Gradient Inversion with Generative Image Prior

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated Learning (FL) is a distributed learning framework, in which the data never leaves clients' devices to preserve privacy, and the server trains models on the data via accessing only the gradients of those local data. Without further privacy mechanisms such as differential privacy, this leaves the system vulnerable against an attacker who inverts those gradients to reveal clients' sensitive data. However, a gradient is often insufficient to reconstruct the user data without any prior knowledge. By exploiting a generative model pretrained on the data distribution, we demonstrate that data privacy can be easily breached. However, such prior knowledge is often unavailable or inappropriate. We hence investigate the possibility of learning the prior from a sequence of gradients seen in the process of FL. We experimentally show that the prior in a form of generative model is learnable from iterative interactions in FL. Our finding strongly suggests using additional mechanisms to prevent privacy leakage in FL.

1 Introduction

Federated learning (FL) is an emerging framework for distributed learning, where central server aggregates model updates, rather than user data, from end users [5, 15]. The main premise of federated learning is that this particular way of distributed learning can protect users' data privacy as there is no explicit data shared by the end users with the central server.

However, recent works [30, 27, 9, 25] demonstrated that one may recover the private user data used for training by observing the gradient. This process of recovering the training data from gradient(s), so-called *gradient inversion*, poses a huge threat to the federated learning community as it may imply the fundamental flaw of its main premise.

Even more worryingly, recent works suggest that such gradient inversion attacks can be made even stronger if certain side-information is available. For instance, in [9], the authors showed that if the attacker knows a prior that user data consists of natural images, then the gradient inversion attack can leverage such prior, achieving a more accurate recovery of the user data. Another instance is when batch norm statistics are available at the attacker in addition to gradients. This can actually happen if the end users share their local batch norm statistics as in [15]. Yin et al. [25] showed that such batch normalization statistics can significantly improve the strength of the gradient inversion attack, enabling precise recovery of high-resolution images.

In this work, we systematically study how one can maximally utilize and even obtain the prior information when inverting gradients. We first consider the case that the attacker has a generative model pretrained on the exact or approximate distribution of the user data as a prior. For this, we propose an efficient gradient inversion algorithm that utilizes the generative model prior. In particular, the algorithm consists of two steps, in which the first step searches the latent space (of lower dimension) defined by the generative model instead of the ambient input space (of higher dimension), and then the second step adapts the generative model to each input given the gradient.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Each step provides substantial improvement in the reconstruction. We dub the algorithm as gradient inversion in alternative spaces (GIAS).

We then consider a realistic scenario in which the user data distribution is not known in advance, and thus the attacker needs to learn it from gradients. For this scenario, we develop a meta-learning framework, called gradient inversion to meta-learn (GIML), which learns a generative model on user data from observing and inverting multiple gradients computed on the data, e.g. across different FL epochs or participating nodes. Our experimental results demonstrate that one can learn a generative model via GIML and reconstruct data from the gradients by making use of the learned generative model.

This implies a great threat on privacy leakage in FL since our methods can be applied for any data type in most FL scenarios unless a specialized architecture prevents the gradient leakage explicitly, e.g., [16].

Our main contributions are as follows:

- We introduce GIAS that fully utilizes a pretrained generative model to invert gradient. In addition, we propose GIML which can train generative model from gradients only in FL.
- We demonstrate significant privacy leakage using GIAS with pretrained generative model in various challenging scenarios, where our method provides substantial gain additionally to other any other existing methods [9, 25].
- We experimentally show that GIML can learn a generative model on the user data from only gradients, which provides the same level of data recovery with a given pretrained model. To our best knowledge, GIML is the first capable of learning explicit prior on a set of gradient inversion tasks.

1.1 Related Work

Early works [17, 20] studied about information leakage from gradients such as image reconstruction or membership inference when FL learns a shallow network such as single-layer perceptron. Using optimization methods to minimize a certain measure of the discrepancy between true gradient and synthesized gradient from reconstructed data, the scenarios that the gradient inversion is possible have been extended to a large class of deep neural networks [18, 23, 30, 9, 25]. In parallel with those optimization-based approaches, Fan et al. [7] and Zhu and Blaschko [28] propose inversion methods solving a sequence of linear systems recursively. As non-convex optimization does not always converge to an optimal solution, solving linear systems have its own benefit.

