Gradient inversion Attacks on Parameter-Efficient Fine-Tuning

Hasin Us Sami, Swapneel Sen, Amit K. Roy-Chowdhury, Srikanth V. Krishnamurthy, Başak Güler

Department of Electrical and Computer Engineering

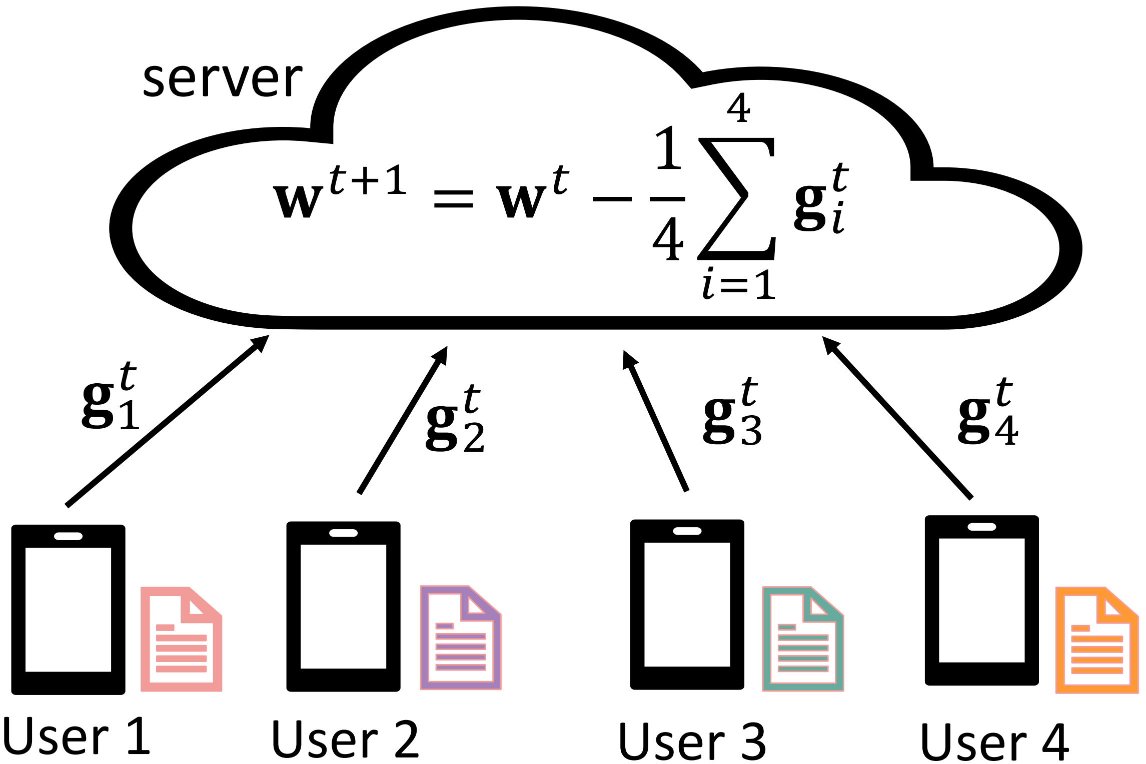
University of California, Riverside

hsami003@ucr.edu, ssen010@ucr.edu, amitrc@ece.ucr.edu,

krish@cs.ucr.edu, bguler@ece.ucr.edu

Federated Learning

Learn a machine learning model using data stored locally across wireless users.



Fine-tuning of Pretrained Models

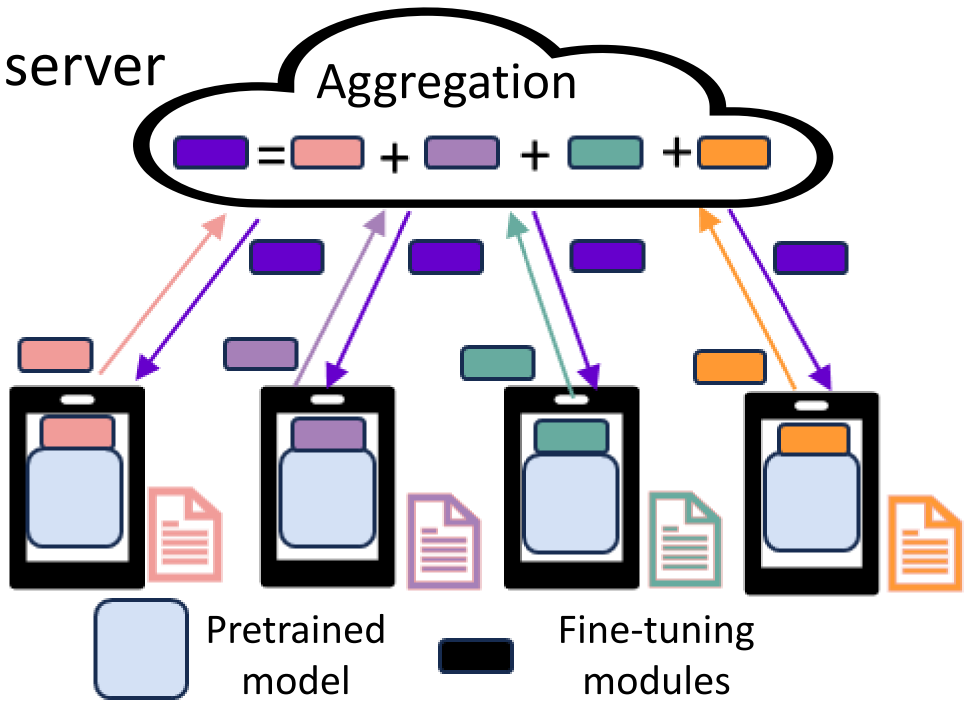
- Gained attention for various downstream tasks
- Recent works extend fine-tuning in federated learning

Limitations

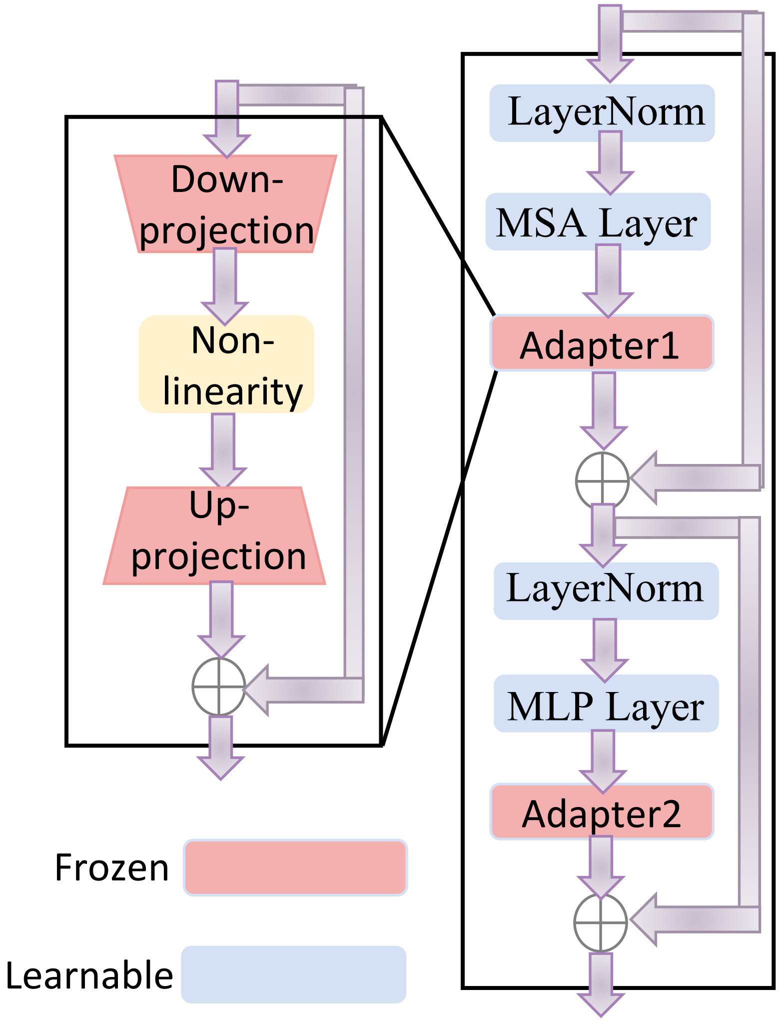
- Prohibitive computational infrastructure and bandwidth
- Users with low-resources abort from the protocol
- Causes bias in the global model

