Privacy Preserving Machine Learning

Jungwon Seo

Ph.D. Candidate at University of Stavanger

jungwon.seo@uis.no

Lecture Overview

- Federated Learning (80%)
  - Training

- Multi-Party Computation with Secret Sharing (20%)
  - Inference

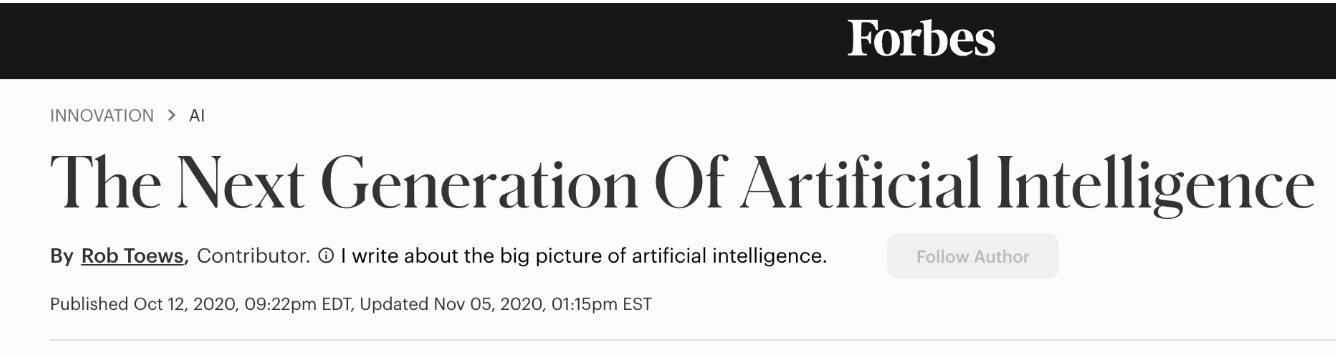
Privacy in Context

- Privacy can be understood in many ways, both broadly and in depth.

- In this lecture, when we discuss *privacy leakage* or *privacy preservation*, we are specifically concerned with the **exposure of raw data**
  - not with cases where generative models may reveal sensitive information.

Federated Learning

Should we work on this?