The optimization-based approaches are particularly useful as one can simply utilize prior knowledge by adding regularization terms, or change discrepancy measure. Geiping et al. [9] demonstrate the gradient inversion for a batch of images on deep convolutional neural networks using cosine similarity for the discrepancy measure and adding total variation regularization as natural image prior. By exploiting BN statistics, Yin et al. [25] showed that the inversion is possible for mini-batch size 48, although BN statistics is not always given since FL can perform BN without exchanging BN statistics [14, 2]. However, we note that those advanced methods have no ability to learn the latent prior on data and consider to invert a gradient solely, while we show that by jointly considering gradient inversion, we can extract the latent prior.

1.2 Organization.

This paper is organized as follows. Section 2 introduces the formal definition. In Section 3, we devise our proposed methods to exploit and train generative model from inverting gradients. The experimental result is presented in Section 4. We conclude this paper with remarks on social impact and limitation of our work in Section 5.

2 Problem Formulation

In this section, we formally describe the gradient inversion (GI) problem. Consider a standard supervised learning for classification, which optimizes neural network model f_θ parameterized by θ

as follows:

$$\min_{\theta} \sum_{(x,y)\in\mathcal{D}} \ell(f_\theta(x), y) \ , \tag{1}$$

where ℓ is a point-wise loss function and $\mathcal{D}$ is a dataset of input $x \in \mathbb{R}^m$ and label $y \in \{0,1\}^L$ (one-hot vector). In federated learning framework, each node reports the gradient of $\ell(f_\theta(x), y)$ for sampled data (x, y)'s instead of directly transferring the data. The problem of inverting gradient is to reconstruct the sampled data used to compute the reported gradient. Specifically, when a node computes the gradient using a batch $\{(x_1^*, y_1^*), ..., (x_B^*, y_B^*)\}$, we consider the following problem of inverting gradient:

$$\min_{(x_1,y_1),...,(x_B,y_B)\in\mathbb{R}^m\times\{0,1\}^L} d\left(\frac{1}{B}\sum_{j=1}^{B}\nabla\ell(f_\theta(x_j), y_j), \frac{1}{B}\sum_{j=1}^{B}\nabla\ell(f_\theta(x_j^*), y_j^*)\right) \ , \tag{2}$$

where $d(\cdot,\cdot)$ is a measure of the discrepancy between two gradient, e.g., ℓ_2-distance [30, 25] or negative cosine similarity [9]. It is known that label y can be almost accurately recovered by simple methods just observing the gradient at the last layer [27, 25], while reconstructing input x remains still challenging as it is often under-determined even when the true label is given. For simplicity, we hence focus on the following minimization to reveal the inputs from the gradient given the true labels:

$$\min_{x_1,...,x_B\in\mathbb{R}^m} c\left(x_1, ..., x_B; \theta, g\right) \ , \tag{3}$$

where we denote by $g = \frac{1}{B}\sum_{j=1}^{B}\nabla\ell(f_\theta(x_j^*), y_j^*)$ the target gradient, and by $c\left(x_1, ..., x_B; \theta, g\right)$ the cost function in (2) given $y_j = y_j^*$ for each $j = 1, ..., B$.

3 Methods

The key challenge of inverting gradient is that solving (2) is often under-determined, i.e., a gradient contains only insufficient information to recover data. Such an under-determined issue is observed even when the dimension of gradient is much larger than that of input data, c.f., twin data [28]. To alleviate this issue, a set of prior knowledge on the nature of data can be considered.

When inverting images, Geiping et al. [9] proposed to add the total variation regularization $R_{\text{TV}}(x)$ for each image x to the cost function in (3) since neighboring pixels of natural images are likely to have similar values. More formally,

$$R_{\text{TV}}(x) := \sum_{(i,j)}\sum_{(i',j')\in\partial(i,j)} \|x(i,j) - x(i',j')\|^2 \ , \tag{4}$$

where $\partial(i,j)$ is the set of neighbors of (i,j). This method is limited to the natural image data.

For general type of data, one can consider exploiting the batch normalization (BN) statistics from nodes. This is available in the case that the server wants to utilize batch normalization (BN) in FL, and thus collects the BN statistics (mean and variance) of batch from each node, in addition, with every gradient report [15]. To be specific, Yin et al. [25] proposed to employ the regularizer $R_{\text{BN}}(x_1, ..., x_B; \theta)$ which quantifies the discrepancy between the BN statistics of estimated x_j's and those of true x_j^*'s on each layer of the learning model. More formally,

$$R_{\text{BN}}(x_1, ..., x_B; \theta) := \sum_{l} \|\mu_l - \mu_{l,\text{exact}}\|_2 + \|\sigma_l^2 - \sigma_{l,\text{exact}}^2\|_2 \ , \tag{5}$$