Parameter-Efficient Fine-tuning (PEFT)

- Pretrained model is kept frozen
- Only a small number of lightweight modules are trained
- Marked reduction in resource consumption and training latency



Vision Transformer (ViT) with Adapters



- A ViT encoder consists of LayerNorm, MSA and MLP layers [Dosovitskiy et al.'21]
- An adapter module is inserted after each MSA/MLP layer
- Each adapter consists of down-projection and up-projection layers with ReLU in-between [Houlsby et al.'19]
- Down-projection projects the input dimension, D to a lower dimension $r \ll D$

Gradient Inversion Attacks

- Can recover training samples from the shared gradients
- Attack on full fine-tuning [Feng and Tramèr, ’24]
  - Malicious pretrained model
  - Leverages MLP layer gradients to recover data
- Attack under PEFT remains underexplored
  - No access to MLP layer gradients
  - Can only access adapter gradients (limited information)

This Work (PEFTLeak)

- First gradient inversion attack on PEFT
- Maliciously designed pretrained model and adapter modules
- Design MSA, MLP, LayerNorm layers as identity mapping
- Captures data inside the adapter gradients
- Leverage gradients from multiple adapters to recover data

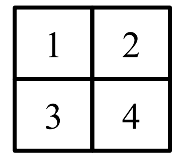
This Work (PEFTLeak)

- Consider an adversarial server
  - Modify the pretrained model
  - Modify the global adapter parameters
  - Send the pretrained model once prior to training
  - Send global adapters in each training round

This Work (PEFTLeak)

- Let us assume M images in the batch
- Each image is divided into N patches





(N=4 patches)

Flattening of patches

$\mathbf{x}^{(n,m)} \in \mathbb{R}^D$ for patch $n \in N$, image $m \in M$

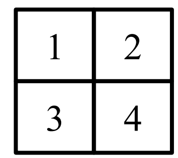
Linear Projection and position encoding

$$\mathbf{y}^{(n,m)} = \mathbf{E}\mathbf{x}^{(n,m)} + \mathbf{E}_{pos}^{(n)} = \mathbf{x}_{map}^{(n,m)} + \mathbf{E}_{pos}^{(n)} \in \mathbb{R}^D$$

Class token $\mathbf{y}^{(0,m)}$

This Work (PEFTLeak)

- Let us assume M images in the batch
- Each image is divided into N patches



(N=4 patches)

Flattening of patches

$\mathbf{x}^{(n,m)} \in \mathbb{R}^D$ for patch $n \in N$, image $m \in M$

Linear Projection and position encoding

$\mathbf{E} = 0.5\mathbf{I}_{D\times D}$, $\mathbf{E}_{pos}^{(n)} \sim \mathcal{N}(0, \sigma)$ with $\sigma = 10$

$$\mathbf{y}^{(n,m)} = \mathbf{E}\mathbf{x}^{(n,m)} + \mathbf{E}_{pos}^{(n)} = \mathbf{x}_{map}^{(n,m)} + \mathbf{E}_{pos}^{(n)} \in \mathbb{R}^D$$

Class token $\mathbf{y}^{(0,m)}$

This Work (PEFTLeak)

- Let us assume M images in the batch
- Each image is divided into N patches

1	2
3	4

(N=4 patches)

Flattening of patches

$\mathbf{x}^{(n,m)} \in \mathbb{R}^D$ for patch $n \in N$, image $m \in M$

Linear Projection and position encoding

$\mathbf{y}^{(n,m)} = \mathbf{E}\mathbf{x}^{(n,m)} + \mathbf{E}_{pos}^{(n)} = \mathbf{x}_{map}^{(n,m)} + \mathbf{E}_{pos}^{(n)} \in \mathbb{R}^D$

1) $(\mathbf{E}_{pos}^{(t)})^T \mathbf{E}_{pos}^{(t)} >> (\mathbf{E}_{pos}^{(t)})^T \mathbf{E}_{pos}^{(n)}$ for $n \neq t$
2) $std(\mathbf{y}^{(n,m)}) \approx \sigma$
3) $mean(\mathbf{y}^{(n,m)}) \approx 0$

Class token $\mathbf{y}^{(0,m)}$

This Work (PEFTLeak)

- Let us assume M images in the batch
- Each image is divided into N patches

1	2
3	4

(N=4 patches)

Flattening of patches

$\mathbf{x}^{(n,m)} \in \mathbb{R}^D$ for patch $n \in N$, image $m \in M$

Linear Projection and position encoding

$\mathbf{y}^{(n,m)} = \mathbf{E}\mathbf{x}^{(n,m)} + \mathbf{E}_{pos}^{(n)} = \mathbf{x}_{map}^{(n,m)} + \mathbf{E}_{pos}^{(n)} \in \mathbb{R}^D$

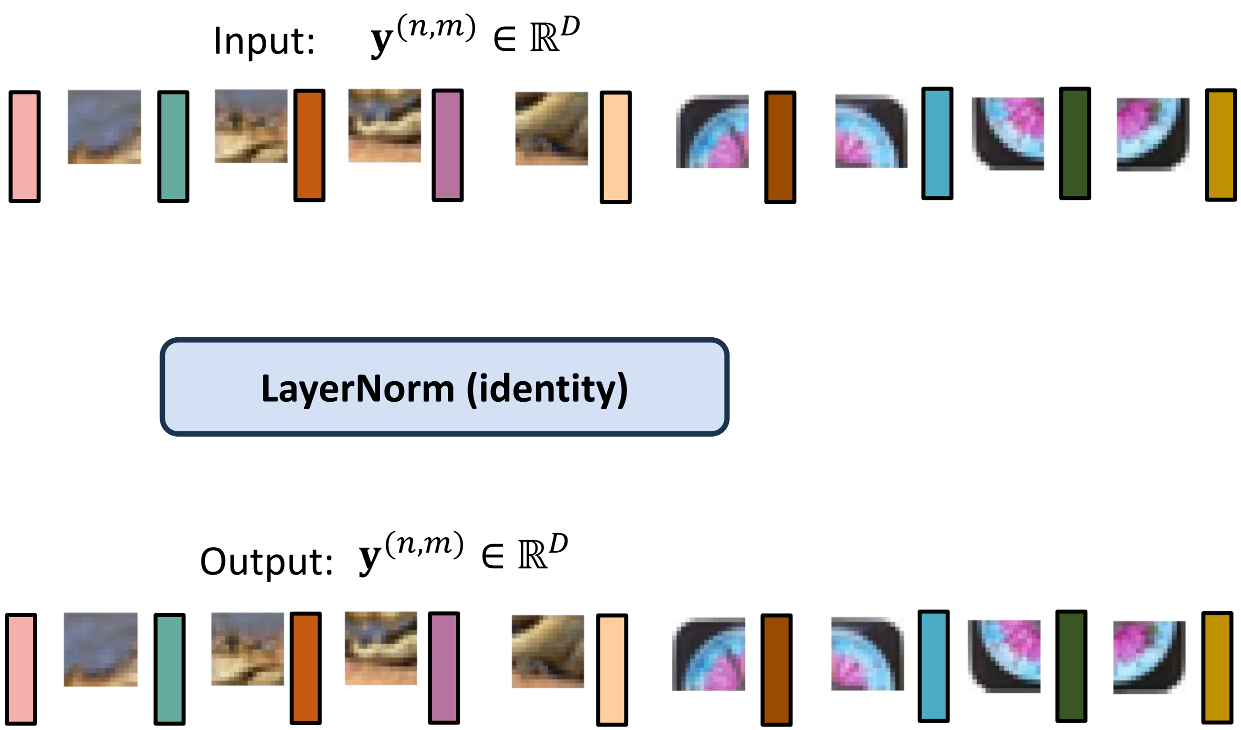
1) $(\mathbf{E}_{pos}^{(t)})^T \mathbf{E}_{pos}^{(t)} >> (\mathbf{E}_{pos}^{(t)})^T \mathbf{E}_{pos}^{(n)}$ for $n \neq t$
2) $std(\mathbf{y}^{(n,m)}) \approx \sigma$
3) $mean(\mathbf{y}^{(n,m)}) \approx 0$

