Gradient Inversion Attacks on Parameter-Efficient Fine-Tuning

Hasin Us Sami, Swapneel Sen, Amit K. Roy-Chowdhury, Srikanth V. Krishnamurthy, Başak Güler
University of California, Riverside, CA
hsami003@ucr.edu, ssen010@ucr.edu, amitrc@ece.ucr.edu, krish@cs.ucr.edu, bguler@ece.ucr.edu

Abstract

Federated learning (FL) allows multiple data-owners to collaboratively train machine learning models by exchanging local gradients, while keeping their private data on-device. To simultaneously enhance privacy and training efficiency, recently parameter-efficient fine-tuning (PEFT) of large-scale pretrained models has gained substantial attention in FL. While keeping a pretrained (backbone) model frozen, each user fine-tunes only a few lightweight modules to be used in conjunction, to fit specific downstream applications. Accordingly, only the gradients with respect to these lightweight modules are shared with the server. In this work, we investigate how the privacy of the fine-tuning data of the users can be compromised via a malicious design of the pretrained model and trainable adapter modules. We demonstrate gradient inversion attacks on a popular PEFT mechanism, the adapter, which allow an attacker to reconstruct local data samples of a target user, using only the accessible adapter gradients. Via extensive experiments, we demonstrate that a large batch of fine-tuning images can be retrieved with high fidelity. Our attack highlights the need for privacy-preserving mechanisms for PEFT, while opening up several future directions. Our code is available at `https://github.com/info-ucr/PEFTLeak`.

1. Introduction

Federated learning (FL) is a collaborative training paradigm to train a machine learning model across multiple data-owners (users) [41]. Users perform training locally using their local data and send the local model updates/gradients to a central server. The server aggregates these to form a global model. By obviating the need to share raw local samples, FL has emerged as a promising framework in fields where data privacy is paramount such as healthcare.

Recently, leveraging large-scale pretrained models in various downstream tasks has gained significant attention, owing to their remarkable training performance. Considering the potential of pretrained models, recent works have extended their application to FL [43, 47, 57]. However, these pretrained models often contain a massive number of parameters, which can be in the range of millions/billions. Full fine-tuning (FFT) of such large models and communicating the gradient parameters require often prohibitive computational infrastructure and bandwidth. This prevents users with low computation/communication resources from participating in training and potentially causing bias in the global model. Thus, recent works have explored parameter-efficient fine-tuning (PEFT) [8, 12, 13, 23, 24, 33, 40, 46], where in lieu of fine-tuning the entire pretrained model, only a small number of lightweight modules are trained; the backbone model is kept frozen. Due to marked reduction in resource consumption and training latency, PEFT has become widely popular in FL [29, 39, 52, 61, 66, 67].

Though FL is popular in privacy-sensitive tasks, an adversarial server can still reverse-engineer the local gradients received from the users to extract privacy-sensitive information about local data samples [20, 21, 25, 26, 37, 42, 44, 55, 59, 63, 64, 70]. These attacks can be grouped into: 1) gradient inversion attacks [17, 20, 21, 63] and, 2) membership inference attacks [42, 49, 55, 60]. Gradient inversion attacks reconstruct the raw data samples held by a target user using the gradient received from the user. In contrast, membership inference attacks seek whether a candidate sample belongs to the local dataset of a target user. The success of membership inference attacks under both FFT and PEFT is shown via a malicious design of the pretrained model [36, 60]. These works do not consider gradient inversion attacks, which can be more detrimental to privacy, as they can operate without the knowledge of candidate samples [16].

A gradient inversion attack is proposed in [16], by poisoning the pretrained vision transformer (ViT) parameters [14]. However, this attack relies on FFT and assumes that the attacker has access to the gradients corresponding to the entire model. Such attacks do not apply to PEFT methods, where the pretrained model remains frozen and the attacker loses access to the full (local) gradient. Due to this, PEFT has been considered to enhance privacy and limit the risk of exposing sensitive information [67]; it has been shown that conventional gradient inversion attacks [70] have a reduced success rate under PEFT. However, these attacks do not con-

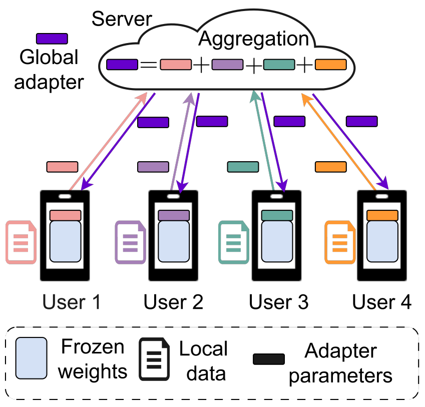


Figure 1. A small scale setup of PEFT based FL with $U = 4$ users. In each training round, users fine-tune the lightweight adapter modules and send the adapter parameters to the server. The backbone model parameters are kept frozen. After aggregation, the server sends the global adapter parameters back to the users.

sider more powerful adversaries where the attacker can deviate from the benign training protocol to breach privacy, such as changing the model architecture and/or parameters sent to the users [16–18].

In this work, we ask the question whether gradient inversion attacks are viable for PEFT. Specifically, we consider gradient inversion attacks on a popular PEFT mechanism, *adapters*, proposed in [23]. This method introduces lightweight modules, called *adapters*, inside the backbone model. Originally introduced for natural language processing tasks, adapters gained significant popularity in vision tasks recently [40, 53, 66]. In an FL setup, each user computes gradients with respect to these adapter parameters only and sends them to the server (attacker), while the backbone model is frozen and not updated during training [29, 66, 67]. The server computes an average of local adapter gradients to form the global (aggregated) adapter parameters. These global parameters are sent back to the users for the next training round. The adversary seeks access to the local image dataset used for local fine-tuning by the users. Compared to conventional gradient inversion attacks targeted at FFT, a major limitation in the PEFT setting (from the attacker's perspective) is that the attacker can only observe the local gradients for a small number of adapter parameters, as the frozen backbone model does not carry any new information. In this context, whether an attacker can recover sensitive training samples of a target user (victim) using the limited information provided by the adapter gradients is yet unknown.

In response to the question above, we propose a novel inversion attack, PEFTLeak, that can successfully recover the data samples of the victim *by only leveraging* the gradients from the adapters. Our attack builds on poisoning the pretrained model and adapter modules, in a way to lead the victim to unintentionally leave a footprint of their raw data inside the adapter gradients. To the best of our knowledge, our work is the first to propose a successful inversion attack applicable to a PEFT method. Our attack highlights the need for stronger defenses for adapters, and provides a number of principles that can also be useful for investigating inversion attacks for other PEFT methods like prefix-tuning [33], bias-tuning [6], or low-rank adaptation [24].

Our contributions are summarized as follows:

1. We demonstrate the first successful gradient inversion attack on PEFT. Our attack can successfully extract local data samples of a victim user using the gradients of lightweight adapters, as opposed to the full model.
2. Extensive experiments on CIFAR-10, CIFAR-100 [30] and TinyImageNet [31] demonstrate that the attacker can breach privacy of fine-tuning data with high-accuracy.
3. We further show that reducing the adapter dimension (number of parameters) does not guarantee privacy. For small dimensions our attack can be executed in multiple rounds to increase the number of reconstructed patches.
4. Our results suggest that PEFT mechanisms, though greatly reducing the number of parameters shared during training, should not be viewed as a defense against inversion attacks, and highlights the necessity of stronger defenses such as differentially private PEFT for FL.

2. Related Works

PEFT. PEFT mechanisms combat large computation and memory costs associated with fine-tuning the entire pretrained model. Such methods freeze the pretrained model while tuning only the bias parameters [6], introducing additional lightweight trainable modules [8, 12, 23, 40, 46], learnable tokens [27, 32, 33] or optimizing low-rank matrices [24, 58, 62]. To enhance training efficiency, recent works have extended PEFT to FL [50, 61, 66, 67].

Privacy attacks. Gradient inversion attacks retrieve local data samples using local gradients. Attacks from [20, 70] recover ground-truth images by minimizing the distance between the true and estimated gradients. Subsequent works leverage batch-normalization statistics [63], blind source separation [28] or a pretrained generative model [15, 26, 34] to recover a batch of images. These works consider fully connected or convolutional neural networks. Attacks on ViTs are explored under the FFT setting in [21, 37, 48, 64]. References [5, 10, 16–18, 68, 69] consider more powerful adversaries who can manipulate the model parameters/architecture to further improve reconstruction.

Membership inference attacks seek whether a specific data sample belongs to the victim user [42, 49, 51]. Such attacks have been explored under both FFT and PEFT settings [2, 19, 55, 60]. Other works recover personally identifiable information or infer the presence of a target property [7, 9, 36, 38, 54]. In contrast, our goal in is to recover the fine-tuning dataset held by the victim user.

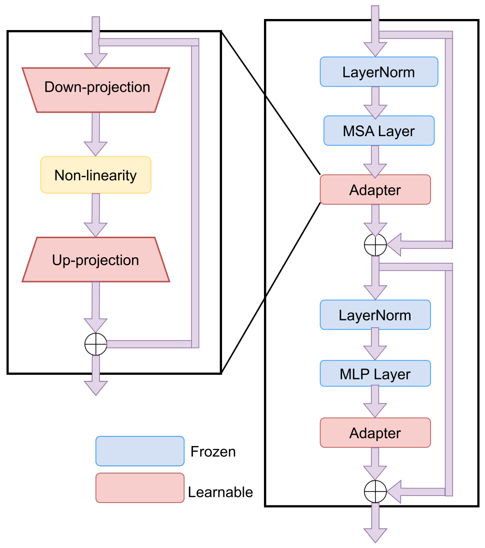


Figure 2. **ViT encoder with adapter modules.** ViT encoder with stacked LayerNorm, MSA, MLP layers and residual connections. An adapter module is inserted after each MSA/MLP layer, consisting of two feed-forward blocks and a non-linearity in-between.

3. Problem Formulation

We consider a typical centralized FL framework with U users. Training is controlled by a server -often an organization with abundant resources- who performs pretraining using proprietary data or publicly available proxy datasets. Fine-tuning enhances task-specific performance by leveraging data from end-users. User i holds a local dataset $\mathcal{D}_i$. At each training round, user i downloads the current state of the global adapter parameters $\mathbf{w}_A \in \mathbb{R}^d$ from the server and performs training locally using its on-device dataset. The goal is to train $\mathbf{w}_A$ to minimize the global loss,

$$\mathcal{L}(\mathbf{w}_F, \mathbf{w}_A) \triangleq \frac{1}{U} \sum_{i\in[U]} \mathcal{L}_i(\mathbf{w}_F, \mathbf{w}_A) \tag{1}$$

where $\mathcal{L}_i$ is the local loss of user i and $\mathbf{w}_F$ is the frozen pretrained model. The frozen parameters are used in forward propagation to compute the loss, but are not updated during backpropagation. This system is illustrated in Fig. 1.

