|_2$$

By straightforward manipulation from this point, it is clear that the server can scale γ, and therefore α appropriately to ensure that

$$\|\nabla_\phi \mathcal{L}(x_*, y_*; \phi, W, b)\|_2 \leq \epsilon$$

which completes the proof.

Note that the derivation for Case 2 is not necessary for the statement of the proposition, however it does reveal that the gradients for images coming from the target class are not suppressed in magnitude in the same way that the non-target class gradients are, and, in fact, we see that for any image x_i from the target class, and any image x_j not from the target class:

$$\lim_{\alpha\to\infty} \left| \frac{\frac{\partial \mathcal{L}}{\partial l_{i,c}}}{\frac{\partial \mathcal{L}}{\partial l_{j,c}}} \right| = \lim_{\alpha\to\infty} \left| \frac{p_{i,c}-1}{p_{j,c}} \right| = \infty$$

□

Proposition. **(Proposition 3.2)** If the server knows the CDF (assumed to be continuous) for the distribution of any feature of interest for the target class, then for any batch of data $\{x_i\}_{i=1}^N$ with $\{x_{i_j}\}_{j=1}^{M\leq N}$ images coming from the target class, if the gradient of $\mathcal{L}$ w.r.t. each data point in the batch is bounded in l_2 norm (where $\mathcal{L}$ is the cross entropy loss), then we have that for the proposed feature fishing strategy, with probability $p \geq \frac{1}{e}$, there exists a data point $x_* \in \{x_{i_j}\}$ so that

$$\|g_c - \frac{1}{N}\nabla_\phi \mathcal{L}(x_*, c; \phi, W, b)\|_2 \leq \varepsilon$$

where g_c is the component of the model update coming from data from the target class.

Proof. If the server knows the CDF for some feature of interest for the target class, then the server can create "bins" of equal mass corresponding to mass $\frac{1}{M}$. Let X be the random variable corresponding to the number of data points from the target class in our batch that fall into the chosen bin. Then, we have:

$$p(X=1) = \binom{M}{1} \cdot \frac{1}{M} \cdot \left(1 - \frac{1}{M}\right)^{M-1} = \left(1 - \frac{1}{M}\right)^{M-1} \geq \frac{1}{e}$$

Now, if we have that one and only one image, x_*, from the target class falls into this bin, then, by construction of our feature fishing attack, for any $\varepsilon_1, \varepsilon_2 > 0$, we can choose parameter β so that for the "fished" image x_*, we have that $p_{*,c} < \varepsilon_1$ and for any image $x_k \in \{x_{i_j}\}$ where $x_k \neq x_*$, then $p_{k,c} > 1 - \varepsilon_2$. We can see this via the calculation in Section 3.2 where the logits of the fished and non-fished logits have oposite signs and depend linearly on the server parameter β.

Then, using a similar calculation to the proof for Proposition 3.1, we have that

$$\|g_c - \frac{1}{N}\nabla_\phi \mathcal{L}(x_*, c; \phi, W, b)\|_2 = \|\frac{1}{N} \sum_{\substack{x_k \in \{x_{i_j}\},\\ x_k \neq x_*}} \nabla_\phi \mathcal{L}(x_k, c; \phi, W, b)\|_2 \leq \max_{\substack{x_k \in \{x_{i_j}\},\\ x_k \neq x_*}} \|\nabla_\phi \mathcal{L}(x_k, c; \phi, W, b)\|_2$$

because the feature fishing attack is combined with the class fishing attack, we have the following:

$$= \max_{\substack{x_k \in \{x_{i_j}\}, \\ x_k \neq x_*}} \| \frac{\partial \mathcal{L}}{\partial l_{k,c}} \cdot \nabla_\phi l_{k,c} \|_2 \leq |(p_{k,c} - 1) \cdot C| \leq \varepsilon$$

for C a constant. The last inequality occurs because $p_{k,c}$ can be made arbitrarily close to 1.

Note again that we did not use the fact that $p_{*,c}$ can be made arbitrarily close to 0, but this is important for the empirical success of the attack as it yields a high magnitude gradient for x_* which improves reconstruction.