Forbes

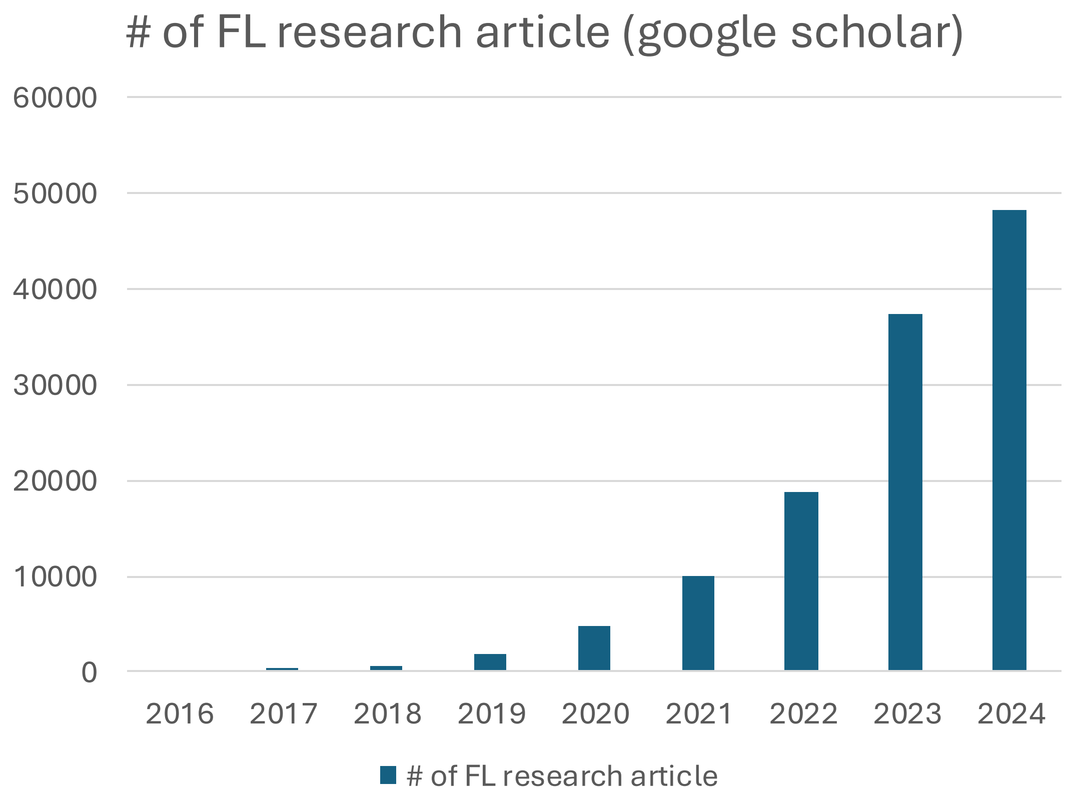
INNOVATION > AI

The Next Generation Of Artificial Intelligence

By Rob Toews, Contributor. ⓘ I write about the big picture of artificial intelligence.

Published Oct 12, 2020, 09:22pm EDT, Updated Nov 05, 2020, 01:15pm EST

1. Unsupervised Learning
2. **Federated Learning**
3. Transformers
4. Neural Network Compression
5. Generative AI
6. System 2 Reasoning

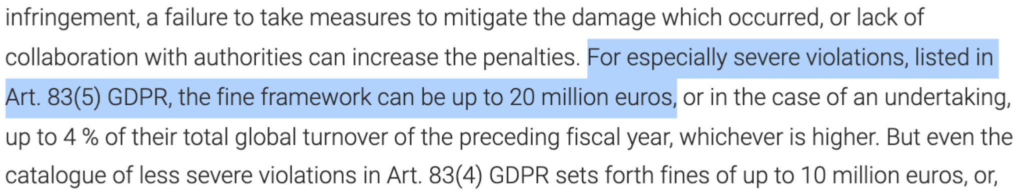


# of FL research article (google scholar)

https://www.forbes.com/sites/robtoews/2020/10/12/the-next-generation-of-artificial-intelligence/
https://www.forbes.com/sites/robtoews/2020/10/29/the-next-generation-of-artificial-intelligence-part-2/

Motivation

- In deep learning, **more data usually means better performance**.
- This creates a need to collect as much data as possible.
- Two key stakeholders:
  - **Model Trainers**
    - Want more data to improve models
    - Must comply with regulations (e.g., GDPR, HIPAA)
    - ...though violations still happen 🤫
  - **Data Providers**
    - Want better, more personalized experiences
    - Reluctant to share sensitive data
    - ...yet we often share anyway 🤫
- Is it possible to train a model without sharing (moving) the data?

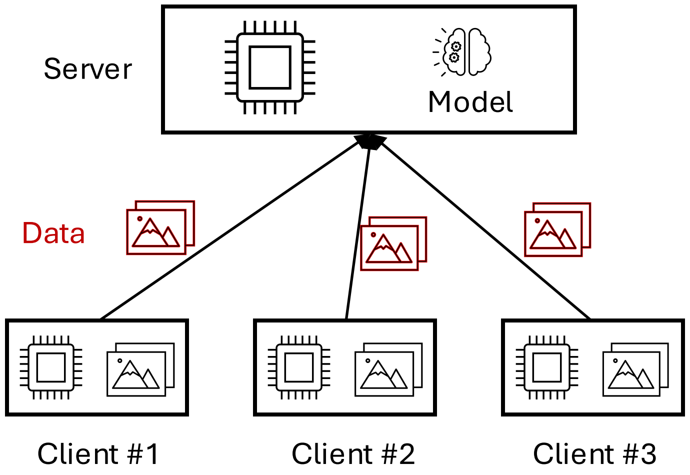


infringement, a failure to take measures to mitigate the damage which occurred, or lack of collaboration with authorities can increase the penalties. For especially severe violations, listed in Art. 83(5) GDPR, the fine framework can be up to 20 million euros, or in the case of an undertaking, up to 4 % of their total global turnover of the preceding fiscal year, whichever is higher. But even the catalogue of less severe violations in Art. 83(4) GDPR sets forth fines of up to 10 million euros, or,