where $\mu_l(x_1, ..., x_B; \theta)$ and $\sigma_l^2(x_1, ..., x_B; \theta)$ (resp. $\mu_{l,\text{exact}}(x_1^*, ..., x_B^*; \theta)$ and $\sigma_{l,\text{exact}}^2(x_1^*, ..., x_B^*; \theta)$) are the mean and variance of l-th layer feature maps for the estimated batch $x_1, ..., x_B$ (resp. the true batch $x_1^*, ..., x_B^*$) given θ. This is available only if clients agreed to report their exact BN statistics at every round. But not every FL framework report BN statistics [14, 2]. In that case, Yin et al. [25] also proposed to use the BN statistics over the entire data distribution as a proxy of the true BN statistics, and reports that the gain from the approximated BN statistics is comparable to that from the exact ones. The applicability of R_{BN} with the approximated BN statistics is still limited as the proxy needs to be additionally recomputed over the entire data distribution at every change of θ. However, this demonstrates the significant impact of knowing the data distribution in the gradient inversion and motivates our methods using and learning a generative model on the user data, described in what follows.

3.1 Gradient Inversion with Pretrained Generative Model

Consider a decent generative model $G_w : \mathbb{R}^k \mapsto \mathbb{R}^m$ trained on the approximate (possibly exact) distribution of user data $\mathcal{D}$ such that $x^* \approx G_w(z^*)$ for $(x^*, \cdot) \in \mathcal{D}$ and its latent code $z^* = \arg\min_z \|G_w(z) - x^*\|$. To fully utilize such a pretrained generative model, we propose gradient inversion in alternative spaces (GIAS) described in Algorithm 1, which performs latent space search over z and then parameter space search over w, of which are elaborated in what follows.

Latent space search. Note that the latent space is typically much smaller than the ambient input space, i.e., $k \ll m$, for instances, DCGAN [21] of $k = 100$ and StyleGAN [11] of $k = 512 \times 16$ for image data of $m = (\text{width}) \times (\text{height}) \times (\text{color})$ such as $32 \times 32 \times 3$, $256 \times 256 \times 3$, or larger. Using such a pretrained generative model with $k \ll m$, the under-determined issues of (3) can be directly mitigated by narrowing down the searching space from $\mathbb{R}^m$ to $\{G_w(z) : z \in \mathbb{R}^k\}$. Hence, GIAS first performs the latent space search in the followings:

$$\min_{z_1,\ldots,z_B \in \mathbb{R}^k} c\left(G_w(z_1), ..., G_w(z_B)\right) \ . \tag{6}$$

Considering a canonical class of neural network model f_θ, we can show that the reconstruction of x^* by latent space search in (6) aligns with that by input space search in (3) if the generative model G_w approximates input data with small enough error.

Property 1 *For an input data $x^* \in \mathbb{R}^m$, consider the gradient inversion problem of minimizing cost c in* (3)*, where the learning model f_θ is a standard form of neural network (such as multi-layer perceptron or convolutional neural network) with C1 (continuously differentiable) activations (e.g., sigmoid and exponential linear), loss function ℓ is C1 (e.g., logistic and exponential), and the discrepancy measure d is ℓ_2-distance. Suppose that it has the unique global minimizer at x^*. Let $\varepsilon \geq 0$ be the approximation error bound on x^* for generative model $G_w : \mathbb{R}^k \mapsto \mathbb{R}^m$ with $k \leq m$, i.e., $\min_{z \in \mathbb{R}^k} \|x^* - G_w(z)\| \leq \varepsilon$. Then, there exists $\delta(\varepsilon) \geq 0$ such that for any $z^* \in \arg\min_z c(G_w(z))$,*

$$\|G_w(z^*) - x^*\| \leq \delta(\varepsilon) \ , \tag{7}$$

of which upper bound $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$.

A rigorous statement of Property 1 and its proof are provided in Appendix, where we prove and use that the cost function is continuous around x^* under the assumptions. This property justifies solving the latent space search in (6) for FL scenarios training neural network model while it requires an accurate generative model.

Parameter space search. Using the latent space search only, there can be inevitable reconstruction error due to the imperfection of generative model. This is mainly because we cannot perfectly prepare the generative model for every plausible data in advance. Similar difficulty of the latent space search has been reported even when inverting GAN [29, 3, 4] for plausible but new data directly, i.e., $\min_z \|G_w(z) - x^*\|$ given x^*, rather than inverting gradient. Bau et al. [3] propose an instance-specific model adaptation, which slightly adjusts the model parameter w to (a part of source image) x^* after obtaining a latent code z^* for x^*. Inspired by such an instance-specific adaptation, in GIAS, we perform the following parameter space search over w preceded by the latent space search over z:

$$\min_{w_1,\ldots,w_B} c\left(G_{w_1}(z_1), ..., G_{w_B}(z_B)\right) \ , \tag{8}$$

where $z_1, \ldots, z_B$ are obtained from (6).

