# Just a Simple Transformation is Enough for Data Protection in Vertical Federated Learning

Andrei Semenov

MLO Group Meeting, 09.10.2024

# Intro: Privacy & FL

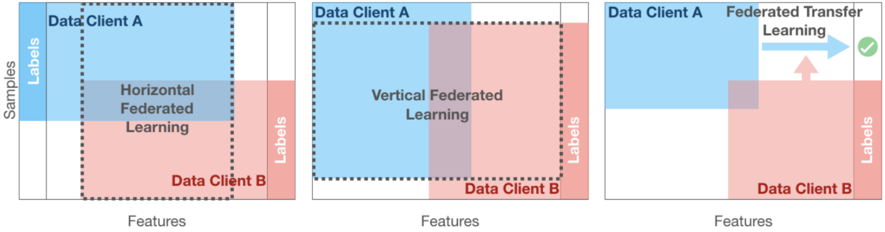
... for Data Protection in Vertical Federated Learning

# Intro: Privacy & FL

- Federated Learning

# Intro: Privacy & FL

- Federated Learning




(a) Horizontal Federated Learning (b) Vertical Federated Learning (c) Federated Transfer Learning

# Intro: Privacy & FL

- (Horizontal) Federated Learning

# Intro: Privacy & FL

- (Horizontal) Federated Learning

**Algorithm 1** `FedAvg`

**The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, $\gamma$ is the learning rate, and $C$ is the fraction of clients.**

```
Server executes:
Initialize W_0
for each round t = 1, 2, ... do
    m ← max(C · K, 1)
    S_t ← (random set of m clients)
    for each client k ∈ S_t in parallel do
        W^k_{t+1} ← ClientUpdate(k, W_t)
    end for
    m_t ← Σ_{k∈S_t} n_k
    W_{t+1} ← Σ_{k∈S_t} (n_k / m_t) W^k_{t+1}
end for
```

**ClientUpdate($k$, $W$):** *// Run on client $k$*

```
for each local epoch i from 1 to E do
    for batch b ∈ 𝓑 do
        W ← W − γ∇𝓛(b, W)
    end for
end for
return W to server
```

# Intro: Privacy & FL

- (Horizontal) Federated Learning

**Algorithm 2** `FedAvg`

**The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, $\gamma$ is the learning rate, and $C$ is the fraction of clients.**

**Server executes:**
Initialize $W_0$
**for** each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of $m$ clients)
  **for** each client $k \in S_t$ **in parallel do**
    $W_{t+1}^k \leftarrow \text{ClientUpdate}(k, W_t)$
  **end for**
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $W_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} W_{t+1}^k$
  **end for**

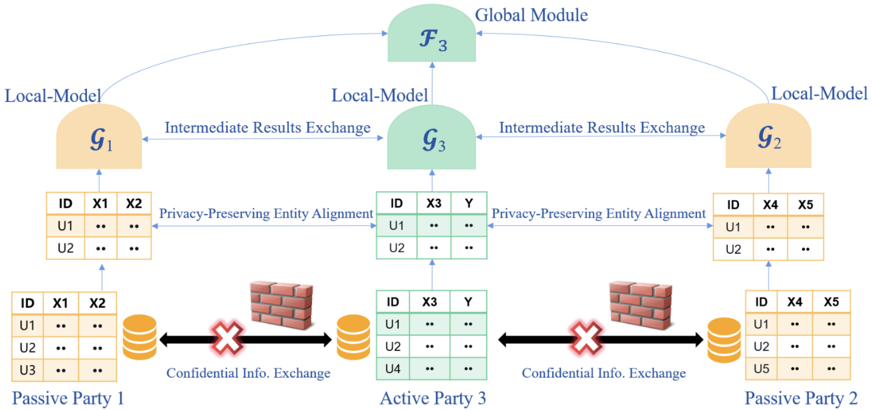
**ClientUpdate($k$, $W$):** *// Run on client $k$*
**for** each local epoch $i$ from 1 to $E$ **do**
  **for** batch $b \in \mathcal{B}$ **do**
    $W \leftarrow W - \gamma \nabla \mathcal{L}(b, W)$
  **end for**
**end for**
**return** $W$ to server

# Intro: Privacy & FL

- (Vertical) Federated Learning

# Intro: Privacy & FL

- (Vertical) Federated Learning

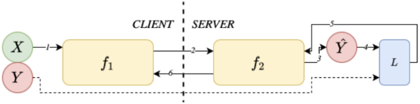


# Intro: Privacy & FL

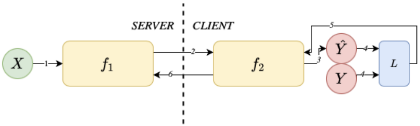
- Vertical Federated Learning → Split Learning

# Intro: Privacy & FL

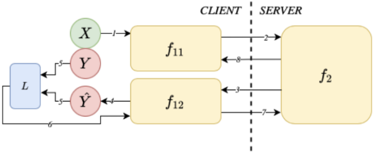
- Vertical Federated Learning → Split Learning




**(a) Training examples and labels at the client.**



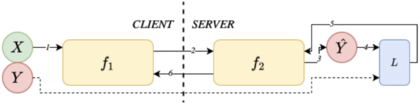

**(b) Training examples and labels are split between the client and the server.**



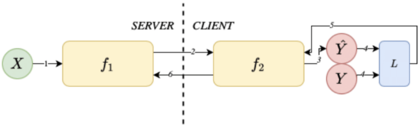

**(c) Training examples and labels stored only at the client.**

# Intro: Privacy & FL

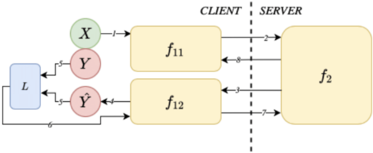
- Vertical Federated Learning → Split Learning




(a) Training examples and labels at the client.




(b) Training examples and labels are split between the client and the server.




(c) Training examples and labels stored only at the client.