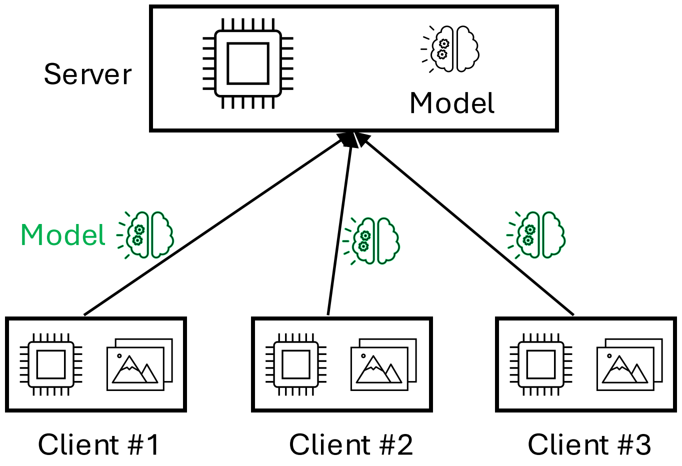
https://gdpr-info.eu/issues/fines-penalties/

Federated Learning

- Federated learning (FL) is a **privacy-preserving distributed** machine learning approach that enables training models without directly accessing locally owned data.



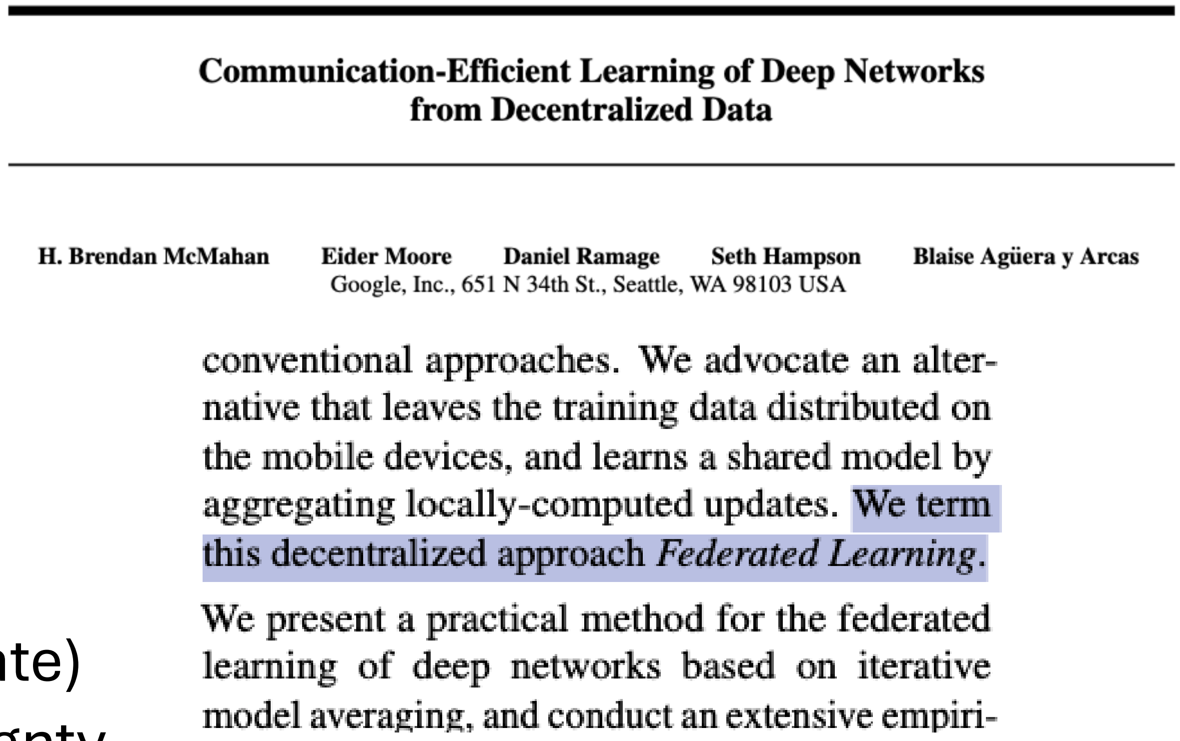
Centralized Learning



Federated Learning

Federated Learning

- Who named it?
- Why "Federated"?
  - Centralized = Monarch
  - Federated = Federation (federal state)
  - To highlight certain level of sovereignty