Threat Model. We consider a malicious server who can modify the pretrained model and global adapter parameters. The server sends the pretrained model to the users once prior to training, after which this model is frozen (not updated), hence does not carry any new information from the users. The server sends the global adapter parameters to the users at each training round, which are then updated by the users. After receiving the local adapter gradients, the server initiates the attack to recover the data belonging to the victim user. Our threat model is motivated by [10, 16–18].

ViT with adapters. In ViT, an image is split into multiple patches. Sequences of patch embeddings are generated first and then a distinct position encoding vector is added to each patch embedding. Next, these position-encoded embeddings are used as inputs to multiple stacked multi-head self-attention (MSA) and multi-layer perceptron (MLP) layers. Each MLP layer consists of two fully connected (FCN) layers with a GELU [22] activation function in-between. In addition, a class token is used to predict classes of the fine-tuning samples at the final classification layer (also known as classification head). For fine-tuning, an adapter block is inserted after each MSA and MLP layer [23, 40, 66] as shown in Fig. 2. Each adapter block consists of two FCN layers. The first layer projects the original embeddings of dimension D into a lower dimension r ($r << D$), followed by a non-linear activation function. The second layer maps the projected dimension back to the original dimension, D. By choosing $r << D$, the number of trainable parameters are significantly reduced in PEFT compared to FFT.

The most relevant work to ours is the recent inversion attack to FFT [16], where users send full transformer gradients to the server (as opposed to PEFT). The attacker maliciously modifies the transformer parameters to recover sensitive local training images. Let us denote the total number of patches as N, images in the batch as M, and the (sensitive) patch embeddings by $\{\mathbf{y}^{(n,m)}\}_{n\in[N],m\in[M]}$, which is the sum of image patch embedding and position encoding vectors. The class token embedding is denoted by $\mathbf{y}^{(0,m)}$. The attack allows uninterrupted flow of $\mathbf{y}^{(n,m)}$ until the MLP layer. Let $\mathbf{w}_j$ and b_j, denote the weight vector and bias corresponding to neuron j in the first FCN layer. The output for neuron j can be written as,

$$v_j^{(n,m)} \triangleq \mathbf{w}_j^{\mathrm{T}}\mathbf{y}^{(n,m)} + b_j \; \forall n \in \{0,\dots,N\}, m \in [M] \tag{2}$$

where the gradient of user i with respect to $\mathbf{w}_j$ and b_j is,

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} = \frac{1}{M(N+1)} \sum_{m=1}^{M}\sum_{n=0}^{N} \frac{\partial \mathcal{L}_i}{\partial v_j^{(n,m)}} \mathbf{y}^{(n,m)} \tag{3}$$

$$\frac{\partial \mathcal{L}_i}{\partial b_j} = \frac{1}{M(N+1)} \sum_{m=1}^{M}\sum_{n=0}^{N} \frac{\partial \mathcal{L}_i}{\partial v_j^{(n,m)}} \tag{4}$$

which represents the average of gradients for all patches and class tokens across the images. Let us assume only patch n from image m propagates through the activation function in neuron j while all other patches are blocked. Then,

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} = \frac{1}{M(N+1)} \frac{\partial \mathcal{L}_i}{\partial v_j^{(n,m)}} \mathbf{y}^{(n,m)},$$

$$\frac{\partial \mathcal{L}_i}{\partial b_j} = \frac{1}{M(N+1)} \frac{\partial \mathcal{L}_i}{\partial v_j^{(n,m)}}$$

and embedding $\mathbf{y}^{(n,m)}$ can be recovered as,

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} \Big/ \frac{\partial \mathcal{L}_i}{\partial b_j} = \mathbf{y}^{(n,m)} \tag{5}$$

The attacker then designs the weight and bias parameters in the MLP layer with the constraint that no two patches/tokens activate the same set of neurons, by leveraging patch statistics obtained from a public dataset which was originally proposed in [17]. After recovering the embeddings $\mathbf{y}^{(n,m)}$, one can recover the image patch embeddings by subtracting the position encoding vectors from $\mathbf{y}^{(n,m)}$.

Challenges. In adapter-based PEFT [23], MLP layers are initialized prior to training and kept frozen afterwards (not updated by users). As a result, these layers do not carry any information about the local fine-tuning data, making the aforementioned attack infeasible. A potential approach is then to use the adapter gradients to recover the image embeddings as in (5). For this, the target embeddings $\{\mathbf{y}^{(n,m)}\}_{n\in[N],m\in[M]}$ should propagate through all intermediate layers before the adapter without any significant shift in the original contents, which requires the frozen MSA and MLP layers act as identity mappings. The key challenge is then the projection to lower dimension in the adapter, due to which the attacker has access to a maximum of r weight and bias gradients. As seen in [8, 23], the typical values for r lie within the range of 1 to 64. As evident in (5), a single neuron's weight and bias gradients play a role in recovering a single image patch. Reconstruction rate increases with the number of available neurons, which was also shown in [17, 68]. The MLP layer of the ViT architecture from [14] has 3072 neurons. These 3072 neurons' gradients were leveraged for the attack in [16], leading to the succesful recovery of a large batch of images. In contrast, fine-tuning results in modifying only a small set of parameters (corresponding to $r \leq 64$ neurons in the adapter) compared to the images, which makes it hard to reconstruct a large fraction of the images using this limited information.

Contributions. We address these challenges by crafting a novel gradient inversion attack on adapter-based PEFT. To do so, we introduce a malicious design of the pretrained model and adapters, which can uncover a large number of private fine-tuning images with a much reduced observable space compared to what was available with FFT. Our attack tackles the dimensionality problem (limited number of neurons) for the adapter by leveraging the gradients from multiple adapter layers, and carefully designs the patch propagation structure to enable different adapter layers to recover images with different features.

4. Framework

We next describe the details of PEFTLeak. We consider the conventional ViT architecture from [14] as the pretrained model. The architecture along with inserted adapter modules is depicted in Fig. 2. LayerNorm, MSA and MLP layers are kept frozen while only adapter modules are trainable [8, 40]. We assume that the victim performs fine-tuning on a batch of M images. We denote the m^{th} image in the batch as $\mathbf{X}(m) \in \mathbb{R}^{C\times H\times W}$ for $m \in [M]$, where C is the number of channels, and (H, W) denotes the image resolution. Each image is divided into N patches of resolution (P, P). Next, each patch is flattened to a vector, and denoted as $\mathbf{x}^{(n,m)} \in \mathbb{R}^{P^2C}$ for $n \in [N], m \in [M]$. We assume that the range of input values in $\mathbf{x}^{(n,m)}$ are in $[-1, 1]$ [45, 68]. The patches are then mapped to a dimension D through linear projection using the weight matrix $\mathbf{E} \in \mathbb{R}^{D\times P^2C}$ from the pretrained model,

$$\mathbf{x}_{map}^{(n,m)} \triangleq \mathbf{E}\mathbf{x}^{(n,m)} \tag{6}$$

for $n \in [N]$. A class token $\mathbf{x}_{map}^{(0,m)}$ is used to predict the class of image m in the classification layer, which propagates like the image patch embeddings throughout the intermediate layers. Next, position encoding vectors $\mathbf{E}_{pos}^{(n)} \in \mathbb{R}^D$ for $n \in \{0, \dots, N\}$ are added to the embeddings,

$$\mathbf{y}^{(n,m)} \triangleq \mathbf{x}_{map}^{(n,m)} + \mathbf{E}_{pos}^{(n)} = \mathbf{E}\mathbf{x}^{(n,m)} + \mathbf{E}_{pos}^{(n)} \tag{7}$$

Key intuition. At a high-level, our goal is to propagate the embeddings $\mathbf{y}^{(n,m)}$ up to the adapter layers without any significant distortion. This should be achieved by a one-time adversarial modification of the pretrained model, which is sent to the users prior to fine-tuning. As these layers are frozen and not modified by the users during fine-tuning, they provide no information about the local image patches. Adapter layers will then be utilized to recover the image patches. These trainable parameters are modified by the users during fine-tuning, and hence encode sensitive information about the local images. On the other hand, the dimensionality of a single adapter layer is not sufficient to recover a large batch of images. To that end, our attack is designed across multiple adapter layers, where different layers capture image patches with varying statistics. We next describe the design of the individual layers.

4.1. First LayerNorm (LN1)

After position encoding, the embeddings $\mathbf{y}^{(n,m)}$ enter a LayerNorm (LN1) layer,

$$\mathbf{z}^{(n,m)} \triangleq \frac{\mathbf{y}^{(n,m)} - \mu^{(n,m)}}{\sigma^{(n,m)}} \odot \mathbf{w}_{LN1} + \mathbf{b}_{LN1} \tag{8}$$

where $\mathbf{w}_{LN1}, \mathbf{b}_{LN1} \in \mathbf{R}^D$ are the weight and bias parameters at LN1. $\mu^{(n,m)}, \sigma^{(n,m)}$ denote the mean and standard deviation across the elements in $\mathbf{y}^{(n,m)}$ and $\odot$ denotes element-wise multiplication. As our goal is to ensure uninterrupted flow of information from the input layer, we want $\mathbf{z}^{(n,m)} \approx \mathbf{y}^{(n,m)}$. For this, we design $\mathbf{E}$ to make the mean and standard deviation across $\mathbf{x}_{map}^{(n,m)}$ in (6) negligible for all $n \in [N], m \in [M]$ compared to the mean and standard deviation of $\mathbf{E}_{pos}^{(n)}$. Selecting $\mathbf{E}_{pos}^{(n)} \sim \mathcal{N}(0, \sigma)$ with $\sigma = 10$ and $\mathbf{E} = 0.5\mathbf{I}_D$ satisfies these conditions, $\mu^{(n,m)} \approx 0$, and $\sigma^{(n,m)} \approx \sigma$ for all $n \in \{0, \dots, N\}$. Then, by setting each element in $\mathbf{w}_{LN1}$ to σ and $\mathbf{b}_{LN1}$ to 0, $\mathbf{z}^{(n,m)} \approx \mathbf{y}^{(n,m)}$.