# Intro: Privacy & FL

- Data Protection

# Intro: Privacy & FL

- Data Protection

**Attacks**

Label Inference
Feature Reconstruction
Model Reconstruction

# Intro: Privacy & FL

- Data Protection

| Attacks | Defenses |
| --- | --- |
| Label Inference | Cryptographic-based methods |
| Feature Reconstruction | Differential Privacy |
| Model Reconstruction | Obfuscation-based approaches |

# Intro: Privacy & FL

- Data Protection

| Attacks | Defenses |
| --- | --- |
| Label Inference | Cryptographic-based methods |
| Feature Reconstruction | Differential Privacy |
| Model Reconstruction | Obfuscation-based approaches |

# Intro: Privacy & FL

- Data Protection

**Attacks**

Label Inference
Feature Reconstruction
Model Reconstruction

**Attacker's knowledge**

**Defenses**

Cryptographic-based methods
Differential Privacy
Obfuscation-based approaches

# Intro: Privacy & FL

- Data Protection

**Attacks**

Label Inference
Feature Reconstruction
Model Reconstruction

**Attacker's knowledge**
White-Box assumption
Black-Box assumption

**Defenses**

Cryptographic-based methods
Differential Privacy
Obfuscation-based approaches

**Attacker's acting**

# Intro: Privacy & FL

- Data Protection

| Attacks | Defenses |
|---|---|
| Label Inference | Cryptographic-based methods |
| Feature Reconstruction | Differential Privacy |
| Model Reconstruction | Obfuscation-based approaches |

**Attacker's knowledge**

White-Box assumption

Black-Box assumption

**Attacker's acting**

Honest-but-curious

Malicious

# Intro: Privacy & FL

- Data Protection

**Attacks**

Label Inference
Feature Reconstruction
Model Reconstruction

**Attacker's knowledge**
White-Box assumption
Black-Box assumption

**Defenses**

Cryptographic-based methods
Differential Privacy
Obfuscation-based approaches

**Attacker's acting**
Honest-but-curious
Malicious

# Setup

# Setup

- Training under the Split Learning protocol

# Setup

- Training under the Split Learning protocol
- Aiming to protect the data against Feature Reconstruction attacks

# Setup

- Training under the Split Learning protocol
- Aiming to protect the data against Feature Reconstruction attacks
- While the attacker can be either Malicious or Honest-but-curious

# UnSplit

# UnSplit

Given a client model $f$, its clone $\tilde{f}$ (i.e., the randomly initialized model with the same architecture), the adversary server attempts to solve the two-step optimization problem:

$$\tilde{X}^* = \arg\min_{\tilde{X}} \mathcal{L}_{\mathrm{MSE}}\left(\tilde{f}(\tilde{X}, \tilde{W}),\ f(X, W)\right) + \lambda \mathrm{TV}(\tilde{X}),$$

$$\tilde{W}^* = \arg\min_{\tilde{W}} \mathcal{L}_{\mathrm{MSE}}\left(\tilde{f}(\tilde{X}, \tilde{W}),\ f(X, W)\right).$$

# UnSplit

Given a client model $f$, its clone $\tilde{f}$ (i.e., the randomly initialized model with the same architecture), the adversary server attempts to solve the two-step optimization problem:

$$\tilde{X}^* = \arg\min_{\tilde{X}} \mathcal{L}_{\mathrm{MSE}}\left(\tilde{f}(\tilde{X}, \tilde{W}),\ f(X, W)\right) + \lambda \mathrm{TV}(\tilde{X}),$$

$$\tilde{W}^* = \arg\min_{\tilde{W}} \mathcal{L}_{\mathrm{MSE}}\left(\tilde{f}(\tilde{X}, \tilde{W}),\ f(X, W)\right).$$

$X$, $W$ are the client model's private inputs and parameters; $\mathrm{TV}$ is the total variation distance for image pixels; $\tilde{X}^*$, $\tilde{W}^*$ are the desired variables for the attacker's reconstructed output and parameters

# Hijacking attack (FSHA)

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.

We have client-side model $f : \mathcal{X} \rightarrow \mathcal{Z}$.

Server initializes three additional models:

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.
We have client-side model $f : \mathcal{X} \to \mathcal{Z}$.
Server initializes three additional models: encoder $\psi_{\mathrm{E}} : \mathcal{X} \to \tilde{\mathcal{Z}} \subset \mathcal{Z}$,

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.
We have client-side model $f : \mathcal{X} \to \mathcal{Z}$.
Server initializes three additional models: encoder $\psi_{\mathrm{E}} : \mathcal{X} \to \tilde{\mathcal{Z}} \subset \mathcal{Z}$, decoder $\psi_{\mathrm{D}} : \tilde{\mathcal{Z}} \to \mathcal{X}$,

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.
We have client-side model $f : \mathcal{X} \to \mathcal{Z}$.
Server initializes three additional models: encoder $\psi_{\mathrm{E}} : \mathcal{X} \to \tilde{\mathcal{Z}} \subset \mathcal{Z}$, decoder $\psi_{\mathrm{D}} : \tilde{\mathcal{Z}} \to \mathcal{X}$, and discriminator $D$.

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.
We have client-side model $f : \mathcal{X} \to \mathcal{Z}$.
Server initializes three additional models: encoder $\psi_{\mathrm{E}} : \mathcal{X} \to \tilde{\mathcal{Z}} \subset \mathcal{Z}$, decoder $\psi_{\mathrm{D}} : \tilde{\mathcal{Z}} \to \mathcal{X}$, and discriminator $D$.

$$\psi^*_{\mathrm{E}},\ \psi^*_{\mathrm{D}} = \arg\min_{\psi_{\mathrm{E}},\psi_{\mathrm{D}}} \mathcal{L}_{\mathrm{MSE}}\left(\psi_{\mathrm{D}}(\psi_{\mathrm{E}}(X_{\mathrm{pub}})),\ X_{\mathrm{pub}}\right),$$
$$D = \arg\min_{\mathrm{D}} \left[\log(1 - D(\psi_{\mathrm{E}}(X_{\mathrm{pub}}))) + \log(D(f(X)))\right],$$
$$\mathcal{L}^* = \arg\min_{f} \left[\log\left(1 - D(f(X))\right)\right].$$

# Hijacking attack (FSHA)

Slightly different assumptions, the attacker has an access to some public part of the dataset.