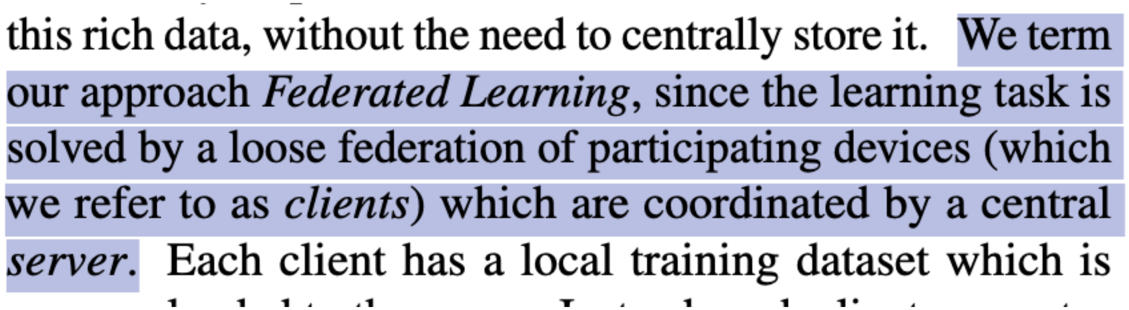


Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

conventional approaches. We advocate an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. We term this decentralized approach *Federated Learning*.

We present a practical method for the federated learning of deep networks based on iterative model averaging, and conduct an extensive empiri-

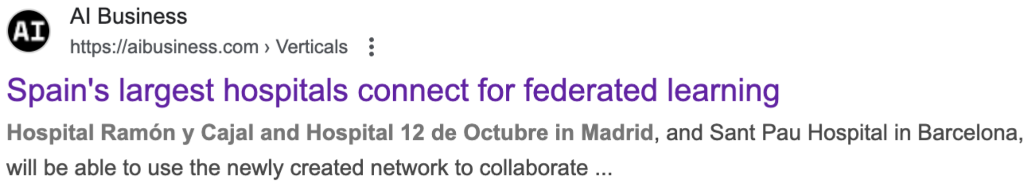


this rich data, without the need to centrally store it. We term our approach *Federated Learning*, since the learning task is solved by a loose federation of participating devices (which we refer to as *clients*) which are coordinated by a central *server*. Each client has a local training dataset which is

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." Artificial intelligence and statistics. PMLR, 2017.

Use cases

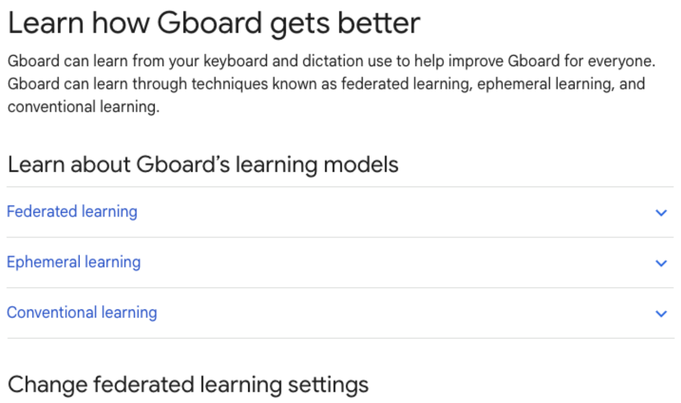
- Cross-Silo
  - Healthcare & Hospital: Patient data
  - Finance & Banking: Customer data
  - Other industries: Proprietary data
- Cross-Device
  - Smart-Keyboard (G-Board)
  - Autonomous Vehicles
  - Smartphone, IoT devices
- For which task?
  - Any types of task that they were working with their own data.
  - Classification, regression..
  - Recommendation, Object Detection…



AI Business
https://aibusiness.com › Verticals

Spain's largest hospitals connect for federated learning

Hospital Ramón y Cajal and Hospital 12 de Octubre in Madrid, and Sant Pau Hospital in Barcelona, will be able to use the newly created network to collaborate ...