$\square$

B. Technical Details

We implement these attacks in a `PyTorch` framework (Paszke et al., 2017) which contains all material to replicate these experiments and which we include in the supplementary material. We run all the optimization-based attacks using single `2080ti` GPUs and run the analytic attacks via APRIL on CPUs, solving the embedding layer and attention inversion under-determined problems via an SVD solver (`dgelss`). For our quantitative experiments we partition the ImageNet validation set into 100 users with the given batch size, and either allocate each user a different class or assign images to users at random (without replacement). For the optimization-based attacks, we follow the experimental protocol of Geiping et al. (2020), in which batch norm layers are fixed in evaluation mode and populated from a pretrained model. In this scenario, the user does not have to send updated batch norm statistics to the server. We further assume, for simplicity, that label information is contained in the update as metadata, but note that novel label recovery algorithms as discussed in Wainakh et al. (2021) are highly successful, even at large aggregation sizes.

We report R-PSNR and SSIM metrics in this work. For R-PSNR (registered-PSNR), as also investigated in Yin et al. (2021), we register the reconstructed image to the ground truth reference data by direction optimization using `kornia` (Riba et al., 2019). For SSIM we also compute a translation-invariant SSIM score by implementinging complex-wavelet SSIM as originally proposed in Wang & Simoncelli (2005). As underlying wavelet transform we use a dual-tree complex wavelet transform over 5 scales (Kingsbury, 2001; Cotter, 2020) with bi-orthogonal wavelet filters.