Makes self-attention an identity mapping

Makes LayerNorm an identity function

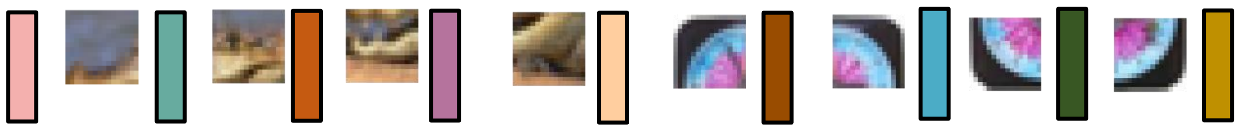
Class token $\mathbf{y}^{(0,m)}$

This Work (PEFTLeak)



This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



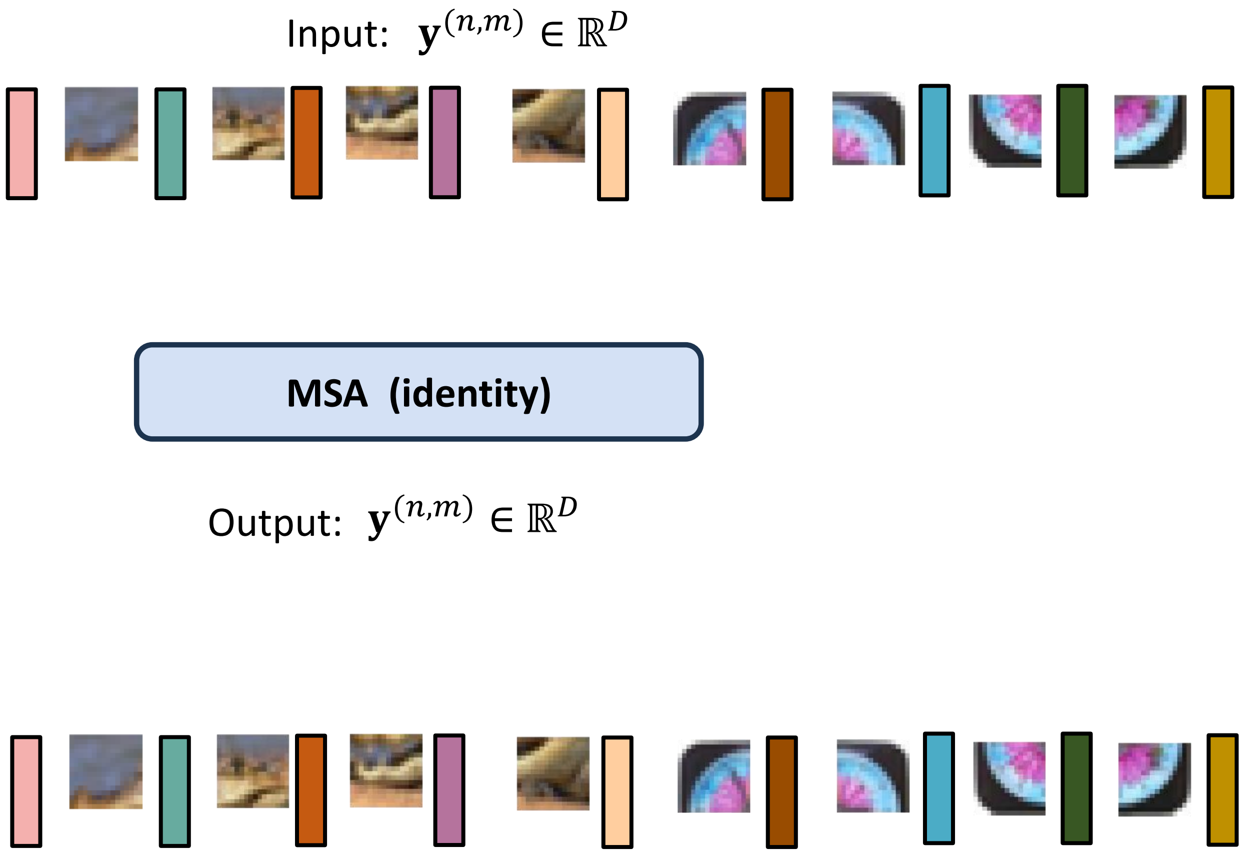
LayerNorm (identity)

weight $\mathbf{w}_{LN}=\sigma\mathbf{1}_D$, bias $\mathbf{b}_{LN}=\mathbf{0}$

Output: $\frac{\boldsymbol{y}^{(n,m)}-\mathbf{0}}{\boldsymbol{\sigma}} \odot \boldsymbol{\sigma} \approx \boldsymbol{y}^{(n,m)}$

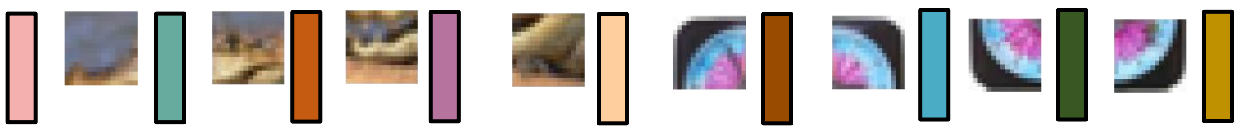
Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

This Work (PEFTLeak)



This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^{D}$



MSA (identity)

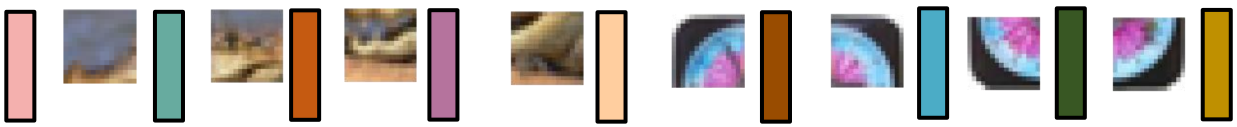
$D_h = D/L$

- Consists of L heads
- weight $\mathbf{W}_{\mathrm{Q}}^{h} = \mathbf{W}_{\mathrm{K}}^{h} = \mathbf{W}_{\mathrm{V}}^{h} = \mathbf{I}_{D_h \times D_h}$
- bias $\mathbf{b}_{\mathrm{Q}}^{h} = \mathbf{b}_{\mathrm{K}}^{h} = \mathbf{b}_{\mathrm{V}}^{h} = \mathbf{0}$ for head $h \in [L]$
- Define $(\boldsymbol{y}^{(n,m)})_h \cong \boldsymbol{y}^{(n,m)}[hD_h{:}(h+1)D_h]$
- Define query, key, value as $\mathbf{Q}^h$, $\boldsymbol{K}^h$, $\boldsymbol{V}^h$
- $\mathbf{Q}^h = \boldsymbol{K}^h = \boldsymbol{V}^h = [(\boldsymbol{y}^{(0,m)})_h \cdots [(\boldsymbol{y}^{(N,m)})_h]$

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^{D}$

This Work (PEFTLeak)

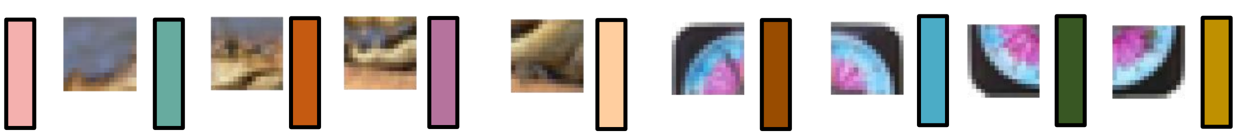
Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



MSA (identity)

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

- $\boldsymbol{A}^h = softmax((\mathbf{Q}^h)^{\boldsymbol{T}}\mathbf{K}^h/D_h)$
  $\cong \mathbf{I}_{(N+1)\times(N+1)}$
- Output projection matrix, $\mathbf{W}_{MSA} = \mathbf{I}_{D\times D}$
- Output: $[\boldsymbol{A}^1(\boldsymbol{V}^1)^T \ldots \boldsymbol{A}^L(\boldsymbol{V}^L)^T]\,\mathbf{W}_{MSA}$
  $= [\boldsymbol{y}^{(0,m)} \ldots \boldsymbol{y}^{(N,m)}]^T$