Learn how Gboard gets better

Gboard can learn from your keyboard and dictation use to help improve Gboard for everyone. Gboard can learn through techniques known as federated learning, ephemeral learning, and conventional learning.

Learn about Gboard's learning models

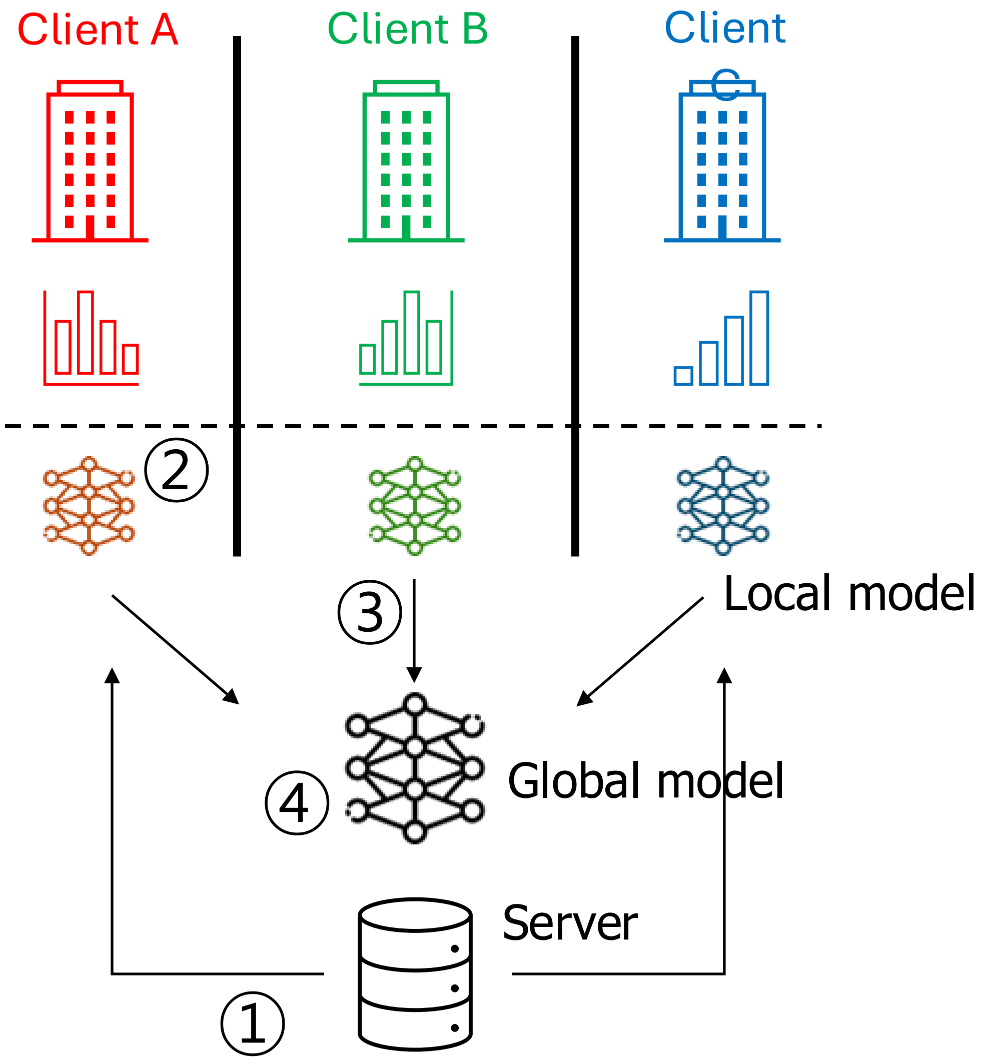
Change federated learning settings

https://aibusiness.com/verticals/spain-s-largest-hospitals-connect-for-federated-learning