C. Additional Results

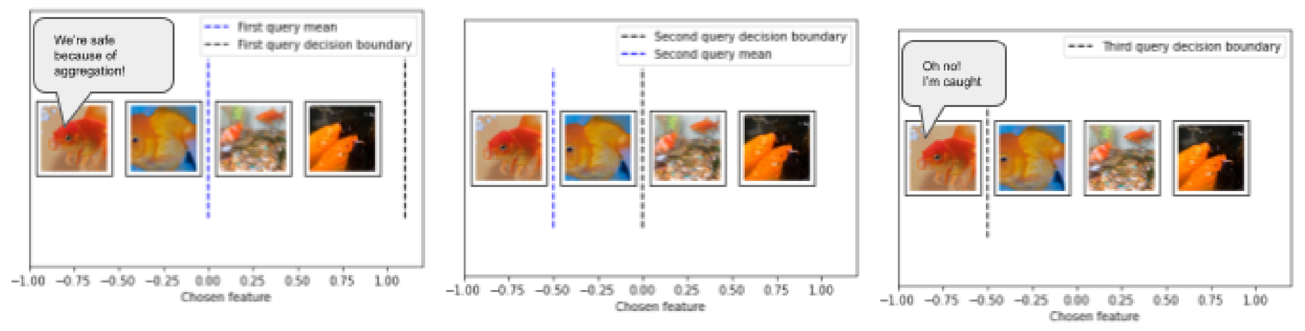
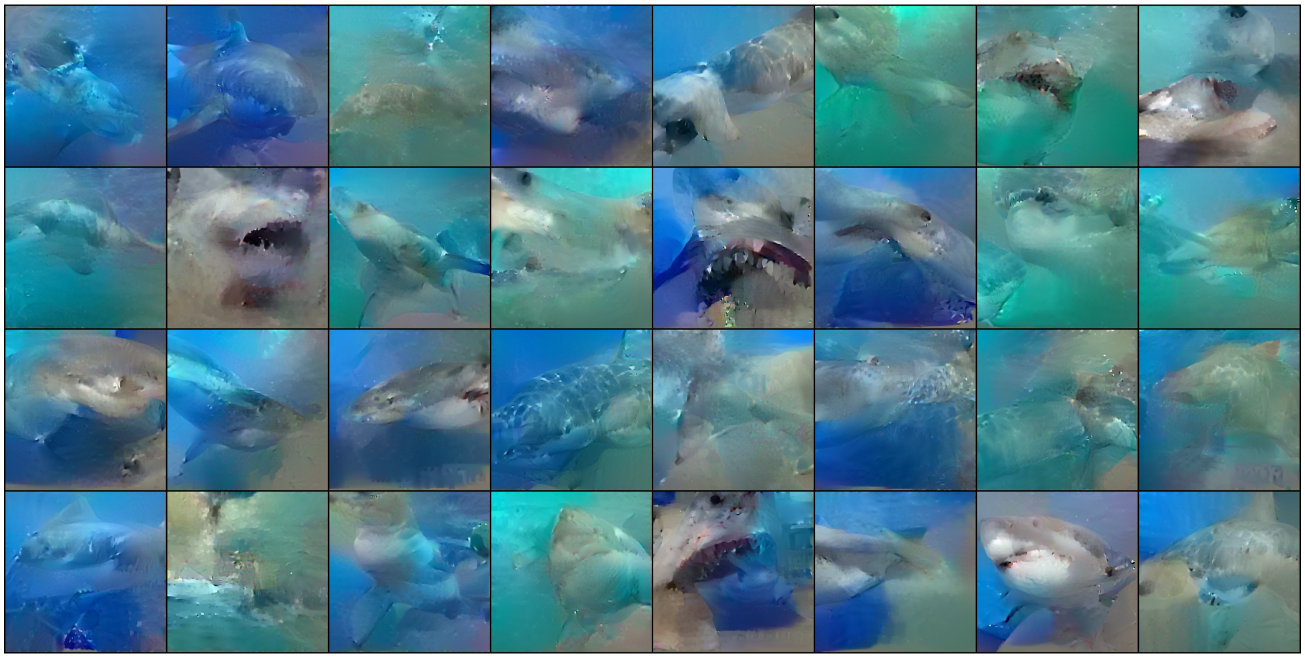
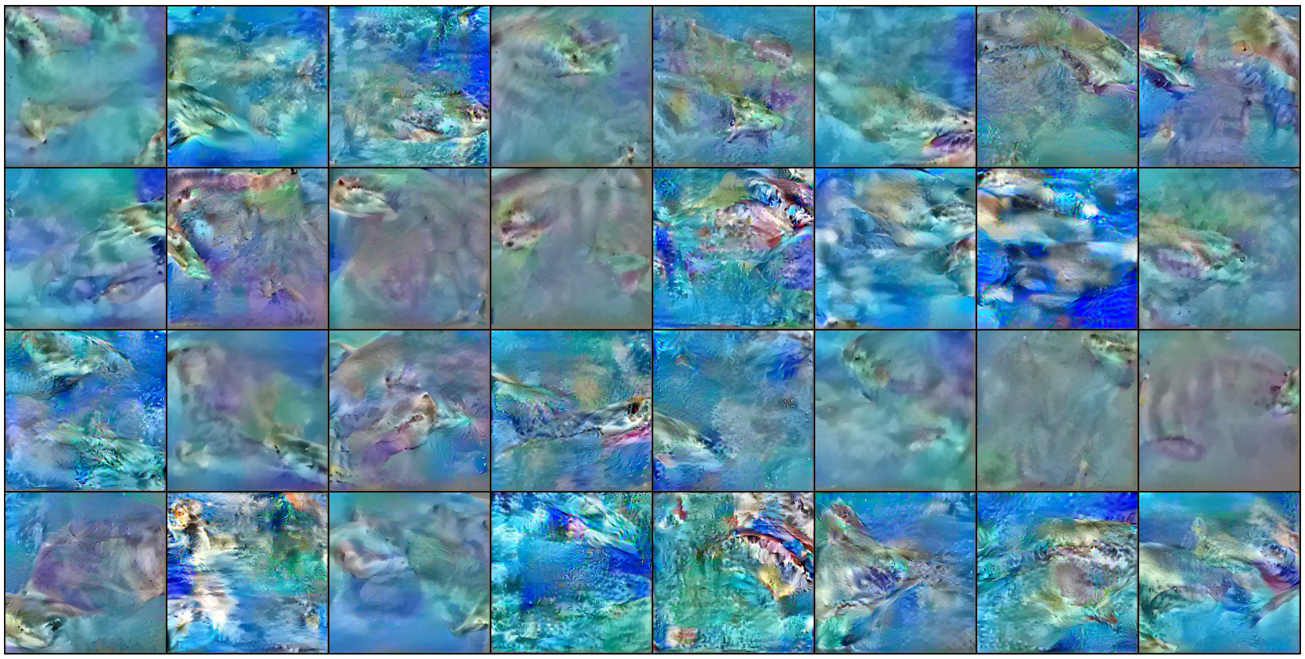

We're safe because of aggregation!
First query mean
First query decision boundary
Second query decision boundary
Second query mean
Oh no! I'm caught
Third query decision boundary
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
Chosen feature

Figure 6. A high-level schematic of our feature fishing strategy. Within a class, “Goldfish”, features can be iteratively suppressed to reveal a single user image.