We have client-side model $f : \mathcal{X} \to \mathcal{Z}$.

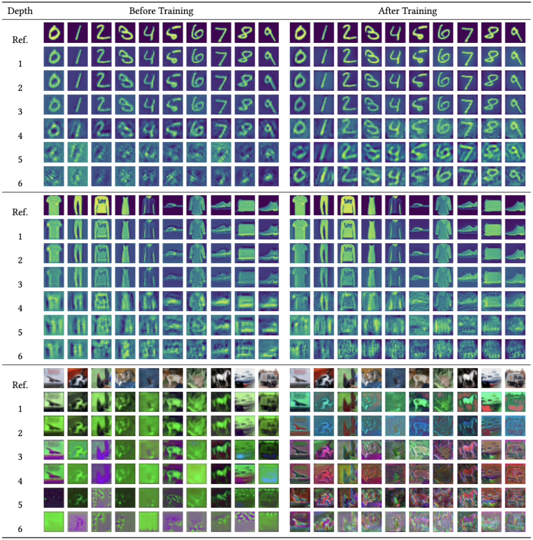
Server initializes three additional models: encoder $\psi_{\mathrm{E}} : \mathcal{X} \to \tilde{\mathcal{Z}} \subset \mathcal{Z}$, decoder $\psi_{\mathrm{D}} : \tilde{\mathcal{Z}} \to \mathcal{X}$, and discriminator $D$.

$$\begin{aligned} \psi_{\mathrm{E}}^*,\ \psi_{\mathrm{D}}^* &= \arg\min_{\psi_{\mathrm{E}},\psi_{\mathrm{D}}} \mathcal{L}_{\mathrm{MSE}}\left(\psi_{\mathrm{D}}(\psi_{\mathrm{E}}(X_{\mathrm{pub}})),\ X_{\mathrm{pub}}\right), \\ D &= \arg\min_{\mathrm{D}} \left[\log(1 - D(\psi_{\mathrm{E}}(X_{\mathrm{pub}}))) + \log(D(f(X)))\right], \\ \mathcal{L}^* &= \arg\min_{f} \left[\log\left(1 - D(f(X))\right)\right]. \end{aligned}$$

And, finally, server recovers features with:

$$\tilde{X}^* = \psi_{\mathrm{D}}^*\left(\mathcal{L}^*(X)\right).$$

# Observations 1



# Observations 2

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures
- Hard to analyze their performance in theory

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures
- Hard to analyze their performance in theory

**???**

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures
- Hard to analyze their performance in theory

**???**

- Does architectural design play a crucial role in the effectiveness of the latter attacks?

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures
- Hard to analyze their performance in theory

**???**

- Does architectural design play a crucial role in the effectiveness of the latter attacks?
- Is it that simple to attack features, or does the data prior knowledge give a lot?

# Observations 2

- Both of these attacks are validated exclusively on image datasets, utilizing CNN architectures
- Hard to analyze their performance in theory

**???**

- Does architectural design play a crucial role in the effectiveness of the latter attacks?
- Is it that simple to attack features, or does the data prior knowledge give a lot?
- Can we develop a theoretical intuition that MLP-based models might be more privacy-preserving againts Feature Reconstruction attacks?

# Observations 3: Theoretical Motivation

# Observations 3: Theoretical Motivation

Let us consider a client $f$ as **one-layer** linear model $f = XW$ with $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d_h}}$

# Observations 3: Theoretical Motivation

Let us consider a client $f$ as **one-layer** linear model $f = XW$ with $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d_h}}$

Introduce a **pairs** $\{X, W\} \rightarrow \{\tilde{\mathcal{X}}, \tilde{W}\} = \{X U, U^\top W\}$, where $U$ is an arbitrary (semi)orthogonal matrix (transformations)

# Observations 3: Theoretical Motivation

Let us consider a client $f$ as **one-layer** linear model $f = XW$ with $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d_h}}$

Introduce a **pairs** $\{X, W\} \to \{\tilde{\mathcal{X}}, \tilde{W}\} = \{XU, U^\top W\}$, where $U$ is an arbitrary (semi)orthogonal matrix (transformations)

Assume training with `(S)GD` for $\mathrm{k}$ iterations

# Observations 3: Theoretical Motivation

Let us consider a client $f$ as **one-layer** linear model $f = XW$ with $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d_h}}$

Introduce a **pairs** $\{X, W\} \rightarrow \{\tilde{\mathcal{X}}, \tilde{W}\} = \{X U, U^\top W\}$, where $U$ is an arbitrary (semi)orthogonal matrix (transformations)

Assume training with `(S)GD` for $\mathrm{k}$ iterations

1. **Base case,** $\mathrm{k} = 1$**:** $H_1 = X_1 W_1 = X_1 U U^\top W_1 = \tilde{X}_1 \tilde{W}_1 = \tilde{H}_1$

# Observations 3: Theoretical Motivation

Let us consider a client $f$ as **one-layer** linear model $f = XW$ with $W \in \mathbb{R}^{\mathrm{d} \times \mathrm{d_h}}$

Introduce a **pairs** $\{X, W\} \to \{\tilde{\mathcal{X}}, \tilde{W}\} = \{XU, U^\top W\}$, where $U$ is an arbitrary (semi)orthogonal matrix (transformations)

Assume training with `(S)GD` for $\mathrm{k}$ iterations

1. **Base case,** $\mathrm{k} = 1$**:** $H_1 = X_1 W_1 = X_1 U U^\top W_1 = \tilde{X}_1 \tilde{W}_1 = \tilde{H}_1$
2. **Induction step,** $\mathrm{k} + 1 > 1$**:** Let $H_\mathrm{k} = \tilde{H}_\mathrm{k}$ by induction hypothesis. Then $\partial \mathcal{L} / \partial H_\mathrm{k} = \partial \mathcal{L} / \partial \tilde{H}_\mathrm{k} = G_\mathrm{k} \in \mathbb{R}^{\mathrm{n} \times \mathrm{d_h}}$. Recall, that

$$\frac{\partial \mathcal{L}}{\partial W_\mathrm{k}} = \frac{\partial \mathcal{L}}{\partial H_\mathrm{k}} \frac{\partial H_\mathrm{k}}{\partial W_\mathrm{k}} = X_\mathrm{k}^\top \frac{\partial \mathcal{L}}{\partial H_\mathrm{k}} = X_\mathrm{k}^\top G_\mathrm{k}.$$