4.2. Multi-head Self Attention (MSA)

The output embeddings of LN1 enter the Multi-head Self Attention (MSA) layer. To enable undistorted propagation of input patch embeddings, we design the weight parameters in the MSA layer to act as identity mappings. Suppose that the MSA layer consists of L heads where the embedding dimension of each head is $D_h \triangleq D/L$. For head h, denote the query, key and value weight matrices as $\mathbf{W}_Q^h, \mathbf{W}_K^h$ and $\mathbf{W}_V^h$, and biases as $\mathbf{b}_Q^h, \mathbf{b}_K^h$ and $\mathbf{b}_V^h$, respectively. Let,

$$\mathbf{W}_Q^h = \mathbf{W}_K^h = \mathbf{W}_V^h = \mathbf{I}_{D_h \times D_h} \tag{9}$$

$$\mathbf{b}_Q^h = \mathbf{b}_K^h = \mathbf{b}_V^h = \mathbf{0} \tag{10}$$

and define $(\mathbf{y}^{(n,m)})_h \triangleq \mathbf{y}^{(n,m)}[hD_h : (h+1)D_h] \in \mathbb{R}^{D_h}$ as the D_h elements from the n^{th} patch embedding that propagates through head h. Then, query, key and value matrices for head h are given by,

$$\begin{aligned}
\mathbf{Q}^h &\triangleq \left[\mathbf{W}_Q^h(\mathbf{y}^{(0,m)})_h \quad \cdots \quad \mathbf{W}_Q^h(\mathbf{y}^{(N,m)})_h\right] \\
\mathbf{K}^h &\triangleq \left[\mathbf{W}_K^h(\mathbf{y}^{(0,m)})_h \quad \cdots \quad \mathbf{W}_K^h(\mathbf{y}^{(N,m)})_h\right] \\
\mathbf{V}^h &\triangleq \left[\mathbf{W}_V^h(\mathbf{y}^{(0,m)})_h \quad \cdots \quad \mathbf{W}_V^h(\mathbf{y}^{(N,m)})_h\right]
\end{aligned} \tag{11}$$

Note that position encoding vectors $\mathbf{E}_{pos}^{(n)}$ are generated independently from $\mathcal{N}(0, 10)$, hence for $n \neq t$,

$$(\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{E}_{pos}^{(t)} >> (\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{E}_{pos}^{(n)} \tag{12}$$

As a result, the attention matrix becomes an identity matrix,

$$\mathbf{A}^h \triangleq softmax((\mathbf{Q}^h)^{\mathsf{T}}\mathbf{K}^h/\sqrt{D_h}) \cong \mathbf{I}_{(N+1)\times(N+1)}$$

After multiplying with the value matrix $\mathbf{V}^h$, the self-attention output for head h is $SA_h([\mathbf{y}^{(0,m)} \cdots \mathbf{y}^{(N,m)}]) \triangleq \mathbf{A}^h(\mathbf{V}^h)^{\mathsf{T}}$. Let $\mathbf{W}_{MSA} \in \mathbb{R}^{D\times D}$ denote the weight matrix that performs the linear transformation on the concatenated outputs from all heads. We set $\mathbf{W}_{MSA} = \mathbf{I}_{D\times D}$, after which the MSA output is,

$$\begin{aligned}
&MSA([\mathbf{y}^{(0,m)} \quad \cdots \quad \mathbf{y}^{(N,m)}]) \\
&\triangleq \left[SA_1([\mathbf{y}^{(0,m)} \quad \cdots \quad \mathbf{y}^{(N,m)}]) \quad \cdots \right. \\
&\qquad \left. SA_L([\mathbf{y}^{(0,m)} \quad \cdots \quad \mathbf{y}^{(N,m)}])\right] \times \mathbf{W}_{MSA} \quad (13) \\
&\cong \left[(\mathbf{y}^{(0,m)}) \quad \cdots \quad (\mathbf{y}^{(N,m)})\right]^{\mathsf{T}}_{(N+1)\times D} \quad (14)
\end{aligned}$$

which ensures the flow of $\mathbf{y}^{(n,m)}$ for $n \in [N]$, $m \in [M]$.

4.3. Adapter Layer

The next layer is the adapter, consisting of two linear layers (one down-projection and one up-projection) with an activation function in-between. Down-projection projects the input embeddings of dimension D to a lower dimension $r << D$. Recall that adapter modules are trainable. By using the gradients in the down-projection layer along with

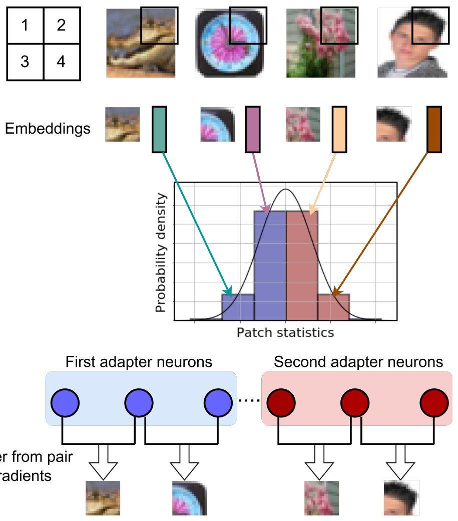


Figure 3. **Image recovery from multiple adapters.** A small-scale example with two adapters, each consisting of 3 neurons in the down-projection layer. The attacker aims to recover 4 images in the batch, where each image is divided into 4 patches. The biases for the two adapters are designed to recover patches from the second position. The plot represents the distribution of patch statistics, $(\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{x}_{map}^{(t,m)}$ for $t = 2$. The attacker targets images whose patch statistics lie in one of the four intervals (colored). Each adapter targets two intervals. By leveraging multiple adapters, the attacker can recover all the patches in the target position.

the activation, one can try to recover the input embeddings $\{\mathbf{y}^{(n,m)}\}_{n\in[N],m\in[M]}$ as in (5). On the other hand, the number of neurons r typically ranges from 1 to 64 [8, 23], which severely limits the number of images recovered, as reconstruction performance in (5) degrades when the number of neurons decreases [17, 68]. To enable recovery of a large batch of images, we propose to use the neurons from multiple adapters, where different layers are designed to recover images with different statistical properties.

We first discuss how to recover a patch from a target position within an image. For a target position $t \in [N]$, the goal is to allow the patches from position t pass the activation function, and filter out the patches from all other positions $n \neq t$. We allocate $k_t > r$ neurons to recover patches from position t. As each adapter has r neurons, $S_t \triangleq \frac{k_t}{r}$ adapters are used to recover patches from position t, and $S \triangleq \sum_{n=1}^{N} S_n$ adapters to recover the patches from all positions $t \in [N]$. Denote the set of neurons used for position t as $\mathcal{N}_t$. For each neuron $j \in \mathcal{N}_t$, we set the corresponding weight vector to $\mathbf{E}_{pos}^{(t)}$, after which the output for patch t is,

$$\begin{aligned}
v_j^{(t,m)} &= (\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{y}^{(t,m)} + b_j && (15) \\
&= (\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{x}_{map}^{(t,m)} + (\mathbf{E}_{pos}^{(t)})^{\mathsf{T}}\mathbf{E}_{pos}^{(t)} + b_j && (16)
\end{aligned}$$

whereas for all other positions $n \in \{0, \ldots, N\}\backslash\{t\}$,

$$\begin{aligned} v_j^{(n,m)} &= (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{y}^{(n,m)} + b_j \\ &= (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{x}_{map}^{(n,m)} + (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{E}_{pos}^{(n)} + b_j \end{aligned} \tag{17}$$

We next discuss how this design plays a key role in propagating patches from target position t while blocking patches from all other positions. Note that the server does not know $(\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{x}_{map}^{(t,m)}$ as it does not have access to local data. Along the lines of [17, 18], we assume the server can instead estimate an approximate distribution for this quantity by utilizing a public dataset, which resembles a Gaussian distribution from the central limit theorem [17]. Define,

$$c_j \triangleq \psi^{-1}(j/k_t) \tag{18}$$

where $\psi^{-1}(\cdot)$ is the inverse CDF of the estimated Gaussian. Then, the bias for neuron j is designed as,

$$b_j \triangleq -(\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{E}_{pos}^{(t)} - c_j \tag{19}$$

If for any given patch t and image m,

$$c_j < (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{x}_{map}^{(t,m)} < c_{j+1} \tag{20}$$

then, from (16), (19) and (20) we observe for patch t,

$$v_j^{(t,m)} = (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{y}^{(t,m)} + b_j = (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{x}_{map}^{(t,m)} - c_j > 0 \tag{21}$$

whereas for all other patches $n \in \{0, \dots, N\}\backslash\{t\}$,

$$v_j^{(n,m)} = (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{y}^{(n,m)} + b_j << 0 \tag{22}$$

which follows from (12), hence (22) filters out the patches from all other positions after the activation function. Next, for the target position t, we need to ensure unique recovery of each image patch in the batch. For this, we utilize the technique from [17] to successively block images by the activation function. According to (16), (19) and (20),

$$v_{j+1}^{(t,m)} = (\mathbf{E}_{pos}^{(t)})^{\mathrm{T}}\mathbf{x}_{map}^{(t,m)} - c_{j+1} < 0 \tag{23}$$

hence $\mathbf{y}^{(t,m)}$ propagates through the activation function at neuron j as in (21), but is blocked at neuron $j+1$ as in (23). If no other image satisfies (20), then by leveraging the gradients for this pair of neurons, the attacker can recover,

$$\left(\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_{j+1}}\right) \Big/ \left(\frac{\partial \mathcal{L}_i}{\partial b_j} - \frac{\partial \mathcal{L}_i}{\partial b_{j+1}}\right) = \mathbf{y}^{(t,m)} \tag{24}$$

After computing c_j for $j \in [k_t]$ as in (18), we can utilize $c_{(s-1)r+1}, \dots, c_{sr}$ to design the biases of r neurons as in (19) for the s^{th} adapter for $s \in [S_t]$. As long as only one patch lies within an interval $[c_j, c_{j+1}]$ for $j \in [k_t]$, perfect recovery can be achieved for that particular patch. From a single adapter, the attacker can recover at most $r-1$ patches that lie within one of the $r-1$ intervals. Attacks to more intervals can be deployed by utilizing S_t adapters for position t. We demonstrate an illustrative example in Fig. 3. After recovering the embedding $\mathbf{y}^{(t,m)}$, the attacker can recover,

$$\mathbf{x}^{(t,m)} = \mathbf{E}^{\dagger}(\mathbf{y}^{(t,m)} - \mathbf{E}_{pos}^{(t)}) \tag{25}$$

where $\mathbf{E}^{\dagger}$ is the pseudoinverse of $\mathbf{E}$ in (6). The weight and bias for the up-projection are designed to produce zero output, to ensure we regain the target embeddings $\{\mathbf{y}^{(n,m)}\}_{n\in[N],m\in[M]}$ after the residual connection in Fig. 2. However, if we simply set all parameters to zero, the gradient for the preceding layer will be zero during backpropagation, making reconstruction impossible. We prevent this by setting the first neuron's weight to a small non-zero value ($\sim 10^{-6}$). We next discuss how to propagate the current layer's output (embeddings $\mathbf{y}^{(n,m)}$), from one adapter layer to the next to continue reconstruction.