Remark. We propose the optimization over w followed by that over z sequentially, as described in Algorithm 1. This is to maximally utilize the benefit of mitigating the under-determined issue from reducing the searching space on the pretrained model. However, the benefit would be degenerated or disappear if z and w are optimized jointly or w is optimized first. We provide an empirical justification on our method in Section 4.2 with Figure 1.

We perform each search in GIAS using a standard gradient method to the cost function directly. It is worth noting that those optimizations (6) and (8) with generative model can be tackled in a recursive manner as R-GAP [28] reconstructs each layer from output to input. We provide details and performance of the recursive procedure in Appendix, where employing generative model improves the inversion accuracy of R-GAP substantially, while R-GAP apparently suffers from an error accumulation issue when f_θ is deep neural network.

Algorithm 1 Gradient Inversion in Alternative Spaces (GIAS)

Require: learning model f_θ; target gradient $g = \nabla_\theta \sum_{j=1}^{B} \ell(f_\theta(x_j^*), y_j^*)$ to be inverted; batch size B; pre-trained generative model G_w;

1: Initialize $\boldsymbol{z} := (z_1, ..., z_B)$ randomly
2: Find $\boldsymbol{z} \in \arg\min_{\boldsymbol{z}} c(G_w(z_1), ..., G_w(z_B))$ // Latent space search
3: Set $\boldsymbol{w} := (w_1, \ldots, w_B) \leftarrow (w, \ldots, w)$
4: Find $\boldsymbol{w} \in \arg\min_{\boldsymbol{w}} c(G_{w_1}(z_1), \ldots, G_{w_B}(z_B))$ // Parameter space search
5: Return result: $G_{w_1'}(z_1), \ldots, G_{w_B'}(z_B)$

Algorithm 2 Gradient Inversion to Meta-Learn generative model (GIML)

Require: inversion task set $\mathcal{S}$; task batch size N; data batch size B (per gradient); number of local iterations τ; z-regularizer coefficient λ; step sizes α, β;

1: Initialize w randomly
2: **while** not done **do**
3: Sample a batch of inversion tasks $(\theta_1, g_1), ..., (\theta_N, g_N)$ from $\mathcal{S}$
4: $w' \leftarrow w$
5: **for all** $i = 1, \ldots, N$ **do**
6: $\boldsymbol{z}_i^* \leftarrow \text{argmin}_{\boldsymbol{z}_i} c(G_{w'}(z_{i1}), \ldots, G_{w'}(z_{iB}); \theta_i, g_i) + \lambda \sum_j \|z_{ij}\|_2$ // Regularized latent space searching
7: **end for**
8: **for all** $t = 1, \ldots, \tau$ **do**
9: $w' \leftarrow w' - \alpha \nabla_{w'} \sum_i c\left(G_{w'}(z_{i1}^*), \ldots, G_{w'}(z_{iB}^*); \theta_i, g_i\right)$ // Meta parameter space search
10: **end for**
11: Update $w \leftarrow w - \beta(w - w') = (1 - \beta)w + \beta w'$
12: **end while**

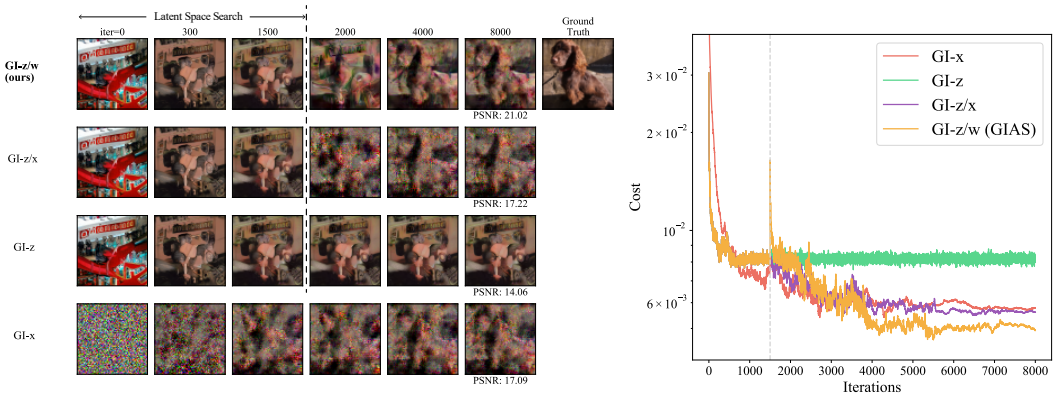
3.2 Gradient Inversion to Meta-Learn Generative Model