# Observations 3: Theoretical Motivation

Then the step of GD for the pairs $\{\mathcal{X}, W_1\}$ and $\{\tilde{\mathcal{X}}, \tilde{W}_1\}$ returns

$$W_{\mathrm{k}+1} = W_{\mathrm{k}} - \gamma X_{\mathrm{k}}^{\top} G_{\mathrm{k}}$$

and

$$\tilde{W}_{\mathrm{k}+1} = \tilde{W}_{\mathrm{k}} - \gamma \tilde{X}_{\mathrm{k}}^{\top} G_{\mathrm{k}} = U^{\top} W_{\mathrm{k}} - \gamma U^{\top} X_{\mathrm{k}}^{\top} G_{\mathrm{k}}$$

respectively.
Thus, at $\mathrm{k}+1$ step

$$\begin{aligned} H_{\mathrm{k}+1} &= X_{\mathrm{k}+1} W_{\mathrm{k}+1} = X_{\mathrm{k}+1} W_{\mathrm{k}} - \gamma X_{\mathrm{k}+1} X_{\mathrm{k}}^{\top} G_{\mathrm{k}} = \\ &= X_{\mathrm{k}+1} U U^{\top} W_{\mathrm{k}} - \gamma X_{\mathrm{k}+1} U U^{\top} X_{\mathrm{k}}^{\top} G_{\mathrm{k}} = \\ &= \tilde{X}_{\mathrm{k}+1} \tilde{W}_{\mathrm{k}} - \gamma \tilde{X}_{\mathrm{k}+1} \tilde{X}_{\mathrm{k}}^{\top} G_{\mathrm{k}} = \\ &= \tilde{X}_{\mathrm{k}+1} \tilde{W}_{\mathrm{k}+1} = \tilde{H}_{\mathrm{k}+1}, \end{aligned}$$

i.e., the activations sent to the server are identical for $\{\mathcal{X}, W_1\}$, $\{\tilde{\mathcal{X}}, \tilde{W}_1\}$ pairs.

# Observations 3: Theoretical Motivation

**Lemma 1**

For a one-layer linear model trained using `GD` or `SGD`, there exist continually many pairs of client data and weights initialization that produce the same activations at each step.

# Observations 3: Theoretical Motivation

**Lemma 1**

For a one-layer linear model trained using `GD` or `SGD`, there exist continually many pairs of client data and weights initialization that produce the same activations at each step.

**Remark 1**

Under the conditions of Lemma 1, if the server has no prior information about the distribution of $X$, the label party cannot reconstruct initial data $X$ (only up to an arbitrary orthogonal transformation).

# Observations 3: Theoretical Motivation

**Lemma 1**

For a one-layer linear model trained using `GD` or `SGD`, there exist continually many pairs of client data and weights initialization that produce the same activations at each step.

**Remark 1**

Under the conditions of Lemma 1, if the server has no prior information about the distribution of $X$, the label party cannot reconstruct initial data $X$ (only up to an arbitrary orthogonal transformation).

What about the Malicious server???

# Observations 4: Theoretical Motivation

**Corollary 1**

Under the conditions of Lemma1, assume that server knows the first layer $W_1$ of $f$, and let this layer be an **invertible matrix**. Then, the label party cannot reconstruct the initial data $X$ (only up to an arbitrary orthogonal transformation).

# Observations 4: Theoretical Motivation

**Corollary 1**

Under the conditions of Lemma1, assume that server knows the first layer $W_1$ of $f$, and let this layer be an **invertible matrix**. Then, the label party cannot reconstruct the initial data $X$ (only up to an arbitrary orthogonal transformation).

Client transmits $H_{\mathrm{k}+1} = XW_{\mathrm{k}+1}$

# Observations 4: Theoretical Motivation

**Corollary 1**

Under the conditions of Lemma1, assume that server knows the first layer $W_1$ of $f$, and let this layer be an **invertible matrix**. Then, the label party cannot reconstruct the initial data $X$ (only up to an arbitrary orthogonal transformation).

Client transmits $H_{\mathrm{k+1}} = XW_{\mathrm{k+1}}$

$$W_{\mathrm{k+1}} = W_{\mathrm{k}} - \gamma X^{\top} G_{\mathrm{k}}^{\mathrm{fake}} =$$
$$= \left(W_{\mathrm{k-1}} - \gamma X^{\top} G_{\mathrm{k-1}}^{\mathrm{fake}}\right) - \gamma X^{\top} G_{\mathrm{k}}^{\mathrm{fake}} =$$
$$= \cdots = W_1 - \gamma X^{\top} \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right].$$

# Observations 4: Theoretical Motivation

$$H_{\mathrm{k+1}} = XW_{\mathrm{k+1}} = XW_1 - \gamma XX^\top \left[ \sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}} \right],$$

# Observations 4: Theoretical Motivation

$$H_{\mathrm{k+1}} = XW_{\mathrm{k+1}} = XW_1 - \gamma XX^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right],$$

$$\tilde{H}_{\mathrm{k+1}} = \tilde{X}W_1 - \gamma \tilde{X}\tilde{X}^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right] = \tilde{X}W_1 - \gamma XX^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right].$$

# Observations 4: Theoretical Motivation

$$H_{\mathrm{k+1}} = XW_{\mathrm{k+1}} = XW_1 - \gamma XX^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right],$$

$$\tilde{H}_{\mathrm{k+1}} = \tilde{X}W_1 - \gamma \tilde{X}\tilde{X}^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right] = \tilde{X}W_1 - \gamma XX^\top \left[\sum_{i=1}^{k} G_{\mathrm{i}}^{\mathrm{fake}}\right].$$

The server can only build its attack based on the knowledge of $\tilde{X} = XU$ and $\tilde{X}\tilde{X}^\top$.
This means that it cannot distinguish between two different pairs $\{\mathcal{X}, U\}$ if they generate the same values $\tilde{X}$ and $\tilde{X}\tilde{X}^\top$.