This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

$(\mathbf{E}_{pos}^{(t)})^T \, \mathbf{E}_{pos}^{(t)} >> (\mathbf{E}_{pos}^{(t)})^T \, \mathbf{E}_{pos}^{(n)}$ for $n \neq t$

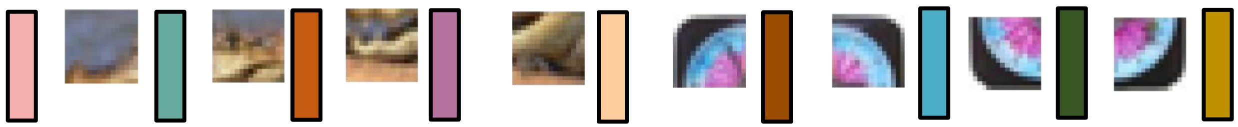
MSA (identity)

- $\boldsymbol{A}^h = softmax((\mathbf{Q}^h)^{\boldsymbol{T}} \mathbf{K}^h / D_h)$
  $\cong \mathbf{I}_{(N+1)\times(N+1)}$
- Output projection matrix, $\mathbf{W}_{MSA} = \mathbf{I}_{D\times D}$
- Output: $[\boldsymbol{A}^1(\boldsymbol{V}^1)^T \ldots \boldsymbol{A}^L(\boldsymbol{V}^L)^T] \, \mathbf{W}_{MSA}$
  $= [\boldsymbol{y}^{(0,m)} \ldots \boldsymbol{y}^{(N,m)}]^T$

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

This Work (PEFTLeak)

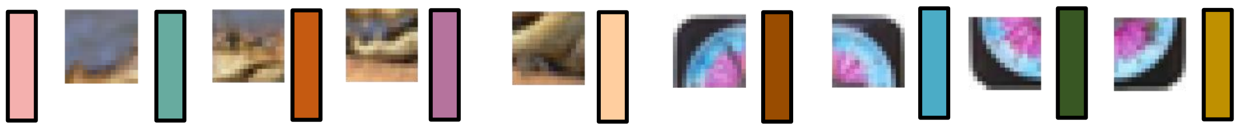
Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



Adapter1
(Gradients accessible to the attacker)

This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

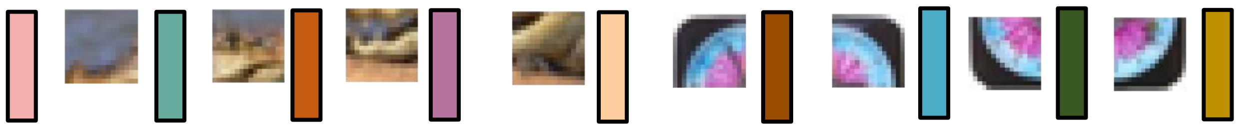


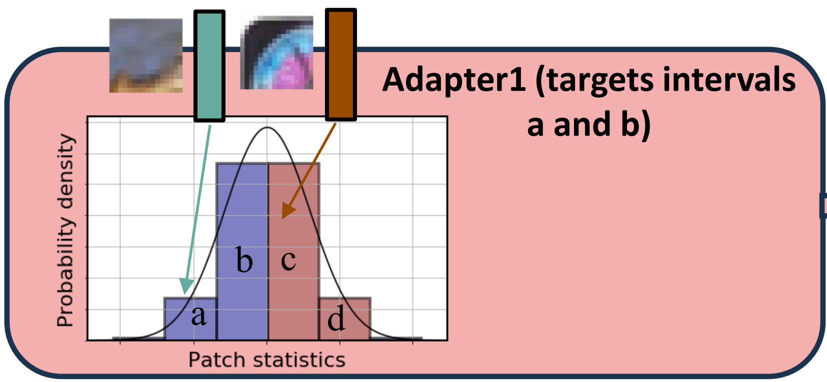
Adapter1
(Gradients accessible to the attacker)

- Target patches from position $t = 1$
- Leverage a public dataset [Fowl et al.'22]
- Estimate distribution of $(\mathbf{E}_{pos}^{(t)})^T \mathbf{x}_{map}^{(t,m)}$

This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



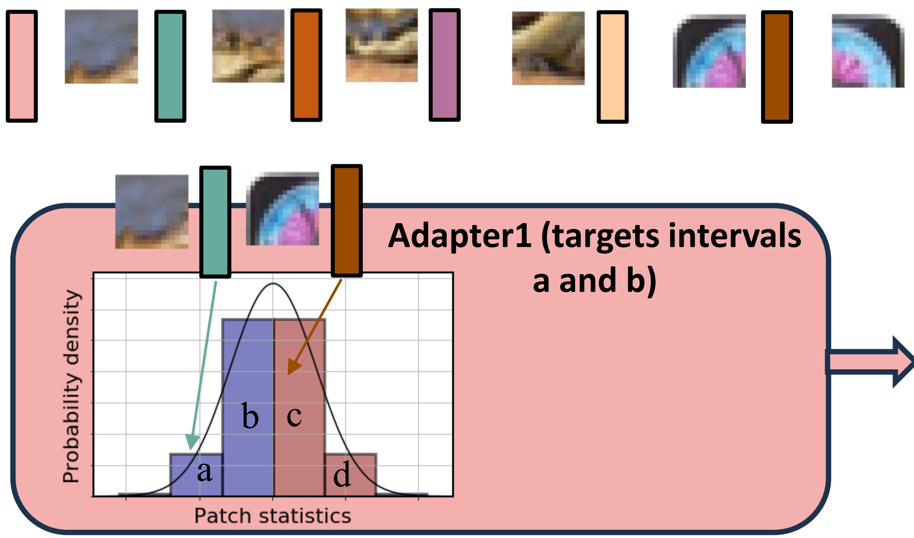


- Target patches from position $t = 1$
- Leverage a public dataset [Fowl et al.'22]
- Estimate distribution of $(\mathbf{E}_{pos}^{(t)})^T \, \mathbf{x}_{map}^{(t,m)}$
- Estimate c_j, c_{j+1} s.t

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

This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$

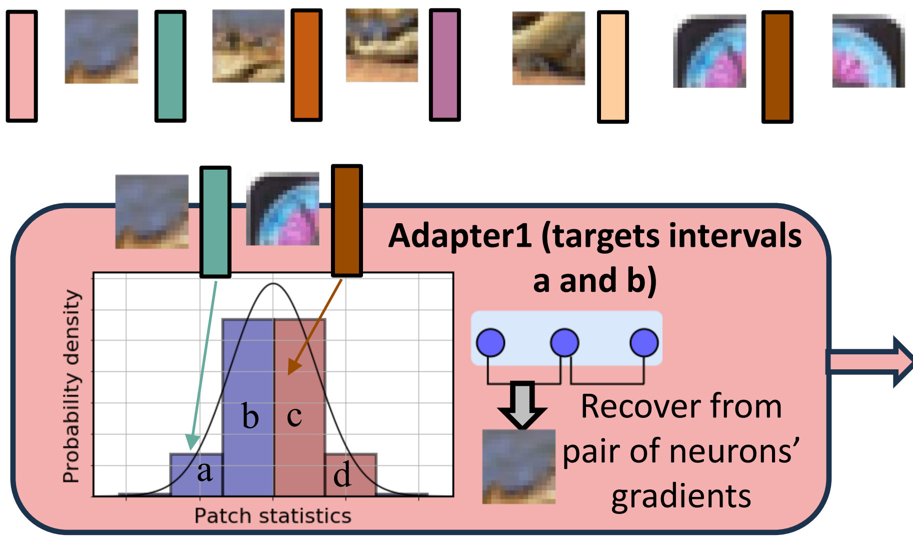


Down-projection:

- For neuron j,
  - set weight vector, $\mathbf{w}_j$ to $\mathbf{E}_{pos}^{(t)}$
  - set bias, b_j to $-(\mathbf{E}_{pos}^{(t)})^T \mathbf{E}_{pos}^{(t)} - c_j$
- Neuron j's output $(\mathbf{E}_{pos}^{(t)})^T \mathbf{x}_{map}^{(t,m)} - c_j > 0$
- Neuron j +1's output <0 (blocked by ReLU)