For the case that pretrained generative model is not available, we devise an algorithm to train a generative model G_w for a set $\mathcal{S} = \{(\theta_i, g_i)\}$ of gradient inversion tasks. Since each inversion task can be considered as a small learning task to adapt generative model per data, we hence call it Gradient Inversion to Meta-Learn generative model (GIML). The procedure of GIML is presented in Algorithm 2, and elaborated in what follows. We start with an arbitrary initialization of w, and iteratively updates toward w' from a variant of GIAS for N tasks sub-sampled from $\mathcal{S}$, which is different than multiple applications of GIAS for each task in two folds: (i) ℓ_2-regularization in latent space search; and (ii) an integrated optimization on model parameter. The variant first finds optimal latent codes $\boldsymbol{z}_i^* = (z_{i1}^*, ..., z_{iB}^*)$ for each task i with respect to the same cost function of GIAS but additional ℓ_2-regularization. Note that the latent space search with untrained generative model easily diverges. The ℓ_2-regularization can prevent the divergence of $\boldsymbol{z}_i^*$. Once we obtained $\boldsymbol{z}_i^*$'s, w' is computed by few steps of gradient descents for an integrated parameter search to minimize $\sum_i c(G_{w'}(z_{i1}^*), ..., G_{w'}(z_{iB}^*); \theta_i, g_i)$. This is because in GIML, we want meta information w to help individual GIAS for each task rather than solving the tasks perfectly, while after performing GIML to train w, we apply GIAS for individual task with the trained w. This shares the philosophy of model-agnostic meta learning in [8].

4 Experiments

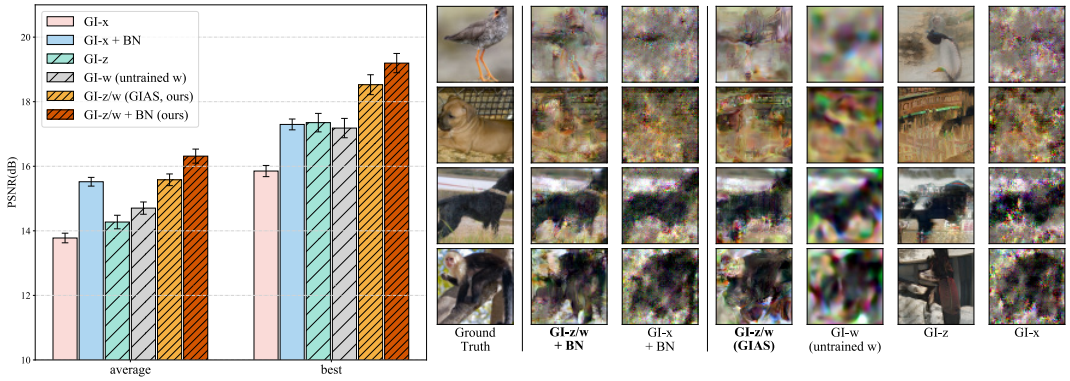
4.1 Experiment Settings

Unless stated otherwise, we consider the image classification task on the validation set of ImageNet [19] dataset resized to 64×64 using ResNet18 [10] as learning model. The resizing is necessary for computational tractability. Recalling the under-determined issue is the major challenge in gradient inversion, deeper and wider f_θ makes the gradient inversion easier [9]. Hence, the choice of ResNet18 as learning model is the most difficult setting within ResNet architectures since it contains the least information. We use a StyleGAN2[12] model trained on ImageNet for GIAS. For



(a) (b)

Figure 1: Comparison of different optimization targets after optimizing z. (a) All methods performed same latent space search up to 1500 iterations except for the last row(GI-x). (b) Optimizing w after z is the most effective way to reconstruct data from the gradient.



(a) (b)

Figure 2: Comparison of state-of-the-art models and ablation study. (a) Average PSNR and best PSNR in a batch throughout the experiments is shown with 95% confidence interval. (b) Comparison of reconstruction results with our models and state-of-the-art models. Our models are bolded.