4.4. Second LayerNorm (LN2)

As shown in Fig. 2, the adapter output goes through a residual connection from the LN1 input, which is equal to $\mathbf{y}^{(n,m)}$ from (8). The output of the residual connection is then,

$$\mathbf{e}^{(n,m)} \triangleq (1+1)\mathbf{y}^{(n,m)} \cong 2\mathbf{y}^{(n,m)} \tag{26}$$

for $n \in \{0, \dots, N\}$, which enters another LayerNorm (LN2). Note that $\mathbf{y}^{(n,m)}$ has mean approximately equal to 0 and standard deviation σ from (8). Therefore, $\mathbf{e}^{(n,m)}$ has mean 0 and standard deviation 2σ. The server can then set the elements in the LN2 weight vector to σ and bias vector to 0 to retrieve back the original embeddings.

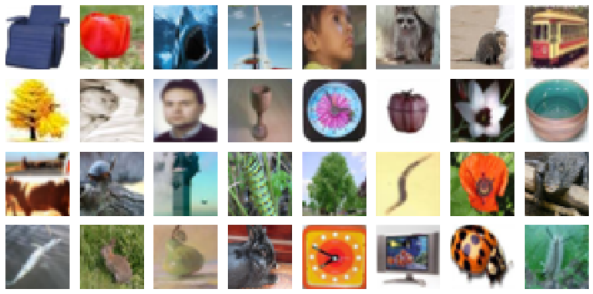
$$\frac{\mathbf{e}^{(n,m)} - 0}{2\sigma}\sigma + 0 \cong \mathbf{y}^{(n,m)} \tag{27}$$

4.5. Multi Layer Perceptron (MLP)

Next is the MLP layer, consisting of two linear layers with a GELU activation in-between. The first layer maps the embeddings to a higher dimension $4D$, whereas the second layer maps them back to the original dimension D [14]. Our goal is to propagate the embeddings $\mathbf{y}^{(n,m)}$ to the next adapter undistorted. For this, the MLP needs to act as an identity function. Let $\mathbf{W}_{MLP,1} \in \mathbb{R}^{4D\times D}$ be the weight matrix in the first layer, with row p and column q given as,

$$\mathbf{W}_{MLP,1}[p,q] \triangleq \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

which makes the output equal to the original embeddings, $\mathbf{y}^{(n,m)}$ in the first D coordinates, while the remaining $3D$ coordinates are 0. Note that original embeddings may contain negative values with a large magnitude, which will be filtered out by GELU, whereas values close to zero are subject to attenuation. To avoid this, we set the bias as $\mathbf{b}_{MLP,1} \triangleq \gamma\mathbf{1}_{4D}$ with a large value $\gamma = 10^4$, where $\mathbf{1}_{4D}$


(a) Original images

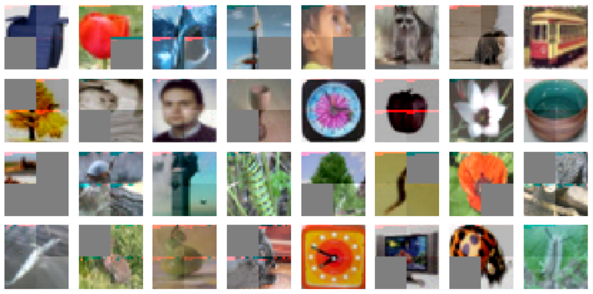

(b) Recovered

Figure 4. CIFAR-100 (recovered images for a batch of 32 images).

	LPIPS		SSIM		MSE	
	Mean	Std	Mean	Std	Mean	Std
CIFAR-10	0.10	0.09	0.74	0.11	0.21	0.12
CIFAR-100	0.08	0.06	0.88	0.12	0.20	0.16
TinyImageNet	0.12	0.04	0.76	0.10	1.06	0.82

Table 1. Mean and standard deviation of MSE, LPIPS and SSIM scores for different datasets.

is a $4D$-dimensional vector containing all 1s. This ensures that the neuron outputs are not affected by GELU. By setting the weight matrix $\mathbf{W}_{MLP,2} \in \mathbb{R}^{D\times 4D}$ and bias $\mathbf{b}_{MLP,2} \in \mathbb{R}^D$ in the second linear layer as,

$$\mathbf{W}_{MLP,2}[p,q] \triangleq \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

and $\mathbf{b}_{MLP,2} \triangleq -\gamma \mathbf{1}_D$, we ensure that the MLP output is equal to the original embeddings $\mathbf{y}^{(n,m)}$ for $n \in \{0,\dots,N\}$, $m \in [M]$. We adopt the same design in the subsequent LayerNorm, MSA and MLP layers. This ensures that the target embeddings $\mathbf{y}^{(n,m)}$ propagate through all intermediate layers towards the adapters. After recovering raw image patches, the final step is to group the patches that belong to the same image. For this, we use a similar approach to [18], which embeds a unique tag in the first MSA layer to each patch belonging to the same source image.

5. Experiments

Our experiments aim to answer the following questions:

- How does PEFTLeak perform in terms of reconstruction?
- How is performance impacted with varying bottleneck dimension r inside the adapters?
- How is performance impacted with increasing batch size?
- How do we benefit from leveraging the layers from all the adapter modules for reconstruction?

Setup. We consider a distributed setting for image classification using ViT-B/16 [14] as the pretrained model. Each user holds data samples from CIFAR-10, CIFAR-100 [30] and TinyImageNet [31] datasets. The experiments are run on a 24 core AMD Ryzen device with NVIDIA RTX4000.

Hyperparameters. In our experiments, each image is divided into patches of size $(16, 16)$ in accordance with [14]. For CIFAR-10 and CIFAR-100, each image (with resolution $(32, 32)$) is divided into 4 patches. For TinyImageNet, each image (with resolution $(64, 64)$) is divided into 16 patches. The embedding dimension is $D = 768$ [14]. The bottleneck dimension for the adapters is $r = 64$.

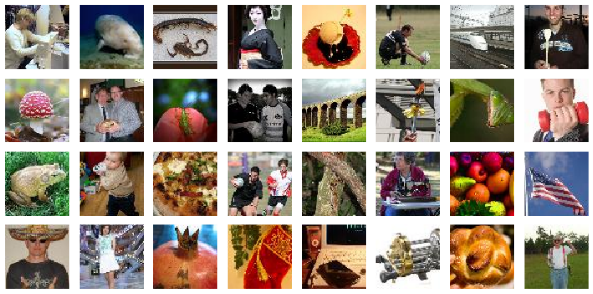
Performance Metrics. Attack performance is evaluated using the mean squared error (MSE), and perceptual/structural similarity scores LPIPS [65] and SSIM [56] between recovered and ground-truth images. Lower MSE/LPIPS or higher SSIM implies better reconstruction.

Results. In Fig. 4, we present the reconstructed images from the local gradient of a target user. The gradient is derived from training on a batch of 32 images from CIFAR-100. Since each image is divided into 4 patches, there are $32\times4 = 128$ target patches that the attacker aims to recover. As we observe, 110-out-of-128 image patches (85.9% of the patches) are recovered. Gray patches indicate that the corresponding ground-truth patches are not recovered. In Fig. 5, we present the reconstructed images for TinyImageNet. Since in this case each image is divided into 16 patches, the attacker aims for $32 \times 16 = 512$ different patches. We observe that around 81% of the total patches are recovered. Reconstruction for CIFAR-10 is presented in App. C.1. In Table 1, we report the mean and standard deviation of MSE, LPIPS, and SSIM scores across the recovered patches.

We next study the impact of batch size, bottleneck dimension, and number of adapter layers on reconstruction. These experiments are conducted using CIFAR-100.

Reconstruction rate vs. batch size. We first evaluate the impact of batch size on the success rate. Since reconstruction is performed patch-wise, we compare the percentage of patches recovered across different batch sizes. As we observe in Fig. 6a, even for a batch size as large as 128, PEFTLeak recovers up to 72.6% of the patches.

Reconstruction rate vs. bottleneck dimension. We present the impact of bottleneck dimension r on the attack efficiency in Fig. 6b. As r increases, more patches are retrieved. Higher values of r imply more neurons in the adapter layer that can be used for recovery. This sug-


(a) Original images

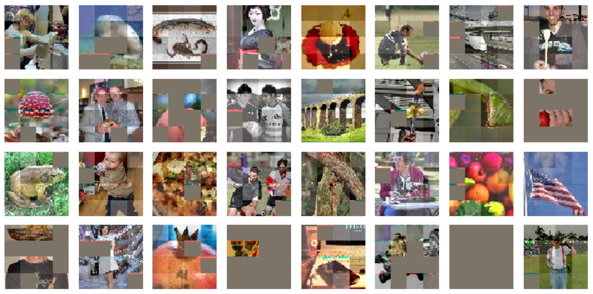
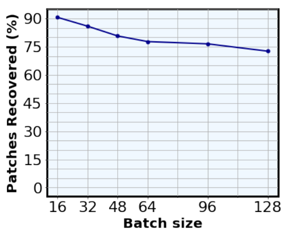
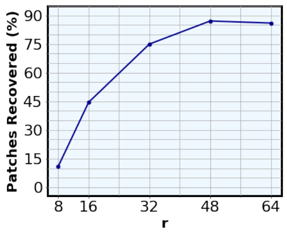

(b) Recovered

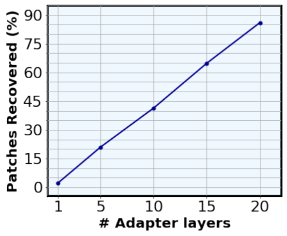
Figure 5. TinyImageNet (recovered images for a batch of 32 images).



(a) Varying batch size



(b) Varying r



(c) Varying # adapter layers

Figure 6. Percentage of patches recovered with varying batch size, bottleneck dimension, and number of adapter layers used.

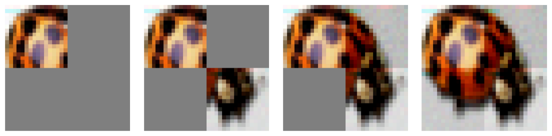

(a) Round 1 (b) Round 2 (c) Round 4 (d) Round 5

Figure 7. Patches recovered over multiple rounds ($r = 8$).

gests that choosing a small r could be a potential defense against privacy attacks. However, as we show later, this can be countered by using multiple training rounds.