This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



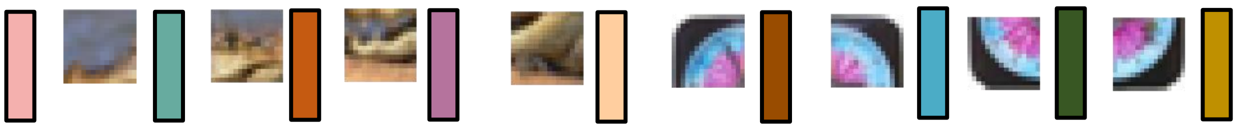
$\mathcal{L}_i$=Loss function

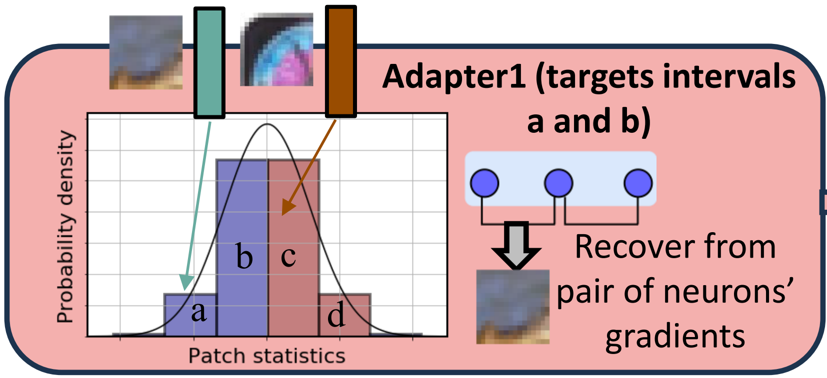
Down-projection:

- Recover $(\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_{j+1}})/(\frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_{j+1}}) = \mathbf{y}^{(t,m)}$
- Recover $\mathbf{x}^{(t,m)} = \mathbf{E}^{\dagger}(\mathbf{y}^{(t,m)} - \mathbf{E}_{pos}^{(t)})$

This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$





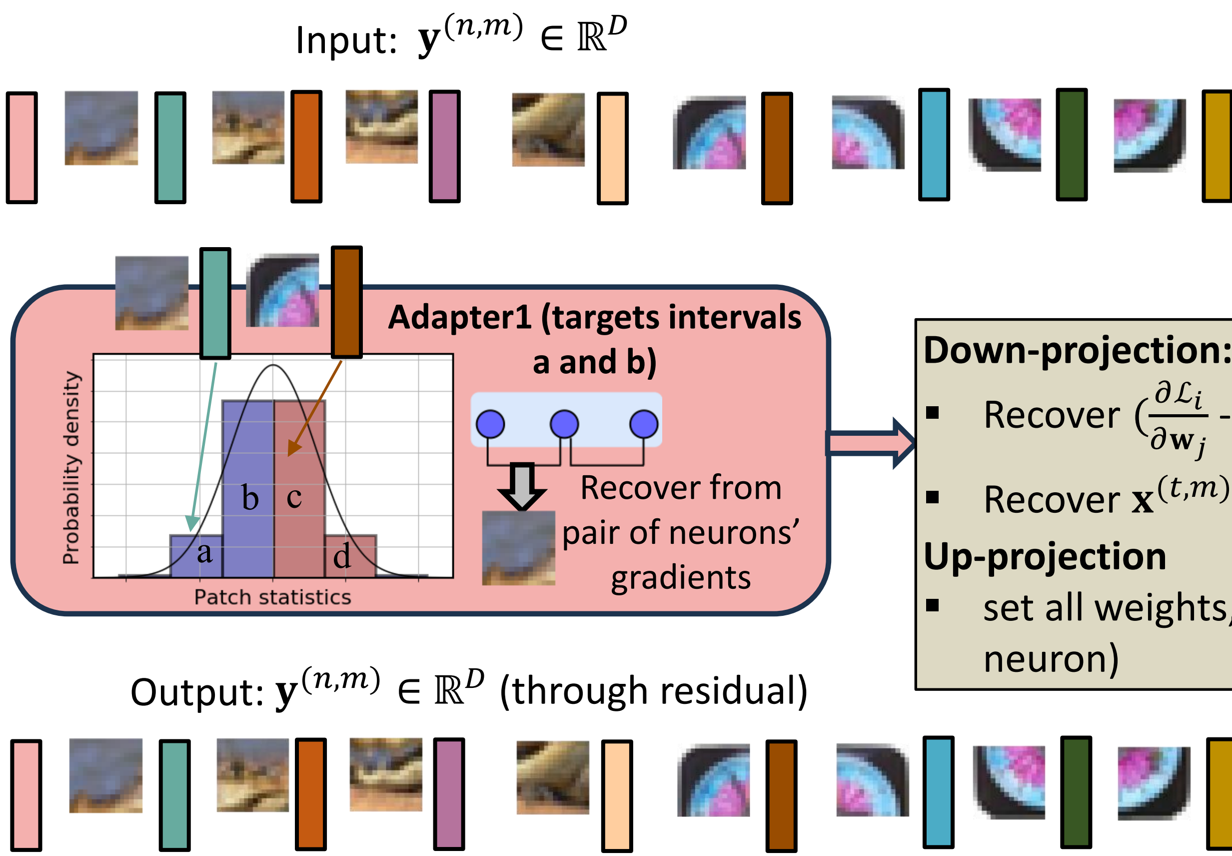
Down-projection:

- Recover $(\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_{j+1}})/(\frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_{j+1}}) = \mathbf{y}^{(t,m)}$
- Recover $\mathbf{x}^{(t,m)} = \mathbf{E}^{\dagger}(\mathbf{y}^{(t,m)} - \mathbf{E}_{pos}^{(t)})$

Up-projection

- set all weights, biases to 0 (except first neuron)

This Work (PEFTLeak)



Down-projection:

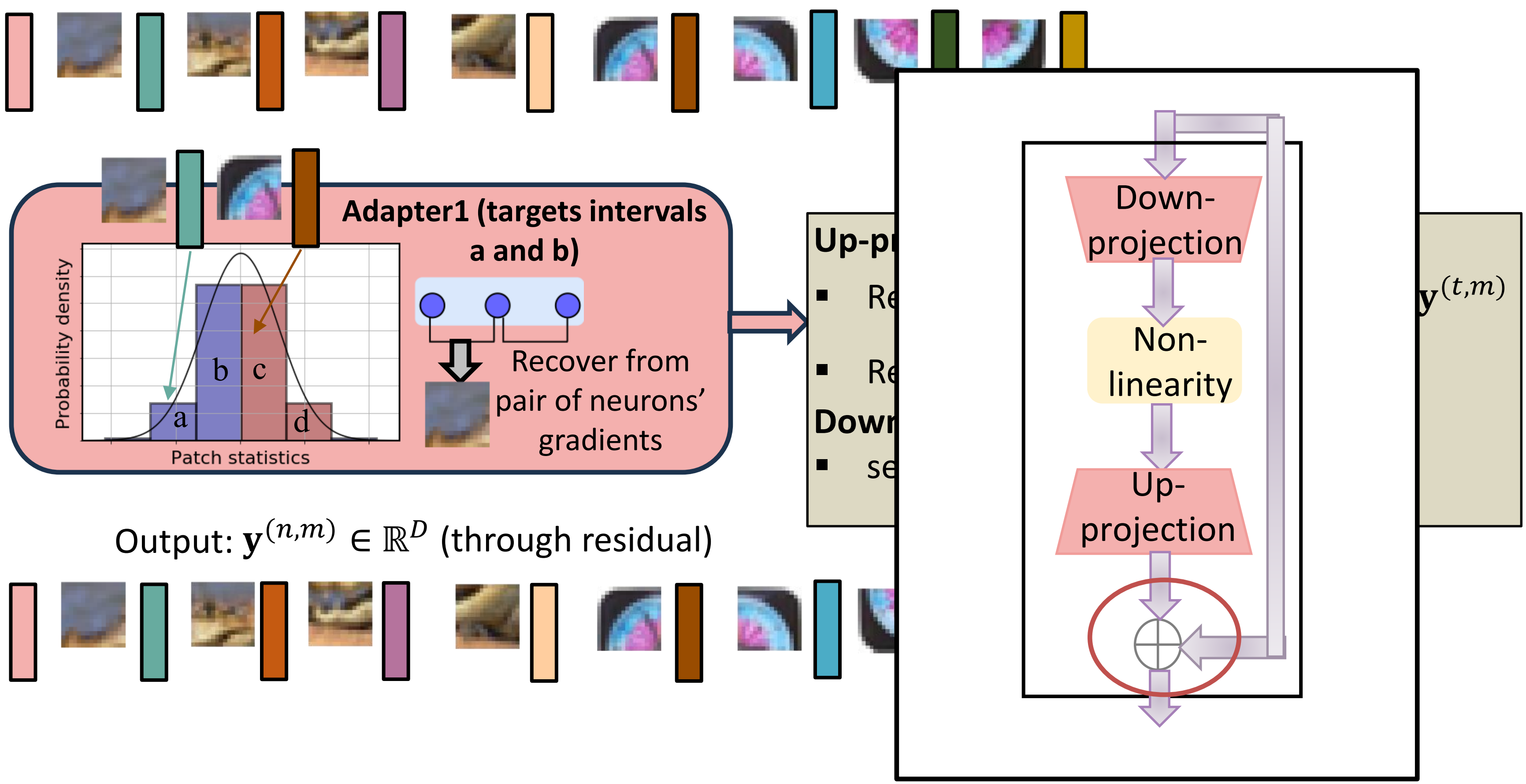
- Recover $(\frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathbf{w}_{j+1}})/(\frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_j} - \frac{\partial \mathcal{L}_i}{\partial \mathrm{b}_{j+1}}) = \mathbf{y}^{(t,m)}$
- Recover $\mathbf{x}^{(t,m)} = \mathbf{E}^{\dagger}(\mathbf{y}^{(t,m)} - \mathbf{E}_{pos}^{(t)})$