StyleGAN2, latent space search was done in intermediate latent space, known as $\mathcal{W}$, as done in [1]. We used the batch size $B = 4$ as default, and negative cosine for the choice of gradient dissimilarity function $d(\cdot, \cdot)$, which apparently provides better inversion performance than ℓ_2-distance in general. For the optimization in GIAS, we used Adam optimizer [13] which decays learning rate by a factor of 0.1 at $3/8, 5/8, 7/8$ of total iterations. from initial learning rates $\eta_z = 3 \times 10^{-2}$ for the latent space search and $\eta_w = 10^{-3}$ for the parameter space search. Since our experiments are conducted with image data, we used total variation regularizer with weight $\lambda_{\text{TV}} = 10^{-4}$ for all experiments. For each inversion, we pick the best recovery among 4 random instances based on the final loss. All experiments are performed on GPU servers equipped with NVIDIA RTX 3090 GPU and NVIDIA RTX 2080 Ti GPU. Numerical results including graphs and table are averaged over 10 samples except Figure 1b and Figure 5.

Algorithms. We evaluate a list of algorithms for the gradient inversion (GI) task in (3), which are mainly distinguished by searching spaces of input x, latent code z, and model parameter w. Each algorithm is denoted by GI-$(\cdot)$, where the suffix indicates the searching space(s). For instances, GI-z/w is identical to the proposed method, GIAS, in which the latent cost searching preceded by the model parameter searching, and GI-x is the one proposed by Geiping et al. [9].

Table 1: Comparison of (our methods) with state-of-the-art methods. PSNR, SSIM, and LPIPS[26] were used for evaluate reconstruction results.

Method	GI-x [9]	GI-z (ours)	GI-w (ours)	GI-z/w (GIAS, ours)	GI-x+BN [25]	GI-z/w+BN (ours)
PSNR ↑	13.78	14.27	14.70	**15.58**	15.52	**16.31**
SSIM ↑	0.2542	0.3106	0.3519	**0.3895**	0.3513	**0.4311**
LPIPS ↓	0.4376	0.3233	0.5121	**0.3023**	0.3645	**0.2861**

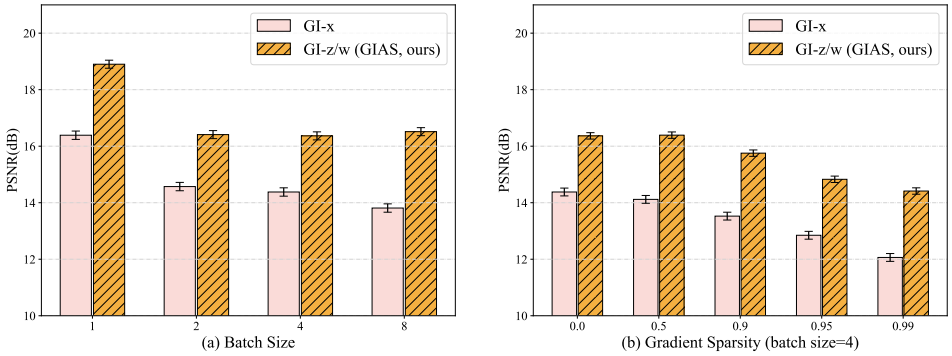


Figure 3: Comparison of reconstruction methods with varying difficulties. (a) Gradient inversion performances varying batch size $B = 1, 2, 4, 8$, where larger B increases the reconstruction difficulty. (b) Gradient inversion performances when the gradient is compressed with different sparsity from 0% to 99%.

4.2 Justification of GIAS Design

We first provide an empirical justification of the specific order of searching spaces in GIAS (GI-z/w) to fully utilize a pretrained generative model. To do so, we provide Figure 2b comparing algorithms with different searching spaces: GI-z, GI-x, GI-z/x, and GI-z/w (GIAS). As shown in Figure 2b(a), By searching only latent space, GI-z finds smooth images, but often perceptually far from original images. This limitation of GI-z is also captured in Figure 2b(b), where the cost function is not decreasing after some optimization steps. To further minimize the cost function, one alternative to GI-z/w (GIAS) is GI-z/x, which can further reduce the loss function but is not better than GI-z/w. This may be because that the parameter space search exploits an implicit bias from optimizing a good architecture for expressing images, c.f., deep image prior [22].

4.3 The gain from fully exploiting pretrained generative model.

Evaluation in different settings We evaluate the gain from using generative models in various scenarios with different difficulties of inverting gradient. As batch size and gradient sparsity[1] [24] increase, the risk of having under-determined inversion and thus the difficulty increase. Figure 3 shows that regardless of the challenge level, the generative model provides significant gain in reconstruction quality. In particular, the quality of GI-x [9] for batch size 1 (resp. gradient sparsity 0%) is comparable to that of GIAS for batch size 8 (resp. gradient sparsity 99%).