Reconstruction rate vs. number of adapter layers. We further analyze the impact of the number of adapter layers on the attack success in Fig. 6c. We observe that as we use more layers for the attack, more patches can be recovered. Even though a single adapter may have insufficient neurons due to down-projection, the leakage rate is enhanced by leveraging multiple adapter layers.

Attack over multiple training rounds. We next demonstrate how the attack can be executed over multiple training rounds to increase the number of reconstructed patches for a small bottleneck dimension, in particular for $r = 8$. As mentioned earlier, each user sends the adapter gradients to the server in each training round, while the pretrained model is kept frozen. The server sends the aggregated adapter parameters in each training round back to the users. Hence, the server can tamper with these adapter parameters so that in each round, different sets of intervals from (20) can be targeted. For the image in Fig. 7, we observe that only a single patch is recovered from gradients within a single training round. However, from gradients over 5 training rounds, the entire image is recovered. Hence, choosing a small value for r does not guarantee privacy.

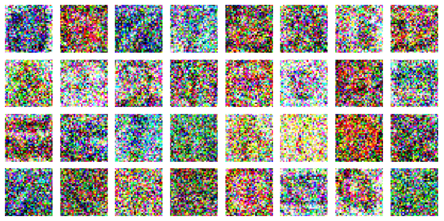


Figure 8. Recovered images with attack from [20] (CIFAR-100).

Comparison with optimization-based baseline. In Fig. 8, we apply the optimization-based attack from [20] for the images in Fig. 4a (details are provided in App C.2).

Batch size	8	16	32	48	64
% **Patches recovered**	90	84.7	73.2	66.5	62.6

Table 2. Reconstruction with varying batch size (ImageNet).

Scalability to high resolution dataset. Table 2 presents patch recovery on ImageNet [11], with image resolution $(224, 224)$. Each image is divided into 196 patches of size $(16, 16)$. Additional experiments are provided in App. C.

6. Conclusion

We show how an adversarial server can manipulate the pretrained model and adapter parameters to uncover users' local fine-tuning data in PEFT for FL, despite a significantly reduced embedding space. Our attacks demonstrate the critical need for privacy-aware PEFT mechanisms, with future directions including certifiable privacy guarantees.

7. Acknowledgement

The research was supported in part by the OUSD (R&E)/RT&L under Cooperative Agreement W911NF-20-2-0267, NSF CAREER Award CCF-2144927, ARO grant W911NF2210260, and the UCR OASIS Grant. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL and OUSD(R&E)/RT&L or the U.S. Government.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. B. McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016. 15

[2] John Abascal, Stanley Wu, Alina Oprea, and Jonathan R. Ullman. Tmi! finetuned models leak private information from their pretraining data. *Proc. Priv. Enhancing Technol.*, 2024(3):202–223, 2024. 2

[3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (Neurips)*, pages 1709–1720, 2017. 15

[4] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (Neurips)*, pages 5977–5987, 2018. 15

[5] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *8th IEEE European Symposium on Security and Privacy, EuroS&P*, pages 175–199, 2023. 2

[6] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. In *Neural Information Processing Systems*, 2020. 2

[7] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium*, pages 2633–2650, 2021. 2

[8] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems (Neurips)*, 2022. 1, 2, 4, 5

[9] Xiaoyi Chen, Siyuan Tang, Rui Zhu, Shijun Yan, Lei Jin, Zihao Wang, Liya Su, XiaoFeng Wang, and Haixu Tang. The janus interface: How fine-tuning in large language models amplifies the privacy risks. *Arxiv*, 2023. 2

[10] Hong-Min Chu, Jonas Geiping, Liam H. Fowl, Micah Goldblum, and Tom Goldstein. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023. 2, 3

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 8

[12] Wei Dong, Dawei Yan, Zhijun Lin, and Peng Wang. Efficient adaptation of large vision transformer via adapter re-composing. In *Advances in Neural Information Processing Systems (Neurips)*, 2023. 1, 2

[13] Wei Dong, Xing Zhang, Bihui Chen, Dawei Yan, Zhijun Lin, Qingsen Yan, Peng Wang, and Yang Yang. Low-rank rescaled vision transformer fine-tuning: A residual design approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 16101–16110. IEEE, 2024. 1

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR*, 2021. 1, 4, 6, 7

[15] Hao Fang, Bin Chen, Xuan Wang, Zhi Wang, and Shu-Tao Xia. GIFD: A generative gradient inversion method with feature domain optimization. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 4944–4953. IEEE, 2023. 2

[16] Shanglun Feng and Florian Tramèr. Privacy backdoors: Stealing data with corrupted pretrained models. In *Forty-first International Conference on Machine Learning, ICML*, 2024. 1, 2, 3, 4

[17] Liam H. Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 1, 4, 5, 6, 15

[18] Liam H. Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023. 2, 3, 6, 7

[19] Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. Practical membership inference attacks against fine-tuned large language models via self-prompt calibration. *Arxiv*, 2023. 2

[20] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 8, 12, 13

[21] Ali Hatamizadeh, Hongxu Yin, Holger Roth, Wenqi Li, Jan Kautz, Daguang Xu, and Pavlo Molchanov. Gradvit: Gradient inversion of vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10011–10020. IEEE, 2022. 1, 2

[22] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv*, 2016. 3

[23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, pages 2790–2799. PMLR, 2019. 1, 2, 3, 4, 5

[24] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR*, 2022. 1, 2

[25] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems (Neurips)*, pages 7232–7241, 2021. 1

[26] Jinwoo Jeon, Jaechang Kim, Kangwook Lee, Sewoong Oh, and Jungseul Ok. Gradient inversion with generative image prior. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 29898–29908, 2021. 1, 2

[27] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision ECCV*, pages 709–727, 2022. 2

[28] Sanjay Kariyappa, Chuan Guo, Kiwan Maeng, Wenjie Xiong, G. Edward Suh, Moinuddin K. Qureshi, and Hsien-Hsin S. Lee. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In *International Conference on Machine Learning, ICML*, 2023. 2

[29] Yeachan Kim, Junho Kim, Wing-Lam Mok, Jun-Hyung Park, and SangKeun Lee. Client-customized adaptation for parameter-efficient federated learning. In *Findings of the Association for Computational Linguistics: ACL*, pages 1159–1172, 2023. 1, 2

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 2, 7

[31] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. 2015. 2, 7

[32] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 3045–3059, 2021. 2

[33] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP*, pages 4582–4597, 2021. 1, 2

[34] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10122–10132. IEEE, 2022. 2

[35] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *6th International Conference on Learning Representations, ICLR*, 2018. 15

[36] Ruixuan Liu, Tianhao Wang, Yang Cao, and Li Xiong. Precurious: How innocent pre-trained language models turn into privacy traps. In *Proceedings of the on ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 3511–3524, 2024. 1, 2

[37] Jiahao Lu, Xi Sheryl Zhang, Tianli Zhao, Xiangyu He, and Jian Cheng. APRIL: finding the achilles' heel on privacy for vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10041–10050. IEEE, 2022. 1, 2

[38] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella Béguelin. Analyzing leakage of personally identifiable information in language models. In *44th IEEE Symposium on Security and Privacy, SP*, pages 346–363. IEEE, 2023. 2

[39] Shubham Malaviya, Manish Shukla, and Sachin Lodha. Reducing communication overhead in federated learning for pre-trained language models using parameter-efficient finetuning. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, pages 456–469, 2023. 1

[40] Imad Eddine Marouf, Enzo Tartaglione, and Stéphane Lathuilière. Mini but mighty: Finetuning vits with mini adapters. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV*, pages 1721–1730. IEEE, 2024. 1, 2, 3, 4

[41] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Int. Conf. on Artificial Int. and Stat. (AISTATS)*, 2017. 1

[42] Milad Nasr, R. Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *IEEE Symposium on Security and Privacy (SP)*, pages 739–753, 2019. 1, 2

[43] John Nguyen, Jianyu Wang, Kshitiz Malik, Maziar Sanjabi, and Michael G. Rabbat. Where to begin? on the impact of pre-training and initialization in federated learning. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023. 1

[44] Truc D. T. Nguyen, Phung Lai, Khang Tran, NhatHai Phan, and My T. Thai. Active membership inference attack under local differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 5714–5730, 2023. 1

[45] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Conference on Computer and Communications Security, CCS*, pages 2429–2443. ACM, 2022. 4

[46] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulic, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing EMNLP*, pages 46–54, 2020. 1, 2

[47] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Li Fei-Fei, Ehsan Adeli, and Daniel L. Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10051–10061, 2022. 1

[48] Daniel Scheliga, Patrick Maeder, and Marco Seeland. Dropout is NOT all you need to prevent gradient leakage. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, pages 9733–9741, 2023. 2

[49] R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016. 1, 2

[50] Aliaksandra Shysheya, John Bronskill, Massimiliano Patacchiola, Sebastian Nowozin, and Richard E. Turner. Fit: Parameter efficient few-shot transfer learning for personalized and federated image classification. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023. 2

[51] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th USENIX Security Symposium*, pages 2615–2632. USENIX Association, 2021. 2

[52] Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024. 1

[53] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. VL-ADAPTER: parameter-efficient transfer learning for vision-and-language tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 5217–5227, 2022. 2

[54] Yulong Tian, Fnu Suya, Anshuman Suri, Fengyuan Xu, and David Evans. Manipulating transfer learning for property inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 15975–15984. IEEE, 2023. 2

[55] Minh N. Vu, Truc D. T. Nguyen, Tre' R. Jeter, and My T. Thai. Analysis of privacy leakage in federated large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1423–1431. PMLR, 2024. 1, 2

[56] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004. 7

[57] Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. Pretrained models for multilingual federated learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1413–1421, 2022. 1

[58] Yeming Wen and Swarat Chaudhuri. Batched low-rank adaptation of foundation models. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024. 2

[59] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *International Conference on Machine Learning, ICML*, pages 23668–23684, 2022. 1

[60] Yuxin Wen, Leo Marchyok, Sanghyun Hong, Jonas Geiping, Tom Goldstein, and Nicholas Carlini. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. In *Annual Conference on Neural Information Processing Systems (Neurips)*, 2024. 1, 2

[61] Chulin Xie, De-An Huang, Wenda Chu, Daguang Xu, Chaowei Xiao, Bo Li, and Anima Anandkumar. Perada: Parameter-efficient federated learning personalization with generalization guarantees. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 23838–23848. IEEE, 2024. 1, 2