Up-projection

- set all weights, biases to 0 (except first neuron)

This Work (PEFTLeak)

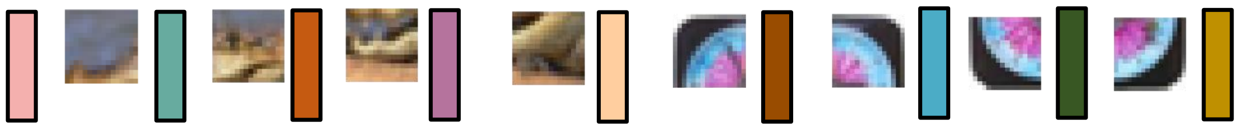
Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$ (through residual)

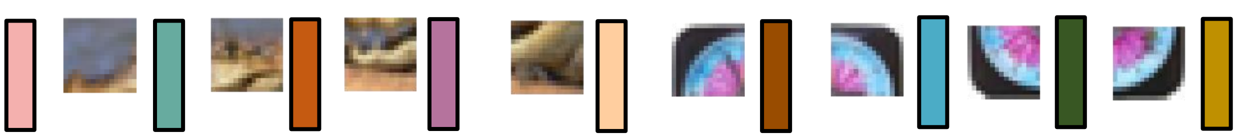
This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



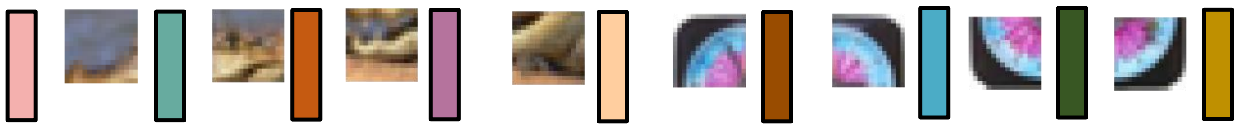
MLP (identity)

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



This Work (PEFTLeak)

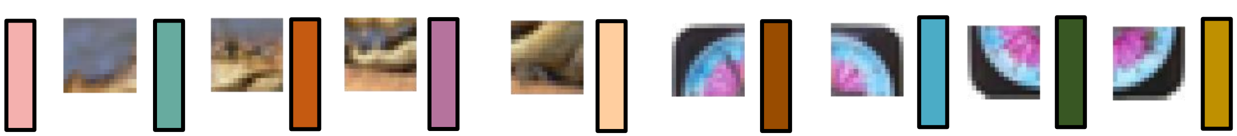
Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



MLP (identity)

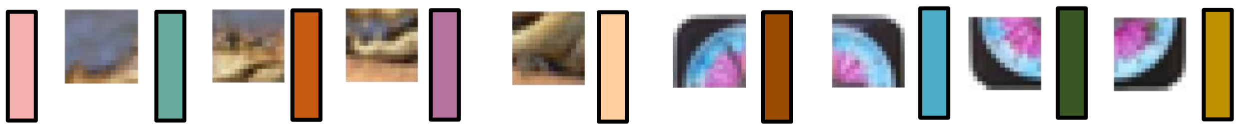
- Two linear layers MLP,1 and MLP,2 with GELU in-between
- Define element in row p and column q of matrix $\mathbf{W}$ as $\mathbf{W}[\mathrm{p},\mathrm{q}]$
- $\mathbf{W}_{MLP,1}[\mathrm{p},\mathrm{q}] = \begin{cases} 1 \text{ if } \mathrm{p}=\mathrm{q} \\ 0 \text{ otherwise} \end{cases}$, $\mathbf{b}_{MLP,1}=10^4\mathbf{1}_{4D}$
- $\mathbf{W}_{MLP,2}[\mathrm{p},\mathrm{q}] = \begin{cases} 1 \text{ if } \mathrm{p}=\mathrm{q} \\ 0 \text{ otherwise} \end{cases}$, $\mathbf{b}_{MLP,2}=-10^4\mathbf{1}_{D}$

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



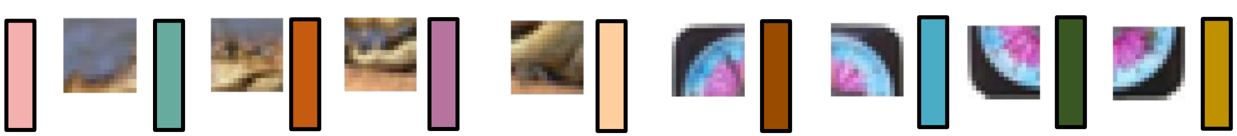
MLP (identity)

Output:

$$\mathbf{W}_{MLP,2}\big(GELU(\mathbf{W}_{MLP,1}\mathbf{y}^{(n,m)} + \mathbf{b}_{MLP,1}\big) + \mathbf{b}_{MLP,2}$$

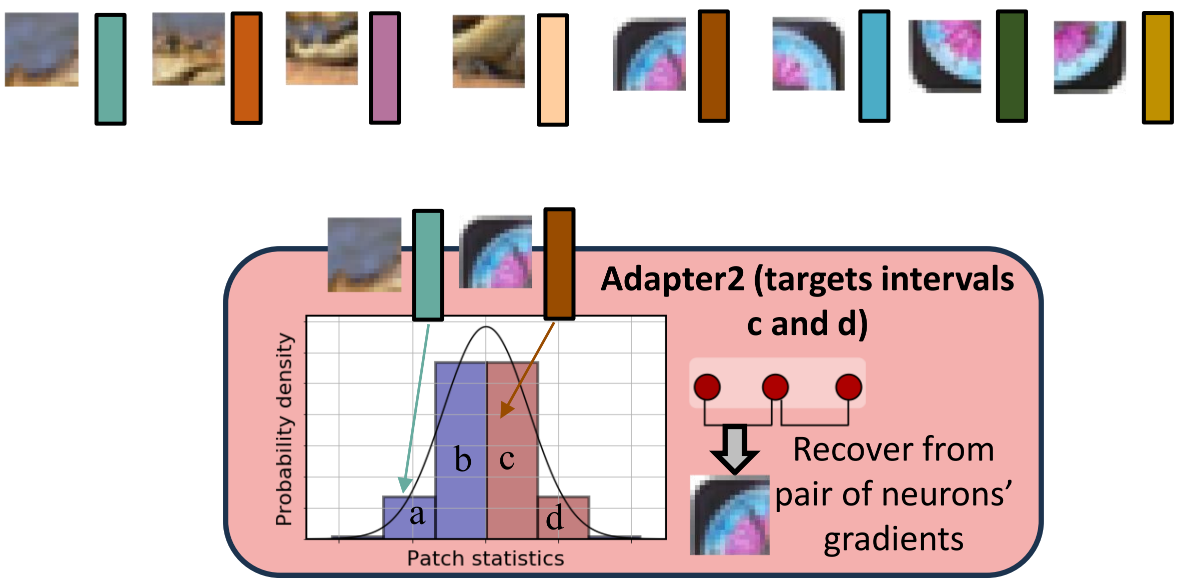
$$= \mathbf{y}^{(n,m)} \text{ for } n \in \{0, \ldots N\}$$

Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



This Work (PEFTLeak)

Input: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$



Output: $\mathbf{y}^{(n,m)} \in \mathbb{R}^D$ (through residual)

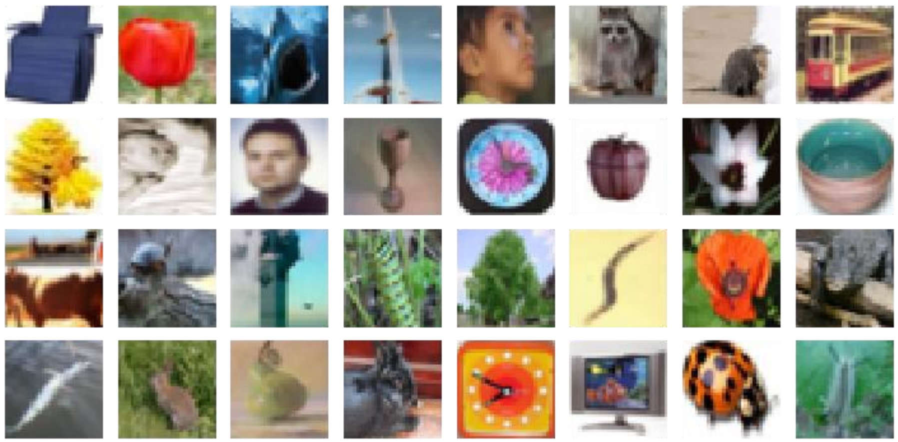
Experimental Setup

- Image classification task
- **Pretrained model:** ViT architecture
- A batch of 32 images
- Embedding dimension $D = 768$
- Bottleneck dimension r $= 64$
- Patch size (16, 16)
- **Datasets:** CIFAR-10, CIFAR-100 (4 patches)

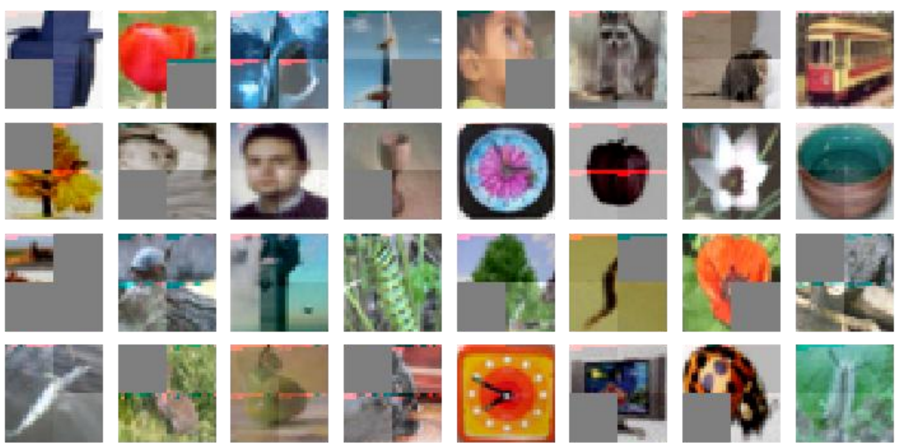
  TinyImageNet (16 patches)

  ImageNet (196 patches)

Reconstruction Quality (CIFAR-100)

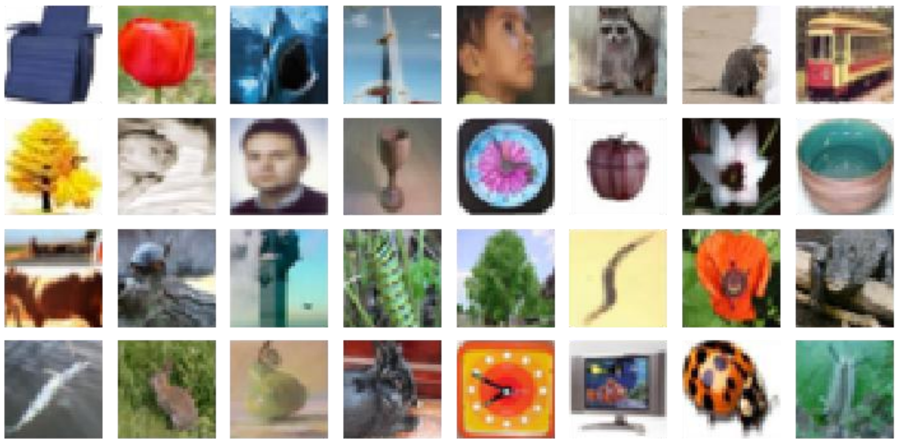


Ground-truth



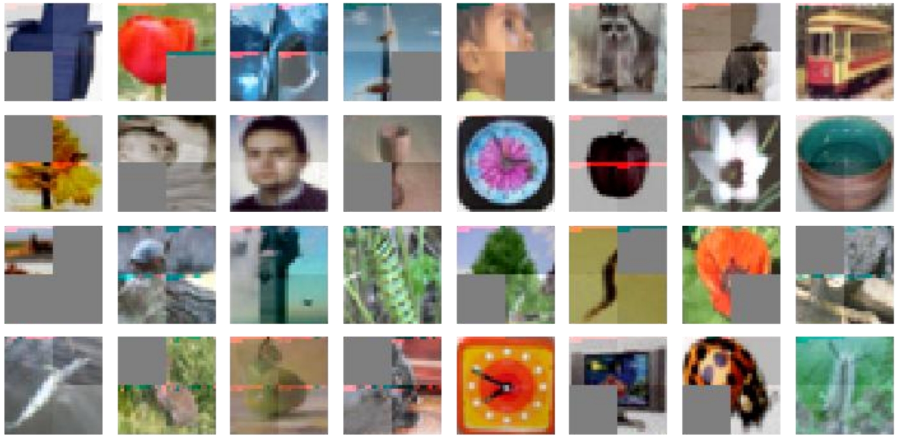
Recovered

Reconstruction Quality (CIFAR-100)



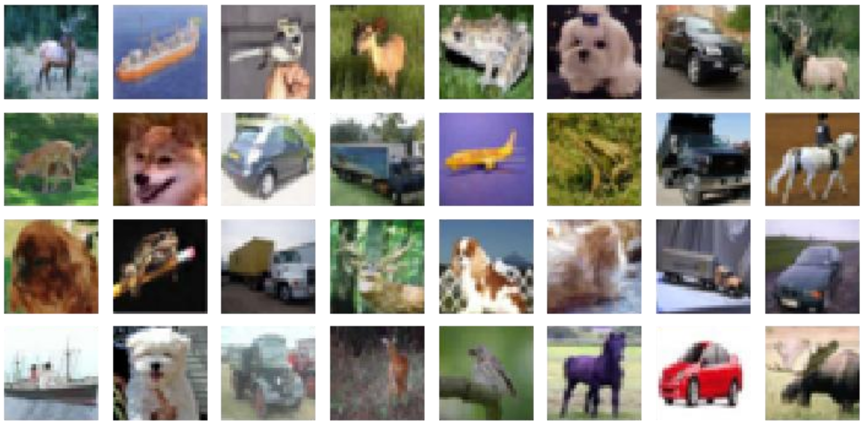
Ground-truth

85.9% of the patches are recovered



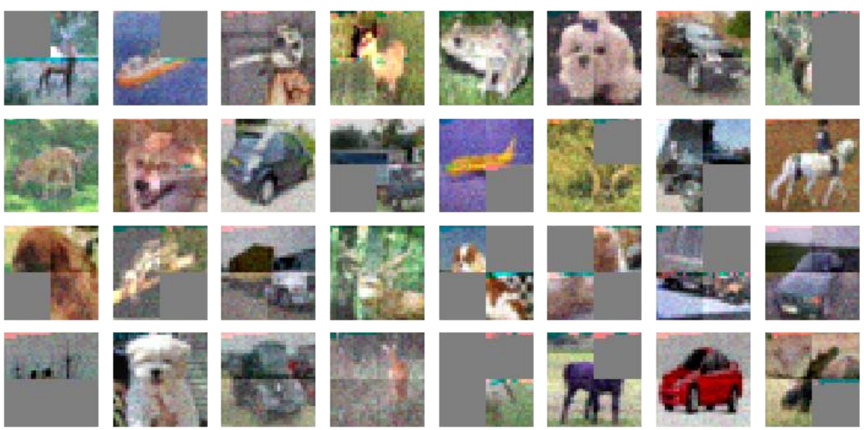
Recovered

Reconstruction Quality (CIFAR-10)