Comparison with state-of-the-art models Our method can be easily added to previous methods [9, 25]. In Table 1 and Figure 2, we compare the state-of-the-art methods with or without our methods in addition. Note that our baseline implementation for [25] includes fidelity regularizer with BN_{exact} and group lazy regularizer, not group registration regularizer. Our baseline implementation might be imperfect, but it still shows adding our methods improves performance. Table 1 shows the result, latent space search and parameter space search shows better result. As in Figure 2, our methods have larger variance, as the best-in-batch results are much better than average evaluation. Since privacy issues should consider the worst case, best-in-batch result shows practical threat. It is also interesting

[1]Having gradient sparsity 0.99% implies that we reconstruct data from 1% of the gradient after removing 99% elements with smallest magnitudes at each layer.

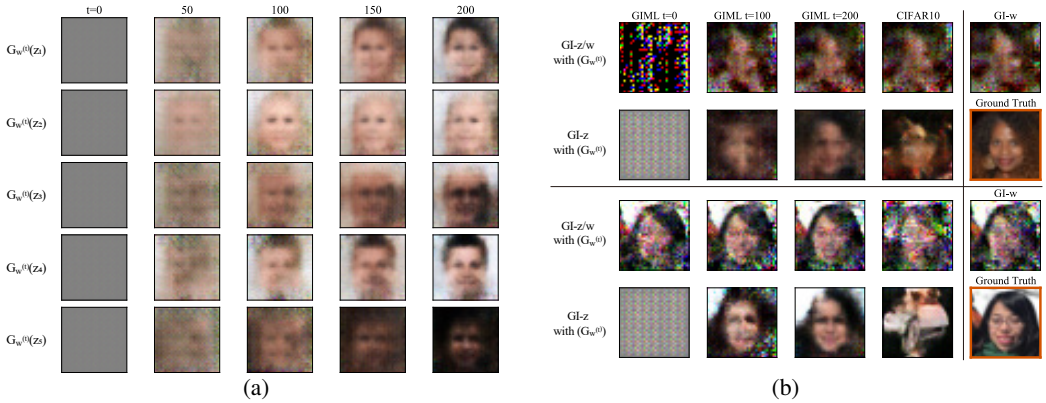


Figure 4: (a) Images sampled from different GIML training steps. Initial generative model is untrained DCGAN, and meta-learned with gradients computed on FFHQ images. same z was used to sample images in same rows. (b) Results of GI-z and GI-z/w using generative model trained with GIML. Experiments were done with gradient sparsity 0.95 for comparison in difficult setting. GI-z/w, GI-z with CIFAR10 pretrained model and GI-w with untrained model is in comparison.

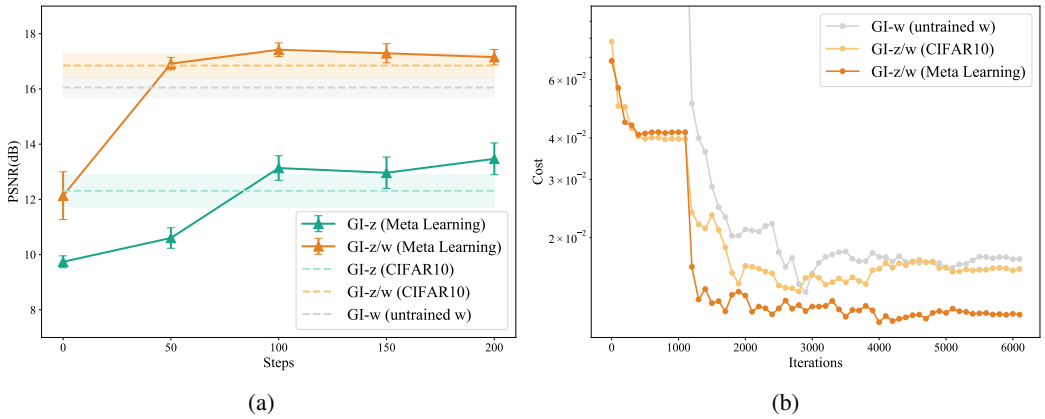


Figure 5: (a) Result of the generative model trained with GIML. With GIML iteration step increases, performance of GI-z and GI-z/w with GIML surpass GI-z and GI-z/w with other models. (b) A typical loss graph of reconstruction process. GI-w with untrained model, GI-z/w with CIFAR10 pretrained model, and GI-z/w with GIML are in comparison. Meta-learned result shows fast convergence than other model's results. Note that GI-w did not perform latent space search of 0 to 1000 iterations.

to note that GI-w with untrained w provides substantial gain compared to GI-x. This may imply that there is a gain of the implicit bias, c.f., [22], from training the architecture of deep generative model.