[62] Yuedong Yang, Hung-Yueh Chiang, Guihong Li, Diana Marculescu, and Radu Marculescu. Efficient low-rank backpropagation for vision transformer adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 2

[63] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov. See through gradients: Image batch recovery via gradinversion. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2021. 1, 2

[64] Guangsheng Zhang, Bo Liu, Huan Tian, Tianqing Zhu, Ming Ding, and Wanlei Zhou. How does a deep learning model architecture impact its privacy? A comprehensive study of privacy attacks on cnns and transformers. In *33rd USENIX Security Symposium, USENIX*, 2024. 1, 2

[65] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 586–595, 2018. 7, 15

[66] Xuechen Zhang, Mingchen Li, Xiangyu Chang, Jiasi Chen, Amit K. Roy-Chowdhury, Ananda Theertha Suresh, and Samet Oymak. Fedyolo: Augmenting federated learning with pretrained transformers. *Arxiv*, 2023. 1, 2, 3

[67] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL*, pages 9963–9977, 2023. 1, 2, 12

[68] J. Zhao, A. Sharma, A. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*, 2024. 2, 4, 5

[69] Joshua C. Zhao, Ahmed Roushdy Elkordy, Atul Sharma, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. The resource problem of using linear layer leakage attack in federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3974–3983. IEEE, 2023. 2

[70] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 12

Appendix

A. Ethical Considerations

Our work points to potential privacy threats that may occur when parameter-efficient fine-tuning (PEFT) is applied under the federated learning (FL) setup. Since (to the best of our knowledge) privacy concerns under PEFT based FL applications are under-explored, our observations suggest an important challenge that local data can be revealed if no additional defense mechanism is applied. Users involved in training might be oblivious to these risks. As a malicious server can deploy such attacks by merely poisoning model parameters, it is crucial to explore robust verification algorithms to examine the authenticity of the models received from the server. Furthermore, defense strategies such as differential privacy under the PEFT setting can prevent the server from observing the true local gradients with a small impact on utility. We hope that our work will motivate new research directions towards certifiable privacy, integrity, and authenticity guarantees for PEFT mechanisms.

B. Algorithm

We provide the pseudocode of our proposed attack, PEFTLeak, in Algorithm 1.

C. Additional Experiments

In this section, we provide additional experimental results for our proposed framework. Unless stated otherwise, for all the experiments below, we use a batch size 32, bottleneck dimension $r = 64$ and ViT-B/16 architecture in accordance with the experiments in Section 5.

C.1. Recovered Images for CIFAR-10

In Fig. 9, we demonstrate the recovery of a batch of 32 images from the gradient for the CIFAR-10 dataset. As we observe, 106-out-of-128 image patches, i.e., 82.8% of the patches are recovered.

C.2. Comparison with the Optimization-Based Baseline

We now describe the details of the optimization-based gradient inversion attack baseline. To the best of our knowledge, there are no successful optimization-based attack baselines under the PEFT setting. Reference [67] studied the performance of the optimization-based attack from [70] for PEFT and observed that it was not successful under the PEFT setup. Attack from [20] improves over [70] in terms of the reconstruction performance by taking the direction of the gradient into consideration. Essentially, the goal is to find a batch of images $\mathbf{X}$ that minimize the cosine distance between the true gradient and the predicted gradient,

Algorithm 1: PEFTLeak

```
Input: Pretrained model $\mathbf{w}_F$, adapter parameters $\mathbf{w}_A$, adapter
       gradients $\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_A}$ of user $i$ (victim)
Output: Recovered patches $\mathbf{x}^{(t,m)}$ for $t \in [N], m \in [M]$ of
        user $i$, where $N$ is the total number of patches and $M$ is
        the number of images in the batch
   // Server: Poisoning pretrained model, $\mathbf{w}_F$
   // (Position encoding vectors)
1  Select $\mathbf{E}_{pos}^{(n)} \sim \mathcal{N}(0, \sigma)$ for $n \in \{0, \ldots, N\}$
   // (Linear embedding matrix)
2  Set $\mathbf{E}$ in (6) to $0.5\mathbf{I}_D$
   // (MSA layer parameters)
3  Set $\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h = \mathbf{I}_{D_h \times D_h}$ for head $h \in [L]$   ▷ Equation (9)
4  Set $\mathbf{b}_Q^h, \mathbf{b}_K^h, \mathbf{b}_V^h = \mathbf{0}$ for head $h \in [L]$   ▷ Equation (10)
5  Set $\mathbf{W}_{MSA} = \mathbf{I}_{D \times D}$   ▷ Section 4.2
   // (MLP layer parameters)
6  Design weights $\mathbf{W}_{MLP,1}, \mathbf{W}_{MLP,2}$ according to (28), (29)
7  Design biases $\mathbf{b}_{MLP,1} = \gamma \mathbf{1}_{4D}, \mathbf{b}_{MLP,2} = -\gamma \mathbf{1}_D$   ▷ Section 4.5
   // (LN1 and LN2 layer parameters)
8  Set weights $\mathbf{w}_{LN1}, \mathbf{w}_{LN2} = \sigma \mathbf{1}_D$   ▷ Sections 4.1, 4.4
9  Set biases $\mathbf{b}_{LN1}, \mathbf{b}_{LN2}$ to $\mathbf{0}_D$   ▷ Sections 4.1, 4.4
10 Send $\mathbf{w}_F$ to the users   ▷ sent once prior to training
   // Server: Poisoning global adapter, $\mathbf{w}_A$
11 Set weights in down-projection layer to $\mathbf{E}_{pos}^{(t)}$ for target position
     $t \in [N]$   ▷ Section 4.3
12 Design biases in down-projection layer according to (19)
13 Set weights and biases in up-projection layer to 0   ▷ Section 4.3
14 Send $\mathbf{w}_A$ to the users   ▷ sent in each training round
   // User $i$: Local training
15 Compute loss $\mathcal{L}_i(\mathbf{w}_F, \mathbf{w}_A)$ for batch of images
16 Compute gradient $\frac{\partial \mathcal{L}_i}{\mathbf{w}_A}$
17 Send $\frac{\partial \mathcal{L}_i}{\mathbf{w}_A}$ to the server   ▷ sent in each training round
   // Server: Reconstruction from gradients
18 Recover embeddings $\mathbf{y}^{(t,m)}$ for $t \in [N], m \in [M]$   ▷ Equation (24)
19 Recover patch $\mathbf{x}^{(t,m)}$ for $t \in [N], m \in [M]$   ▷ Equation (25)
20 Return $\mathbf{x}^{(t,m)}$ for $t \in [N], m \in [M]$   ▷ recovered patches
```

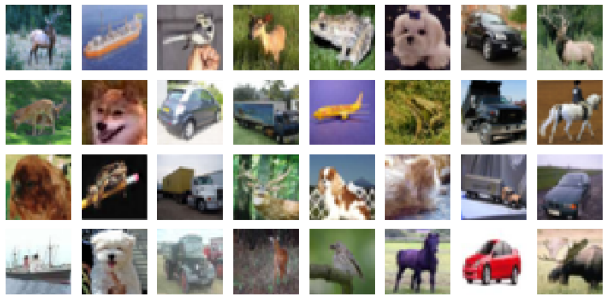
$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \mathcal{F}(\mathbf{X}) \tag{30}$$

such that,

$$\mathcal{F}(\mathbf{X}) \triangleq 1 - \frac{\langle \Delta \mathbf{g}, \Delta \mathbf{g}^{pred} \rangle}{||\Delta \mathbf{g}|| ||\Delta \mathbf{g}^{pred}||} + TV(\mathbf{X}) \tag{31}$$

where $\Delta \mathbf{g}$ is the actual gradient received from the victim user, $\Delta \mathbf{g}^{pred}$ is the predicted gradient from training on dummy images. The total variation regularization TV$(\cdot)$ is used as a standard image prior to ensure the smoothness of the recovered image. We note that this attack considers adversaries with limited capability, who do not adopt any malicious tampering with the protocol, such as changing the model parameters or architecture.

We applied this attack to our PEFT setting and studied how well this gradient matching algorithm performs


(a) Original images

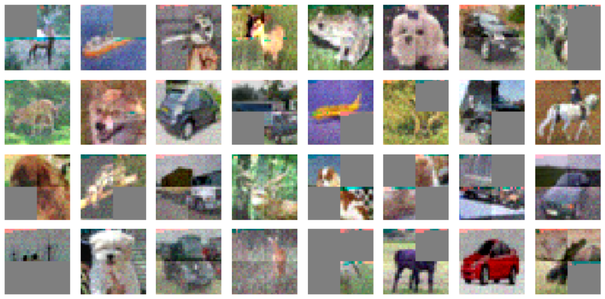
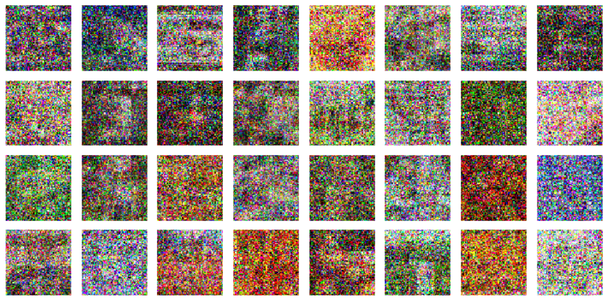

(b) Recovered

Figure 9. CIFAR-10 (recovered images for a batch of 32 images).


(a) Recovered from [20]

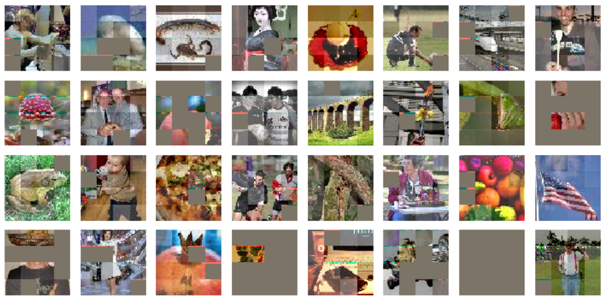
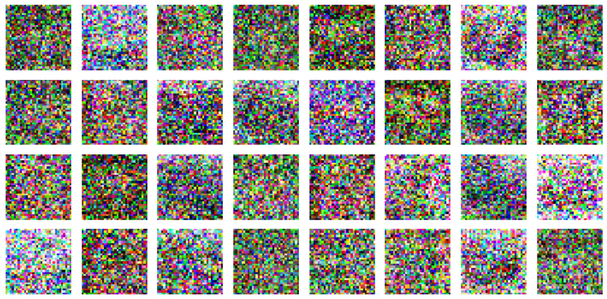

(b) Recovered (PEFTLeak)

Figure 10. Comparison with optimization-based benchmark from [20] (TinyImageNet).