(a)



(b)

Figure 7. The recovered images from the batch containing class ”shark” only. (a) With the class attack only without feature fishing. (b) Unmodified model.

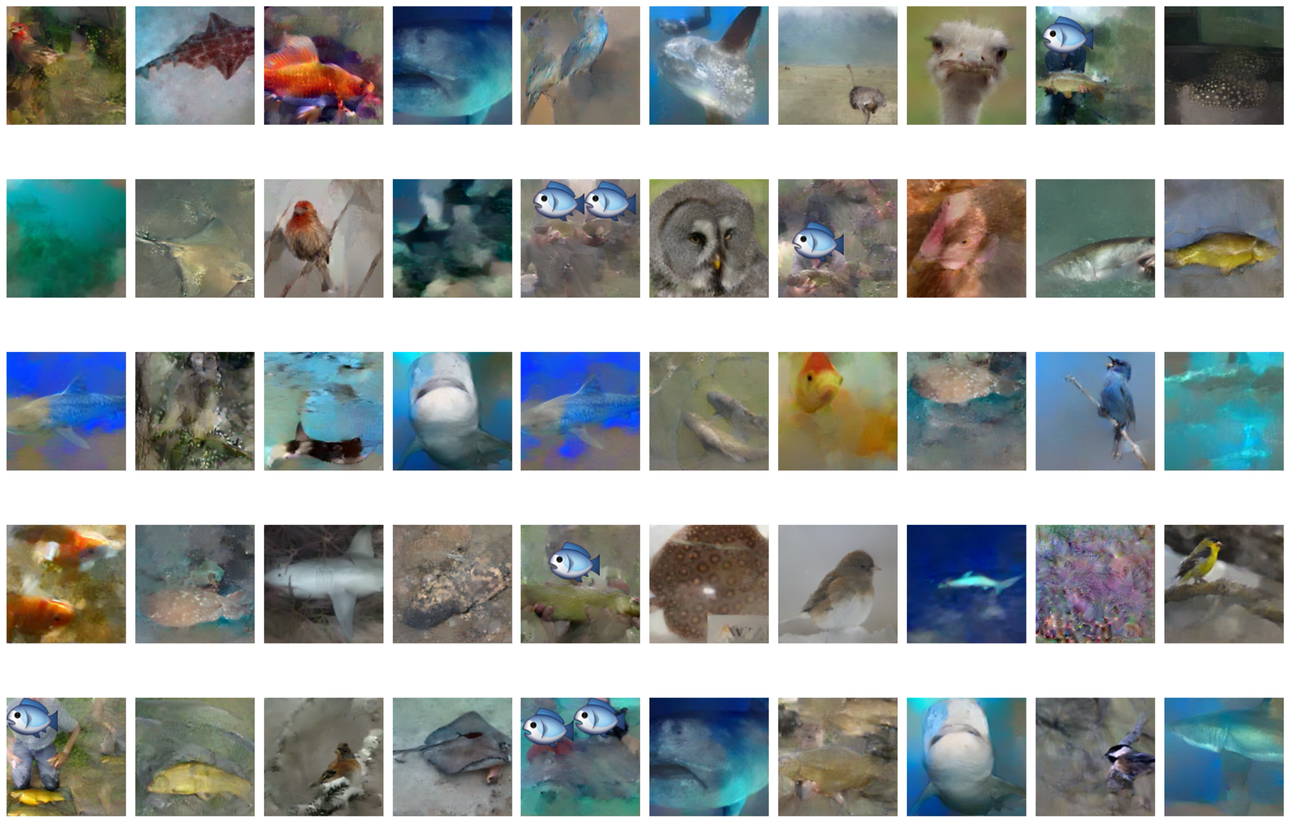


Figure 8. The recovered images from 50 different users with batch size 128. We add a fish emoji to human faces from the ImageNet dataset for privacy reasons. All recoveries are at least as successful as optimization-based recoveries from a single data point.