4.4 Learning generative model from gradients

In this section, we demonstrate the possibility of training a generative model only with gradients. Due to the computational tractability, we use DCGAN and 32x32 images resized from FFHQ [11]. We generate a set of gradients from 4 rounds of gradient reports from 200 nodes, in which node computes gradient for a binary classification task based on the annotation provided in [6]. From the set of gradients, we perform GIML to train DCGAN to generate FFHQ data.

Figure 4 shows that the quality of images from the generative model is improved in the training process of GIML. Along training steps, the generative model starts to generate face images. In Figure 4, as baseline, we also compare the performance of GIAS using three generative models, untrained DCGAN, DCGAN pretrained on CIFAR10, DCGAN trained with meta-learning. For the latent space search, only wrong-dataset model and meta-learned model was compared because latent space search of untrained model is meaningless. The result of latent space search using CIFAR10

pretrained model represents similar color, but does not look like a face. After parameter space search, three models find images close to target image, GI-z/w with GIML achieves the best performance. Not only final image quality, Figure 5 shows meta-learned model converges faster.

5 Conclusion

We propose GIAS fully extracting the prior information on user data from a pretrained generative model when inverting gradient. We demonstrate significant privacy leakage using GIAS with pretrained generative model in various challenging scenarios, where our method provides substantial gain additionally to other any other existing methods [9, 25]. In addition, we propose GIML which can train generative model from gradients only in FL. We experimentally show that GIML can learn a generative model on the user data from only gradients, which provides the same level of data recovery with a given pretrained model. To our best knowledge, GIML is the first capable of learning explicit prior on a set of gradient inversion tasks.

Societal impact and limitation. We demonstrated that in FL, it possible to learn a generative model on user data from gradients. The generative model itself is privacy leakage, and also it can help to invert gradients to explicitly reveal the user data. Although works about gradient inversion, including ours, might be abused to leak privacy, but we believe that the gain from warning against using FL without additional protocol to protect privacy is greater than the risk of abuse. It is unfortunate that we only studied the privacy attacker's algorithm in FL rather than devising privacy protection mechanism, which is definitely our next future work.

References

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel. Siloed federated learning for multi-centric histopathology datasets. In S. Albarqouni, S. Bakas, K. Kamnitsas, M. J. Cardoso, B. Landman, W. Li, F. Milletari, N. Rieke, H. Roth, D. Xu, and Z. Xu, editors, *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pages 129–139, Cham, 2020. Springer International Publishing. ISBN 978-3-030-60548-3.

[3] D. Bau, H. Strobelt, W. Peebles, J. Wulff, B. Zhou, J.-Y. Zhu, and A. Torralba. Semantic photo manipulation with a generative image prior. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi: 10.1145/3306346.3323023. URL `https://doi.org/10.1145/3306346.3323023`.

[4] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[5] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi. Federated learning of predictive models from federated electronic health records. *IJMI*, 2018.

[6] DCGM. ffhq-features-dataset, 2019. URL `https://github.com/DCGM/ffhq-features-dataset`.

[7] L. Fan, K. W. Ng, C. Ju, T. Zhang, C. Liu, C. S. Chan, and Q. Yang. Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks. In Q. Yang, L. Fan, and H. Yu, editors, *Federated Learning: Privacy and Incentive*, pages 32–50. Springer, 2020.

[8] C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*, 2017.

[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients–How easy is it to break privacy in federated learning? In *NeurIPS*, 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[14] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou. Fed{bn}: Federated learning on non-{iid} features via local batch normalization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/pdf?id=6YEQUn0QICG`.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, 2017.

[16] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=BJ0hF1Z0b`.

[17] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symp. Security and Privacy (SP)*, pages 691–706, 2019. doi: 10.1109/SP.2019.00029.

[18] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[20] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017. doi: 10.1109/SP.2017.41.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[22] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2512–2520, 2019.

[24] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu. A framework for evaluating client privacy leakages in federated learning. In L. Chen, N. Li, K. Liang, and S. Schneider, editors, *Computer Security – ESORICS 2020*, pages 545–566, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58951-6.

[25] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, 2021.

[26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[27] B. Zhao, K. R. Mopuri, and H. Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[28] J. Zhu and M. B. Blaschko. R-GAP: Recursive gradient attack on privacy. In *International Conference on Learning Representations*, 2021.

[29] J. Zhu, Y. Shen, D. Zhao, and B. Zhou. In-domain gan inversion for real image editing. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[30] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5 and Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3 and Appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See AppendiX.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our source codes are given in supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]