https://support.google.com/gboard/answer/12373137?hl=en

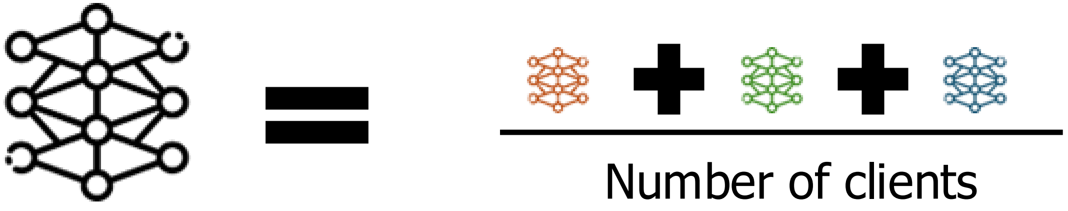
FL Training Steps

1. Model Deployment
2. Local Training
3. Model Upload
4. **Model Aggregation**
    1. Repeat from Step1



Model Aggregation in FL

- Averaging local models



$$\left\{\begin{matrix} 0.23 & 0.23 \\ 0.16 & 0.2 \end{matrix}\right\} = \frac{\left\{\begin{matrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{matrix}\right\} + \left\{\begin{matrix} 0.5 & 0.4 \\ 0.1 & 0.3 \end{matrix}\right\} + \left\{\begin{matrix} 0.1 & 0.1 \\ 0.1 & -0.1 \end{matrix}\right\}}{3}$$

- How can this simple averaging work?