# Observations 4: Theoretical Motivation

**Lemma 2**

Under the conditions of Lemma 1, assume training with the malicious server sending arbitrary vectors instead of real gradients $G = \partial f / \partial H$. In addition, the server knows the initialization of the weight matrix $W_1$. Then, if the client applies a non-trainable orthogonal matrix before $W_1$, the malicious server cannot reconstruct initial data $X$ (only up to an arbitrary orthogonal transformation).

# Observations 4: Theoretical Motivation

**Lemma 2**

Under the conditions of Lemma 1, assume training with the malicious server sending arbitrary vectors instead of real gradients $G = \partial f / \partial H$. In addition, the server knows the initialization of the weight matrix $W_1$. Then, if the client applies a non-trainable orthogonal matrix before $W_1$, the malicious server cannot reconstruct initial data $X$ (only up to an arbitrary orthogonal transformation).

**Remark 2**

With the same reasons as for Lemma 1, if even the malicious server from Lemma 2 has no prior information about the distribution of $X$, it is impossible for the label party to reconstruct the initial data $X$.

# Cut Layer

Up until now, we considered the client-side model with one linear layer $W$ and proved that orthogonal transformation of data $X$ and weights $W$ lead to the same training protocol

# Cut Layer

Up until now, we considered the client-side model with one linear layer $W$ and proved that orthogonal transformation of data $X$ and weights $W$ lead to the same training protocol

The intuition behind Lemmas 1 and 2 suggests that in the client model, one should look for layers whose inputs cannot be given the prior distribution.

What about Cut Layer?

# Cut Layer

## Cut Layer Lemma

There exist continually many distributions of the activations before the linear Cut Layer that produce the same Split Learning protocol.

# Cut Layer

## Cut Layer Lemma

There exist continually many distributions of the activations before the linear Cut Layer that produce the same Split Learning protocol.

$H = ZW$, $Z \to \tilde{Z} = ZU$ and $W_1 \to \tilde{W}_1 = U^\top W_1$.

# Cut Layer

**Cut Layer Lemma**

There exist continually many distributions of the activations before the linear Cut Layer that produce the same Split Learning protocol.

$H = ZW$, $Z \to \tilde{Z} = ZU$ and $W_1 \to \tilde{W}_1 = U^\top W_1$.

Let us define the client's "previous" parameters (before $W$) as $\theta$ and function of this parameters as $f_\theta : f_\theta(\theta, X) = Z$.

# Cut Layer

## Cut Layer Lemma

There exist continually many distributions of the activations before the linear Cut Layer that produce the same Split Learning protocol.

$H = ZW$, $Z \to \tilde{Z} = ZU$ and $W_1 \to \tilde{W}_1 = U^\top W_1$.

Let us define the client's "previous" parameters (before $W$) as $\theta$ and function of this parameters as $f_\theta : f_\theta(\theta, X) = Z$.

Then

$$f(X, \theta, W) = H = f_\theta(\theta, X)W = ZW, \; \mathcal{L} = \mathcal{L}(H) = \mathcal{L}(ZW).$$

# Cut Layer

Let us consider the gradient of loss $\mathcal{L}$ w.r.t. $W$ and $\theta$:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial H}\frac{\partial H}{\partial W} = Z^\top \frac{\partial \mathcal{L}}{\partial H},$$
$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial H}\frac{\partial H}{\partial Z}\frac{\partial Z}{\partial \theta} = \left[\frac{\partial Z}{\partial \theta}\right]^* \frac{\partial \mathcal{L}}{\partial H} W^\top = J^* \frac{\partial \mathcal{L}}{\partial H} W^\top,$$

where $J = \frac{\partial Z}{\partial \theta}$ – Jacobian of $f_\theta$.

# Cut Layer

Let us consider the gradient of loss $\mathcal{L}$ w.r.t. $W$ and $\theta$:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial H}\frac{\partial H}{\partial W} = Z^\top \frac{\partial \mathcal{L}}{\partial H},$$
$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial H}\frac{\partial H}{\partial Z}\frac{\partial Z}{\partial \theta} = \left[\frac{\partial Z}{\partial \theta}\right]^* \frac{\partial \mathcal{L}}{\partial H} W^\top = J^* \frac{\partial \mathcal{L}}{\partial H} W^\top,$$

where $J = \frac{\partial Z}{\partial \theta}$ – Jacobian of $f_\theta$.
Thus, after the first two iterations we conclude:

$$H_1 = Z_1 W_1, \quad \theta_2 = \theta_1 - \gamma J_1^* \frac{\partial \mathcal{L}}{\partial H_1} W_1^\top, \quad W_2 = W_1 - \gamma Z_1^\top \frac{\partial \mathcal{L}}{\partial H_1},$$

and

$$H_2 = Z_2 W_2 = f_\theta(\theta_2, X) W_2 = f_\theta(\theta_2, X) W_1 - \gamma f_\theta(\theta_2, X) Z_1^\top \frac{\partial \mathcal{L}}{\partial H_1}.$$

# Cut Layer

Adding the additional orthogonal matrix $U$ results in:

$$\tilde{W}_1 = U^\top W_1, \quad \tilde{H}_1 = \tilde{Z}_1 \tilde{W}_1 = (Z_1 U)\tilde{W}_1 = H_1, \quad \tilde{W}_2 = \tilde{W}_1 - \gamma \tilde{Z}_1^\top \frac{\partial \mathcal{L}}{\partial H_1}$$