Ground-truth

82.8% of the patches are recovered

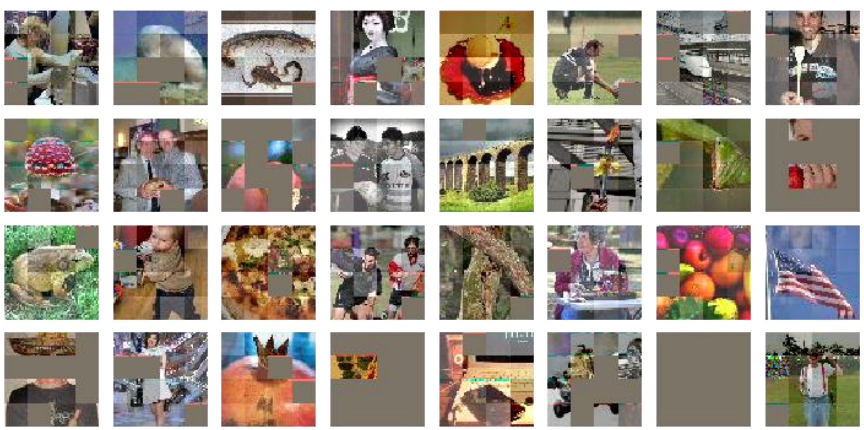


Recovered

Reconstruction Quality (TinyImageNet)

Ground-truth

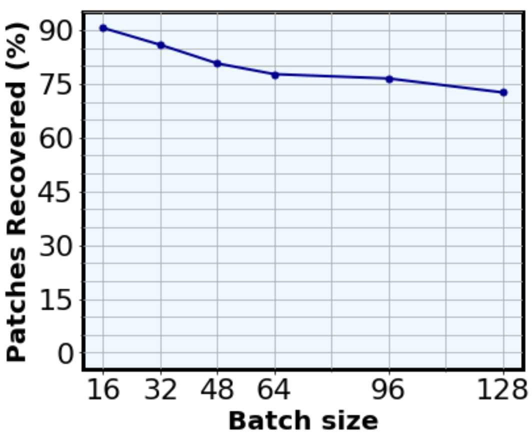
81% of the patches are recovered



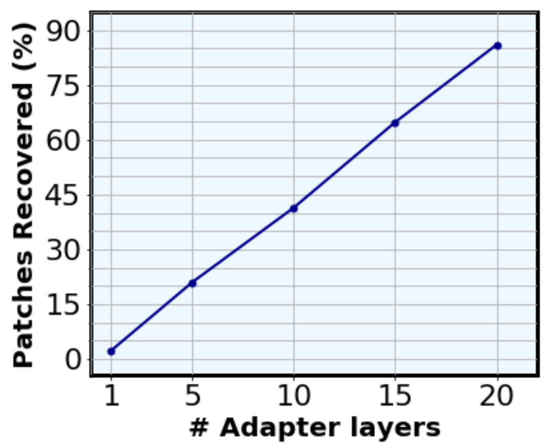
Recovered

Ablation Study (CIFAR-100)

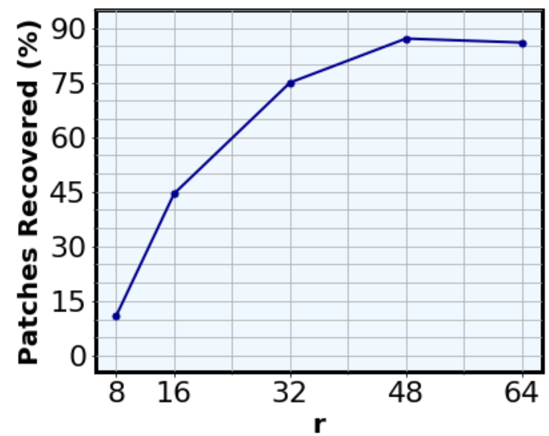
Varying batch size



Varying # layers

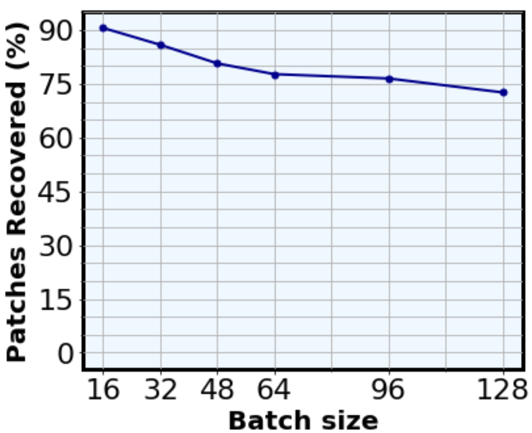


Varying bottleneck dimension, r

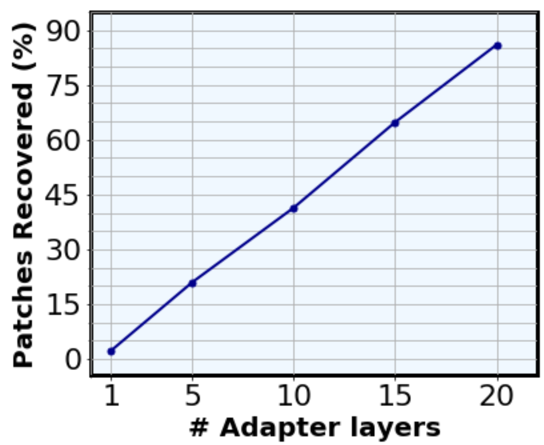


Ablation Study (CIFAR-100)

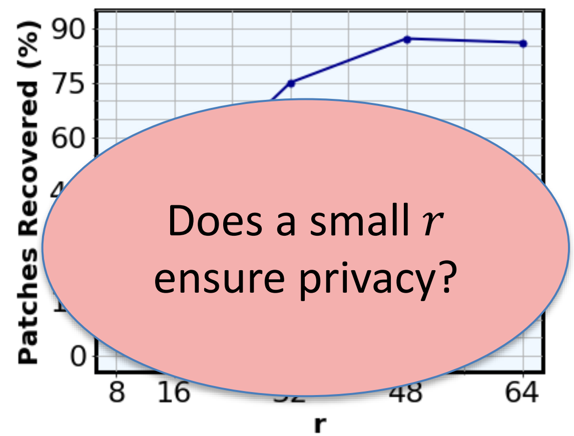
Varying batch size



Varying # layers



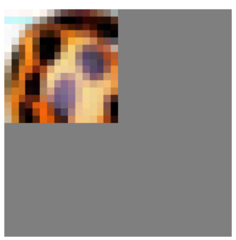
Varying bottleneck dimension, r



Ablation Study (CIFAR-100)

- Targets different intervals at different training rounds
- Recovers all patches over multiple rounds ($r = 8$)

Round 1



Round 2

Round 4

Round 5

Conclusions

- We discover the privacy risks associated with PEFT-based FL.
- We demonstrate how fine-tuning data can be recovered by malicious tampering with the pretrained model and adapters
- Our work demonstrates the critical need of developing privacy-aware PEFT framework

Thank you

Hasin Us Sami

hsami003@ucr.edu