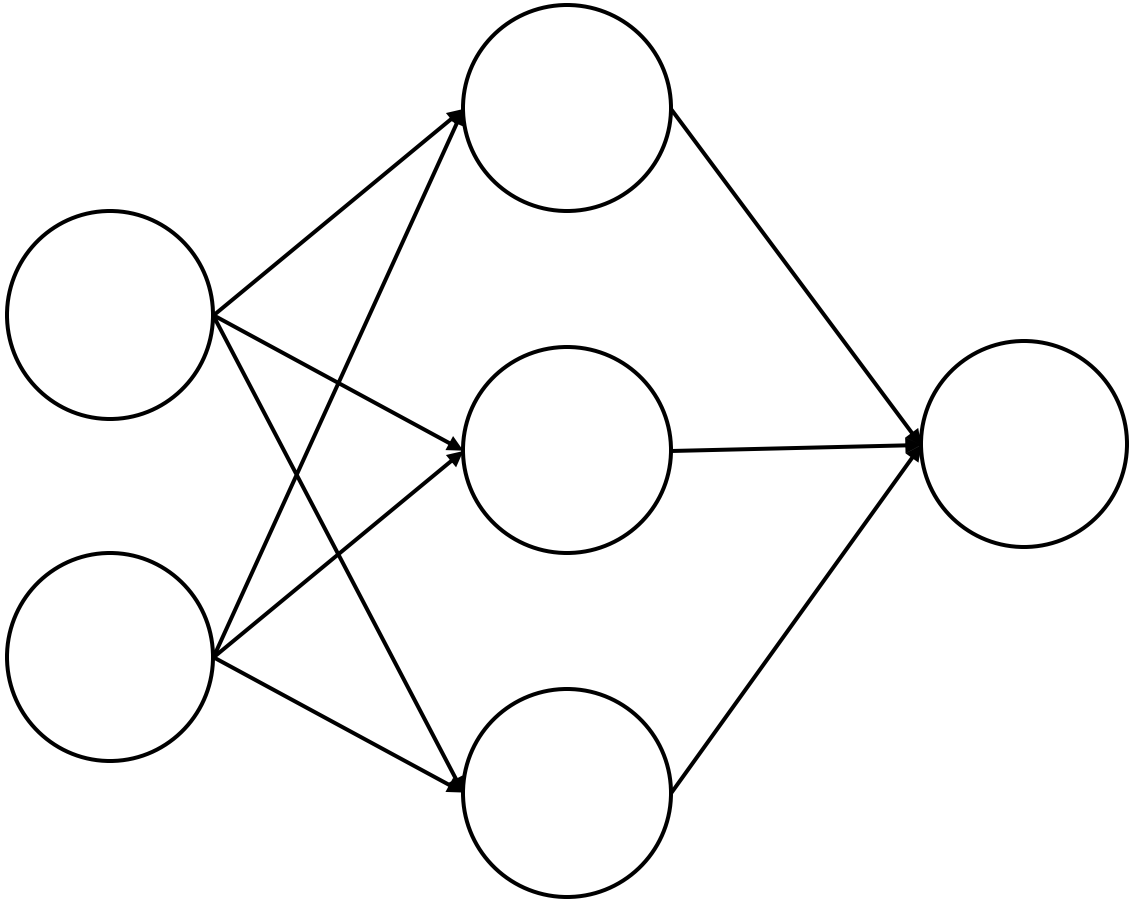
Our Goal

- Understanding Federated Learning from gradient descent to FedAvg

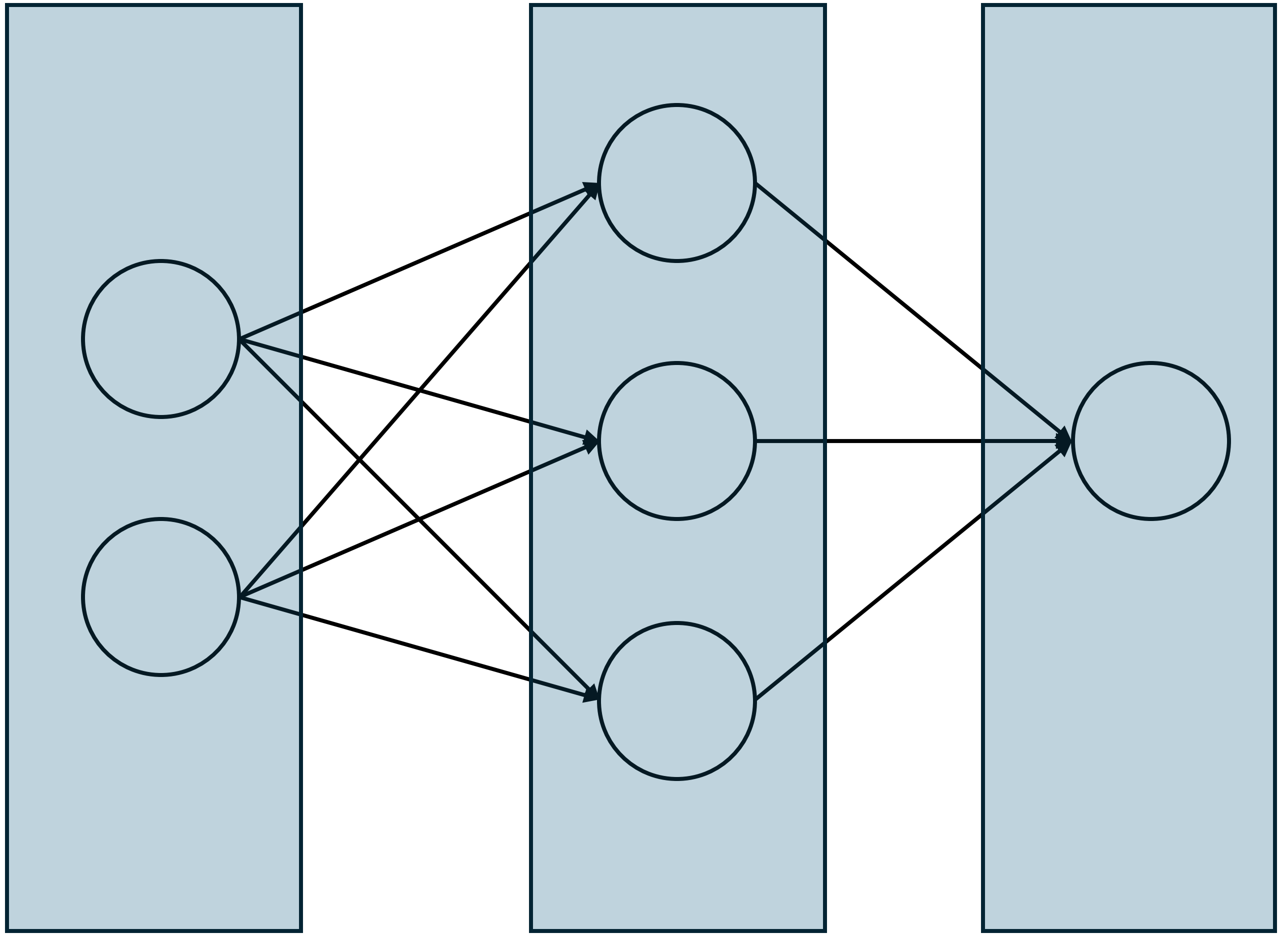
- Building intuition and insight about Federated Learning
  - So that we can quickly understand other FL research.