# Cut Layer

Adding the additional orthogonal matrix $U$ results in:

$$\tilde{W}_1 = U^\top W_1, \quad \tilde{H}_1 = \tilde{Z}_1 \tilde{W}_1 = (Z_1 U)\tilde{W}_1 = H_1, \quad \tilde{W}_2 = \tilde{W}_1 - \gamma \tilde{Z}_1^\top \frac{\partial \mathcal{L}}{\partial H_1}$$

$$\begin{aligned}\frac{\partial \tilde{\mathcal{L}}}{\partial \theta_1} &= \frac{\partial \tilde{\mathcal{L}}}{\partial H_1}\frac{\partial H_1}{\partial \tilde{Z}_1}\frac{\partial \tilde{Z}_1}{\partial Z_1}\frac{\partial Z_1}{\partial \theta} = J_1^* \frac{\partial \mathcal{L}}{\partial H_1}\tilde{W}_1^\top U^\top = J_1^* \frac{\partial \mathcal{L}}{\partial H_1}(U^\top W_1)^\top U^\top \\ &= J_1^* \frac{\partial \mathcal{L}}{\partial H_1} W_1^\top = \frac{\partial \mathcal{L}}{\partial \theta_1}.\end{aligned}$$

# Cut Layer

Adding the additional orthogonal matrix $U$ results in:

$$\tilde{W}_1 = U^\top W_1, \quad \tilde{H}_1 = \tilde{Z}_1 \tilde{W}_1 = (Z_1 U)\tilde{W}_1 = H_1, \quad \tilde{W}_2 = \tilde{W}_1 - \gamma \tilde{Z}_1^\top \frac{\partial \mathcal{L}}{\partial H_1}$$

$$\begin{aligned}\frac{\partial \tilde{\mathcal{L}}}{\partial \theta_1} &= \frac{\partial \tilde{\mathcal{L}}}{\partial H_1}\frac{\partial H_1}{\partial \tilde{Z}_1}\frac{\partial \tilde{Z}_1}{\partial Z_1}\frac{\partial Z_1}{\partial \theta} = J_1^* \frac{\partial \mathcal{L}}{\partial H_1}\tilde{W}_1^\top U^\top = J_1^* \frac{\partial \mathcal{L}}{\partial H_1}(U^\top W_1)^\top U^\top \\ &= J_1^* \frac{\partial \mathcal{L}}{\partial H_1} W_1^\top = \frac{\partial \mathcal{L}}{\partial \theta_1}.\end{aligned}$$

Then, for the activations obtained with and without $U$ we claim:

$$\begin{aligned}\tilde{H}_2 &= \tilde{Z}_2 \tilde{W}_2 = f(\theta_2, X) U \tilde{W}_2 \\ &= f(\theta_2, X) U \tilde{W}_1 - \gamma f(\theta_2, X) U \tilde{Z}_1^\top \frac{\partial \mathcal{L}}{\partial H_1} \\ &= H_2.\end{aligned}$$

# Hypothesis

**Hypothesis 1**

*Could it be that the attacks are successful due to the lack of dense layers in the client architecture? Will usage of MLP-based architectures for $f$, instead of CNNs, be more privacy preserving against Model Inversion attack and FSHA?*

# Experiments

Figure: Results of UnSplit attack on CIFAR-10. (**Top**): Original images. (**Middle**): CNN-based client model. (**Bottom**): MLP-Mixer client model.

# Experiments

Figure: Results of UnSplit attack on MNIST. (**Top**): Original images. (**Middle**): CNN-based client model. (**Bottom**): MLP-based client model.

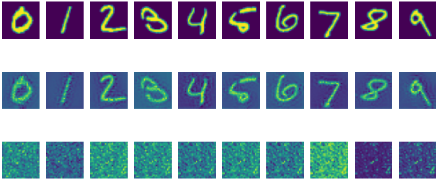
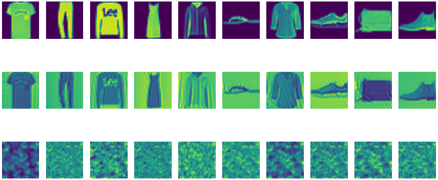


Figure: Results of UnSplit attack on F-MNIST. (**Top**): Original images. (**Middle**): CNN-based client model. (**Bottom**): MLP-based client model.



# Experiments

Figure: Results of FSHA attack on MNIST. (**Top**): Original images. (**Middle**): CNN-based client model. (**Bottom**): MLP-based client model.

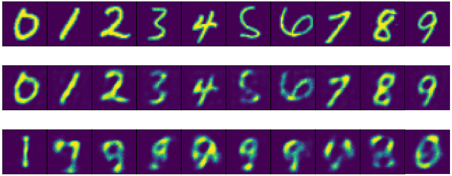
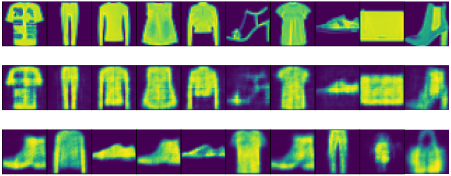
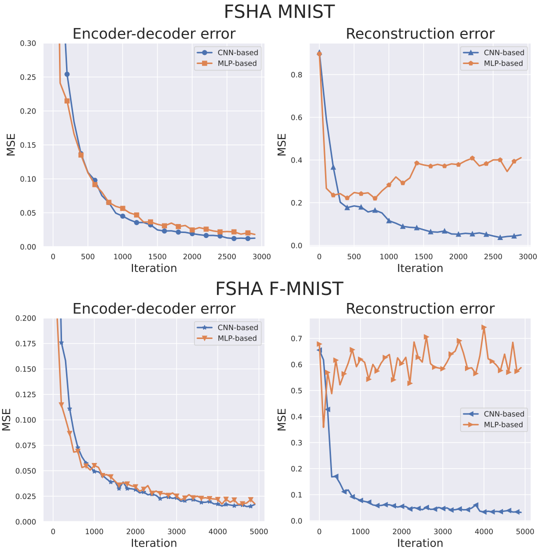


Figure: Results of FSHA attack on F-MNIST. (**Top**): Original images. (**Middle**): CNN-based client model. (**Bottom**): MLP-based client model.



# Experiments



# Split Learning with Adam

Previous theory works only for `(S)GD`-like methods. In practice, all experiments are correct with `Adam` also

# Split Learning with Adam

Previous theory works only for (S)GD-like methods. In practice, all experiments are correct with Adam also

What about Adam in theory?