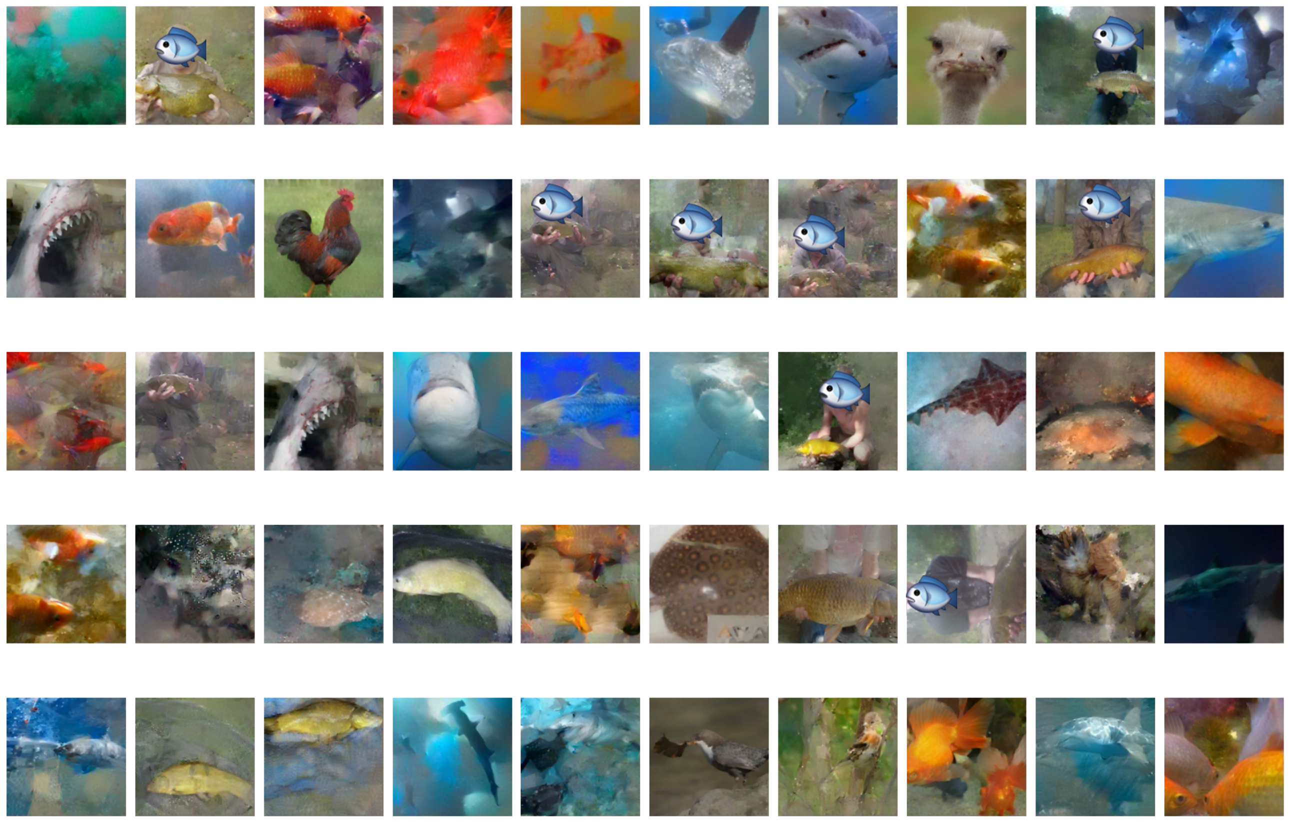


Figure 9. The recovered images from 50 different users with batch size 256. We add a fish emoji to human faces from the ImageNet dataset for privacy reasons. All recoveries are at least as successful as optimization-based recoveries from a single data point.