(a) Recovered from [20]

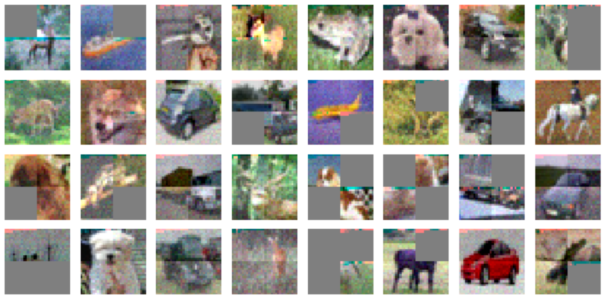

(b) Recovered (PEFTLeak)

Figure 11. Comparison with optimization-based benchmark from [20] (CIFAR-10).

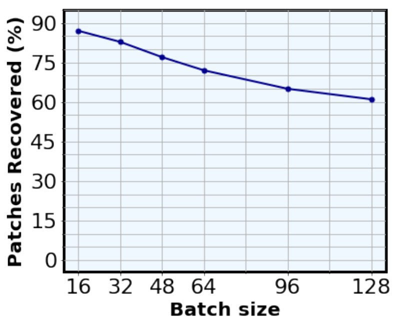
Architecture	ViT-B/16	ViT-L/16	ViT-B/32	
% **Patches recovered**	81	81	20.2 (naive)	79.6 (improved)

Table 3. Reconstruction for a batch of 32 images (TinyImageNet).

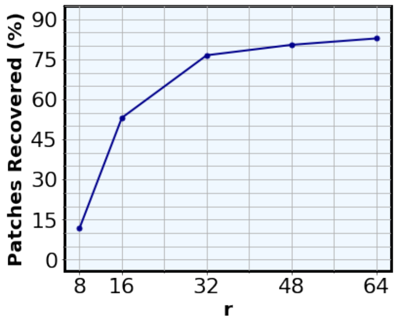
by leveraging the adapter gradients only. For this, we run the experiments for the images in Figs. 5a (in our main paper) and 9a from TinyImageNet and CIFAR-10 datasets (CIFAR-100 results were already provided in Fig. 8 in our main paper.) We demonstrate our results in Figs. 10 and 11, where we present the images reconstructed by the optimization-based attack vs. PEFTLeak. As we observe from Figs. 10 and 11, the optimization-based attack fails to reconstruct any of the images in the batch whereas PEFTLeak recovers most of the images with high fidelity.

C.3. Different Model Architectures

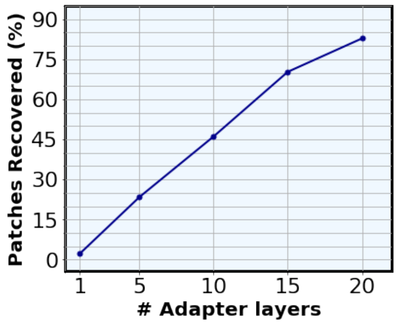
Table 3 shows our results for ViT-L/16 and ViT-B/32 with a batch size of 32. We observed that for a fixed embedding dimension D, more encoders (ViT-L/16) can speed up our attack. ViT-L/16 (24 encoders) recovers an image in just 2 rounds, compared to 4 rounds for ViT-B/16 (12 encoders). When the number of encoders is fixed, we observed an interesting relation between D and patch size P. In ViT-B/32, each $(P, P) = (32, 32)$ patch flattens to a $P^2C = 3072$-dimensional vector (C channels). If $D \geq P^2C$, as in ViT-B/16 ($P = 16, D = 768$) and ViT-L/16 ($P = 16, D = 1024$), all pixels can be recovered. In ViT-B/32, $D < P^2C$, limiting naive recovery to $D = 768$ pixels. A simple solution is then to recover an average pixel from each $(2, 2)$ region, yielding a lower resolution recon-



(a) Varying batch size

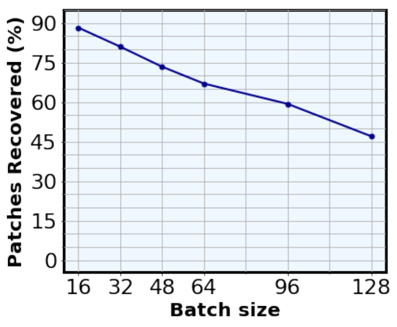


(b) Varying r

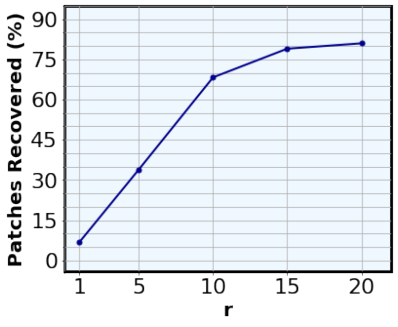


(c) Varying # adapter layers

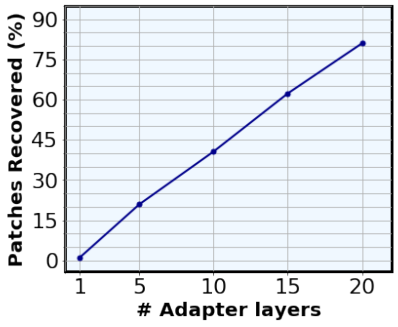
Figure 12. Percentage of patches recovered with varying batch size, bottleneck dimension, and number of adapter layers used within a single training round (CIFAR-10).



(a) Varying batch size



(b) Varying r



(c) Varying # adapter layers

Figure 13. Percentage of patches recovered with varying batch size, bottleneck dimension, and number of adapter layers used within a single training round (TinyImageNet).

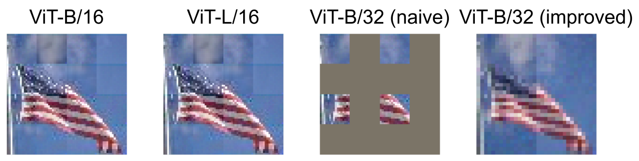


Figure 14. Recovered images from different model architectures.

struction. Fig. 14 illustrates this for a recovered sample.

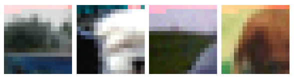
C.4. Ablation Study

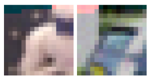
Varying batch size. We next demonstrate the reconstruction performance with varying batch size, bottleneck dimension and number of adapter layers for CIFAR-10 and TinyImageNet dataset (CIFAR-100 results were provided in Section 5 in our main paper). In Figs. 12a and 13a, we observe that even for batch sizes as large as $64, 96, 128$, a notable amount of the patches are recovered.

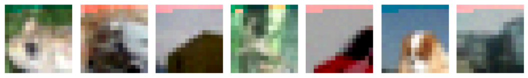
Varying bottleneck dimension. We next report the reconstruction rate for varying r, the bottleneck dimension within each adapter layer. Higher value of r implies that more neurons are available in each adapter layer that can be leveraged for reconstruction. In Figs. 12b, 13b, we observe that as r increases, more patches are recovered.

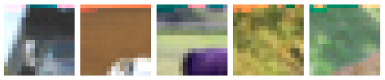
Benefits of using multiple adapter layers. For the experiments in Figs. 4b, 5b and 9b, we have allocated 5 adapter layers for the reconstruction of patches from each position. As mentioned in Section 5, images from CIFAR-10 and CIFAR-100 datasets are divided into 4 patches. Therefore, for 4 patches, we utilize 20 adapter layers in total within a single training round. For TinyImageNet, each image is divided into 16 patches. The server aims to recover 4 patches from 20 adapter layers per training round. For this, the server sends malicious adapter parameters to recover patches from 4 target positions by leveraging the adapter gradients received from the user in each round. Hence, all the patches are recovered over 4 training rounds. In this regard, we next demonstrate the benefit of using multiple adapter layers in terms of attack success. In Figs. 12c and 13c, we report the percentage of patches recovered per training round. As we observe, more patches are recovered as more adapter layers are being utilized.

We further provide the illustration of the recovered patches in Figs. 15-20. Figs. 15, 16, and 17, demonstrate the recovery of the patches from the first position, i.e., top-left patch of the images from Figs. 9a, 4a and 5a. As de-


(a) 1st layer


(b) 2nd layer


(c) 3rd layer


(d) 4th layer

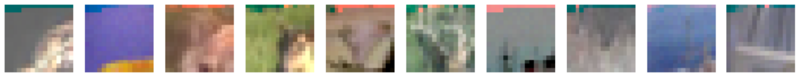
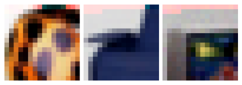
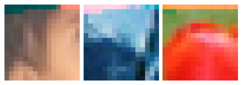
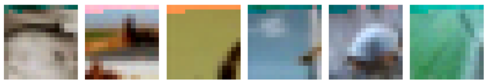

(e) 5th layer

Figure 15. Recovered patches from the first position using multiple adapter layers (CIFAR-10).


(a) 1st layer

(b) 2nd layer


(c) 3rd layer


(d) 4th layer

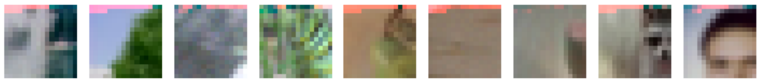

(e) 5th layer

Figure 16. Recovered patches from the first position using multiple adapter layers (CIFAR-100).

scribed in Section 4, the weight and bias parameters in the adapter layers are designed such that patches from the target position can be recovered by leveraging the adapter gradients. Patches from all other positions will be filtered out by the activation function. We observe that by utilizing multiple adapter layers, we recover most of the target patches for this position. Moreover, in Figs. 18, 19, and 20, we demonstrate the recovered patches from the same target position for $r = 8$ in comparison with $r = 64$. As we observe, more patches are retrieved from the adapter gradients when r is increased from 8 to 64.

C.5. Robustness Against Defense Mechanisms

Fig. 21 presents the attack performance against potential defense mechanisms, including noise addition [1], pruning (top-K) [4, 35] and stochastic quantization [3]. Attack performance is measured in terms of average LPIPS score [65] between recovered and ground-truth images. In Fig. 21a, we vary the standard deviation of added Gaussian noise with respect to the gradient norm.

C.6. Attack to FedAvg

We next consider the FedAvg setup, where each user performs multiple rounds of local training before sending the gradient to the server. We again leverage the activation structure from [17] (proposed for the FedAvg setting) in the down-projection layer within each adapter. At each global training round, each user performs local training for 5 epochs, and sends the local gradient to the server. We demonstrate the reconstructed images in Fig. 22b, and observe that image patches can be recovered with high fidelity.