# Split Learning with Adam

Previous theory works only for `(S)GD`-like methods. In practice, all experiments are correct with `Adam` also

## What about `Adam` in theory?

In `Adam`, the bias-corrected first and second moment estimates, i.e. $m_{\mathrm{k}}$ and $\hat{D}_{\mathrm{k}}$ are:

$$\begin{cases} m_k = \frac{1-\beta_1}{1-\beta_1^{\mathrm{k}}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \beta_1^{\mathrm{k}-\mathrm{i}} \nabla \mathcal{L}(W_{\mathrm{i}}), \\ \hat{D}_{\mathrm{k}}^2 = \frac{1-\beta_2}{1-\beta_2^{\mathrm{k}}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \beta_2^{\mathrm{k}-\mathrm{i}} \,\mathrm{diag}\left(\nabla \mathcal{L}(W_{\mathrm{i}}) \odot \nabla \mathcal{L}(W_{\mathrm{i}})\right), \end{cases}$$

# Split Learning with Adam

Previous theory works only for (S)GD-like methods. In practice, all experiments are correct with Adam also

## What about Adam in theory?

In Adam, the bias-corrected first and second moment estimates, i.e. $m_\mathrm{k}$ and $\hat{D}_\mathrm{k}$ are:

$$\begin{cases} m_k = \frac{1-\beta_1}{1-\beta_1^\mathrm{k}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \beta_1^{\mathrm{k}-\mathrm{i}} \nabla \mathcal{L}(W_\mathrm{i}), \\ \hat{D}_\mathrm{k}^2 = \frac{1-\beta_2}{1-\beta_2^\mathrm{k}} \sum_{\mathrm{i}=1}^{\mathrm{k}} \beta_2^{\mathrm{k}-\mathrm{i}} \operatorname{diag}\left(\nabla \mathcal{L}(W_\mathrm{i}) \odot \nabla \mathcal{L}(W_\mathrm{i})\right), \end{cases}$$

with the following update rule (without bias-correctness):

$$\begin{cases} \hat{m}_{\mathrm{k}+1} = \beta_1 \hat{m}_\mathrm{k} + (1-\beta_1) \nabla \mathcal{L}(W_\mathrm{k}), \\ \hat{D}_{\mathrm{k}+1}^2 = \beta_2 \hat{D}_\mathrm{k}^2 + (1-\beta_2) \operatorname{diag}(\nabla \mathcal{L}(W_\mathrm{k}) \odot \nabla \mathcal{L}(W_\mathrm{k})). \end{cases}$$

# Split Learning with Adam

**Remark Adam**

Let $\{\tilde{X}, \tilde{W}\} = \{XU, U^\top W\}$ pairs are an orthogonal(semi-orthogonal) transformations of data and weights. Then, these pairs, in general, do not produce the same activations at each step of the Split Learning process with `Adam`.

# Split Learning with Adam

**Remark Adam**

Let $\{\tilde{X}, \tilde{W}\} = \{XU, U^\top W\}$ pairs are an orthogonal(semi-orthogonal) transformations of data and weights. Then, these pairs, in general, do not produce the same activations at each step of the Split Learning process with `Adam`.

Indeed,

$$\hat{\tilde{D}}_{\mathrm{k}}^2 - \beta_2 \hat{D}_{\mathrm{k}-1}^2 = (1 - \beta_2)\,\mathrm{diag}\left(\frac{\partial \mathcal{L}}{\partial \tilde{W}_{\mathrm{k}}} \odot \frac{\partial \mathcal{L}}{\partial \tilde{W}_{\mathrm{k}}}\right)$$

# Split Learning with Adam

$$
\begin{aligned}
\hat{\tilde{D}}_{\mathrm{k}}^2 - \beta_2 \hat{D}_{\mathrm{k}-1}^2 &= (1-\beta_2)\,\mathsf{diag}\left(\frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}}\frac{\partial \tilde{H}_{\mathrm{k}}}{\partial \tilde{W}_{\mathrm{k}}} \odot \frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}}\frac{\partial \tilde{H}_{\mathrm{k}}}{\partial \tilde{W}_{\mathrm{k}}}\right) \\
&= (1-\beta_2)\,\mathsf{diag}\left(\tilde{X}^\top \frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}} \odot \tilde{X}^\top \frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}}\right) \\
&\overset{(\tilde{H}_{\mathrm{k}} = H_{\mathrm{k}})}{=} (1-\beta_2)\,\mathsf{diag}\left(\tilde{X}^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}} \odot \tilde{X}^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}}\right) \\
&= (1-\beta_2)\,\mathsf{diag}\left(U^\top X^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}} \odot U^\top X^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}}\right),
\end{aligned}
$$

# Split Learning with Adam

the similar holds for $\hat{\tilde{m}}_{\mathrm{k}}$

$$\hat{\tilde{m}}_{\mathrm{k}} - \beta_1 \hat{m}_{\mathrm{k}-1} = (1-\beta_1)\frac{\partial \mathcal{L}}{\partial \tilde{W}_{\mathrm{k}}} = (1-\beta_1)\frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}}\frac{\partial \tilde{H}_{\mathrm{k}}}{\partial \tilde{W}_{\mathrm{k}}} = (1-\beta_1)\tilde{X}^\top \frac{\partial \mathcal{L}}{\partial \tilde{H}_{\mathrm{k}}}$$
$$\overset{(\tilde{H}_{\mathrm{k}}=H_{\mathrm{k}})}{=} (1-\beta_1)\tilde{X}^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}} = (1-\beta_1)U^\top X^\top \frac{\partial \mathcal{L}}{\partial H_{\mathrm{k}}}.$$

# Split Learning with Adam

Then, it is clear how to compare the activations at $\mathrm{k}+1$-th step

$$\begin{cases} \tilde{H}_{\mathrm{k}+1} = \tilde{X}\tilde{W}_{\mathrm{k}+1} = XW_{\mathrm{k}} - \gamma XU\hat{\tilde{D}}_{\mathrm{k}}^{-1}\hat{\tilde{m}}_{\mathrm{k}}, \\ \\ H_{\mathrm{k}+1} = XW_{\mathrm{k}+1} = XW_{\mathrm{k}} - \gamma X\hat{D}_{\mathrm{k}}^{-1}\hat{m}_{\mathrm{k}}. \end{cases}$$

# Split Learning with Adam

Then, it is clear how to compare the activations at $\mathrm{k}+1$-th step

$$\begin{cases} \tilde{H}_{\mathrm{k}+1} = \tilde{X}\tilde{W}_{\mathrm{k}+1} = XW_{\mathrm{k}} - \gamma XU\hat{\tilde{D}}_{\mathrm{k}}^{-1}\hat{\tilde{m}}_{\mathrm{k}}, \\ \\ H_{\mathrm{k}+1} = XW_{\mathrm{k}+1} = XW_{\mathrm{k}} - \gamma X\hat{D}_{\mathrm{k}}^{-1}\hat{m}_{\mathrm{k}}. \end{cases}$$

Therefore, a descrepancy between $\tilde{H}_{\mathrm{k}+1}$ and $H_{\mathrm{k}+1}$ vanishes when

$$U\hat{\tilde{D}}_{\mathrm{k}}^{-1}\hat{\tilde{m}}_{\mathrm{k}} = \hat{D}_{\mathrm{k}}^{-1}\hat{m}_{\mathrm{k}}.$$

# Split Learning with Adam

As usual, `Adam` may not converge on general non-convex functions after the roation of data and weights

# Split Learning with Adam

As usual, `Adam` may not converge on general non-convex functions after the roation of data and weights

**Example** Indeed, let the initial weight and data vectors equal:

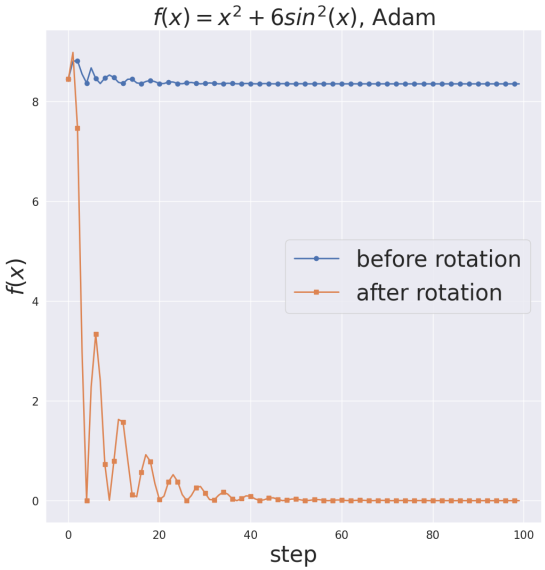
$$w = \left(1.915 + \sqrt{2} \cdot 0.6,\ 0\right)^\top,$$
$$y = (1,\ 0)^\top.$$

We rotate these arguments by an angle of $\frac{\pi}{4}$ with:

$$R = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

In addition we pick the learning rate $\gamma = 0.6$. After that, the optimization algorithm stack in the local minima if starting from $(Rw,\ Ry)$ point.

# Split Learning with Adam



# Split Learning with Adam

But, Adam will converge to the same optimal value on PL-functions

# Split Learning with Adam

But, Adam will converge to the same optimal value on PL-functions

**Remark**

The model's optimal value $\mathcal{L}^*$ after Split Learning is the same for any orthogonal data transformation. Indeed,
$\forall \tilde{X} = XU \; \exists \tilde{W}^* = U^\top W^* : \; \mathcal{L}(\tilde{X}, \tilde{W}^*) = \mathcal{L}^* = \mathcal{L}(X, W^*)$.

# Split Learning with Adam

But, Adam will converge to the same optimal value on PL-functions

**Remark**

The model's optimal value $\mathcal{L}^*$ after Split Learning is the same for any orthogonal data transformation. Indeed,
$\forall \tilde{X} = XU \; \exists \tilde{W}^* = U^\top W^* : \; \mathcal{L}(\tilde{X}, \tilde{W}^*) = \mathcal{L}^* = \mathcal{L}(X, W^*)$.

In addition, Split Learning protocol is preserved for PL functions with orthogonal transformations of data and weights

# Split Learning with Adam

## Descent Lemma

Suppose the $L$-smooth Assumption holds for function $\mathcal{L}$. Then we have for all $\mathrm{k} \geq 0$ and $\gamma$, it is true for `Adam` that

$$\mathcal{L}(W_{\mathrm{k}+1}) \leq \mathcal{L}(W_{\mathrm{k}}) + \frac{\gamma}{2\alpha}\|\nabla\mathcal{L}(W_{\mathrm{k}}) - m_{\mathrm{k}}\|^2 - \left(\frac{1}{2\gamma} - \frac{L}{2\alpha}\right)\|W_{\mathrm{k}+1} - W_{\mathrm{k}}\|^2_{\hat{D}_{\mathrm{k}}} - \frac{\gamma}{2}\|\nabla\mathcal{L}(W_{\mathrm{k}})\|^2_{\hat{D}_{\mathrm{k}}^{-1}}.$$

# Split Learning with Adam

**Descent Lemma**

Suppose the $L$-smooth Assumption holds for function $\mathcal{L}$. Then we have for all $\mathrm{k} \geq 0$ and $\gamma$, it is true for `Adam` that

$$\mathcal{L}(W_{\mathrm{k}+1}) \leq \mathcal{L}(W_{\mathrm{k}}) + \frac{\gamma}{2\alpha}\|\nabla\mathcal{L}(W_{\mathrm{k}}) - m_{\mathrm{k}}\|^2 - \left(\frac{1}{2\gamma} - \frac{L}{2\alpha}\right)\|W_{\mathrm{k}+1} - W_{\mathrm{k}}\|^2_{\hat{D}_{\mathrm{k}}} - \frac{\gamma}{2}\|\nabla\mathcal{L}(W_{\mathrm{k}})\|^2_{\hat{D}_{\mathrm{k}}^{-1}}.$$

Without proof here:)

# The End

**Thanks!**