C.7. Reconstruction on Additional Images

In Fig. 23, we further demonstrate the reconstructed images from a larger batch size. For this, we consider the images from CIFAR-100 dataset for a batch of size 64. As we observe in Fig. 23, successful reconstruction of 75% of the patches is obtained from the adapter gradients.

σ	1	2	3	5	10
Gaussian	12	30.4	52.3	77.3	85.9
Laplacian	12.5	35.1	57.8	70	92.9

Table 4. % patches recovered with different σ and distributions.

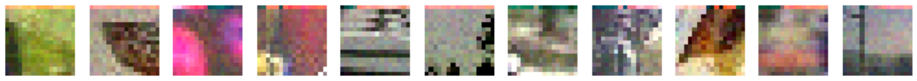
C.8. Attack Detectability

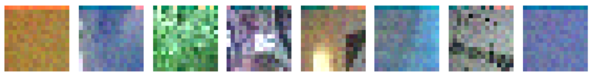
Our attack leverages the fact that users implicitly trust the server for the pretrained model and fine-tuning parameters. However, our malicious design may cause the users to question the integrity of the server. As described in Section 4.3, to recover patches from a target position, our attack sets the weight rows to be identical in the first linear layer of the adapter modules. To make this design more stealthy, the server can introduce non-malicious weight rows and biases in-between. Moreover, for position encoding, any distribution can be used if they meet the criteria outlined in (12) and Section 4.1. Table 4 shows our results with lower standard deviation σ across multiple distributions to improve stealth. Even with a σ as small as 3, our attack can recover 57.8% of the patches (batch size 32, CIFAR-100).

C.9. Reconstructed Images from the ImageNet Dataset

In Fig. 24, we present sample images from ImageNet.

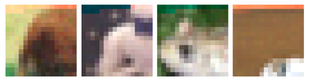
(a) 2nd layer


(b) 3rd layer


(c) 4th layer

(d) 5th layer

Figure 17. Recovered patches from the first position using multiple adapter layers (TinyImageNet). None of the patches are recovered from the 1st layer gradients.


(a) $r = 8$

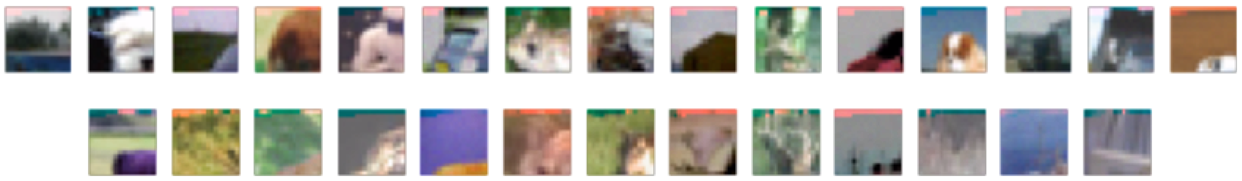
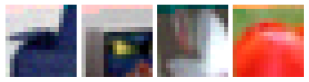

(b) $r = 64$

Figure 18. Impact of bottleneck dimension r on patch reconstruction (CIFAR-10).


(a) $r = 8$

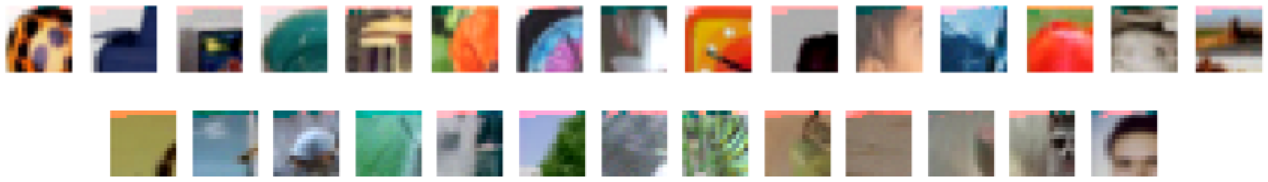
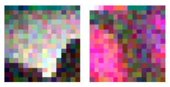

(b) $r = 64$

Figure 19. Impact of bottleneck dimension r on patch reconstruction (CIFAR-100).


(a) $r = 8$

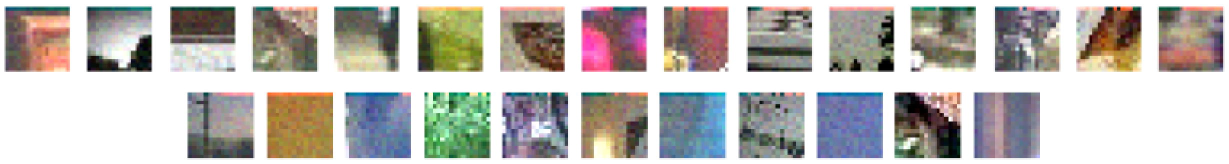
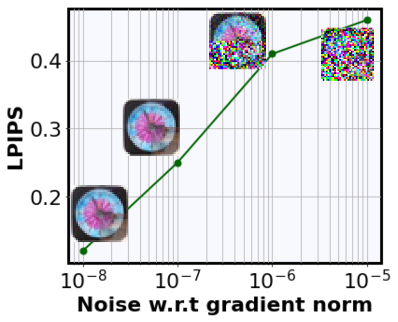
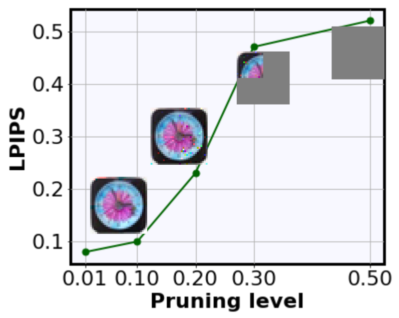

(b) $r = 64$

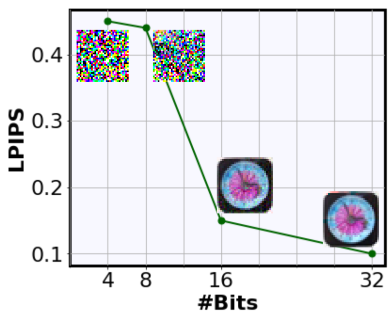
Figure 20. Impact of bottleneck dimension r on patch reconstruction (TinyImageNet).



(a) Noise

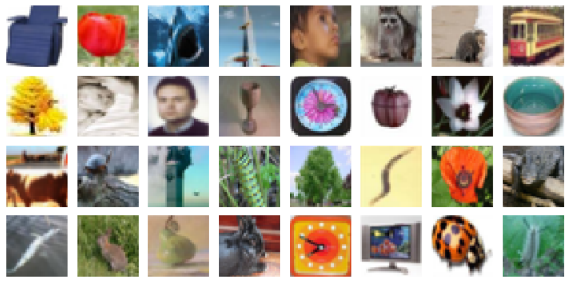


(b) Pruning

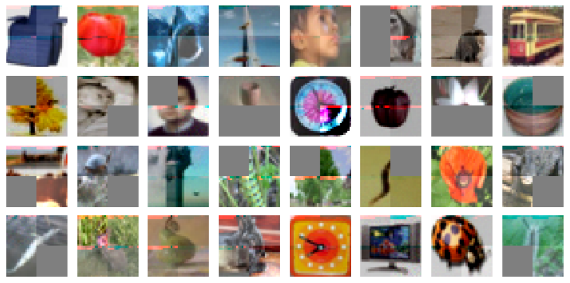


(c) Quantization

Figure 21. Performance against mitigation strategies (CIFAR-100, batch size 32). Lower LPIPS denotes better reconstruction.

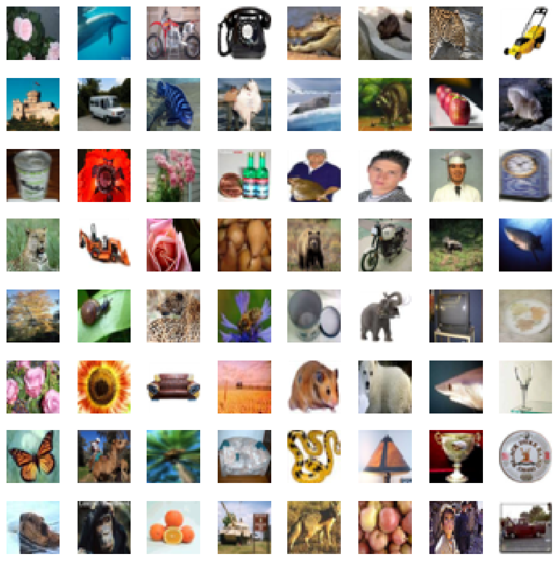


(a) Original images

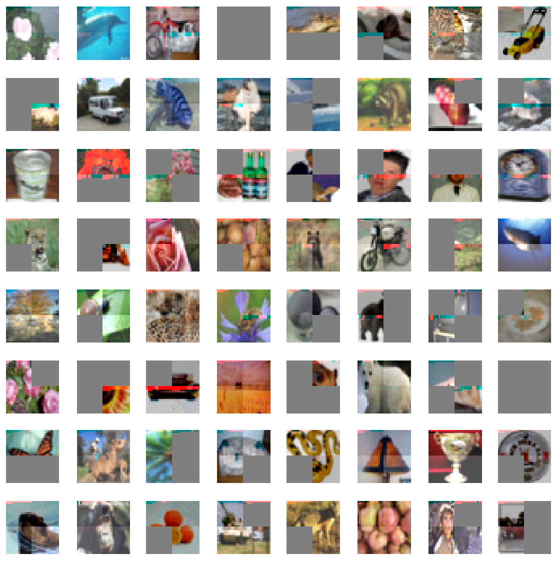


(b) Recovered

Figure 22. Recovered images for FedAvg with 5 local training rounds (CIFAR-100).

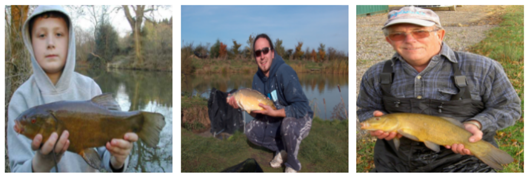


(a) Original images

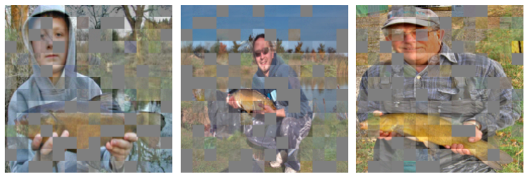


(b) Recovered (PEFTLeak)

Figure 23. Recovered images for a batch of size 64 (CIFAR-100).



(a) Ground-truth



(b) Recovered

Figure 24. Recovered images (ImageNet).