Neural Network 101

- Layer
- Weight
- Gradient



Neural Network: Layer

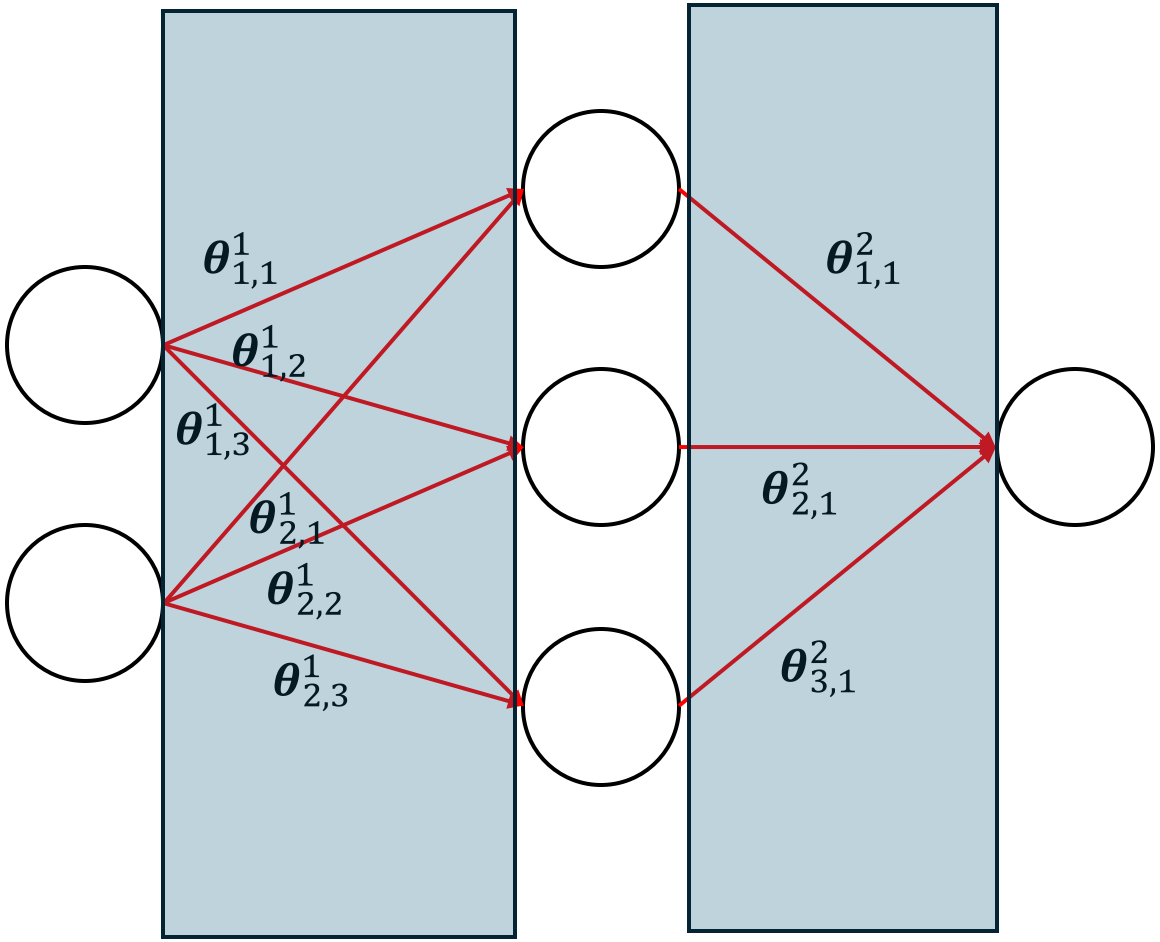


Layer 1
(Input)

Layer 2
(hidden)

Layer 3
(output)

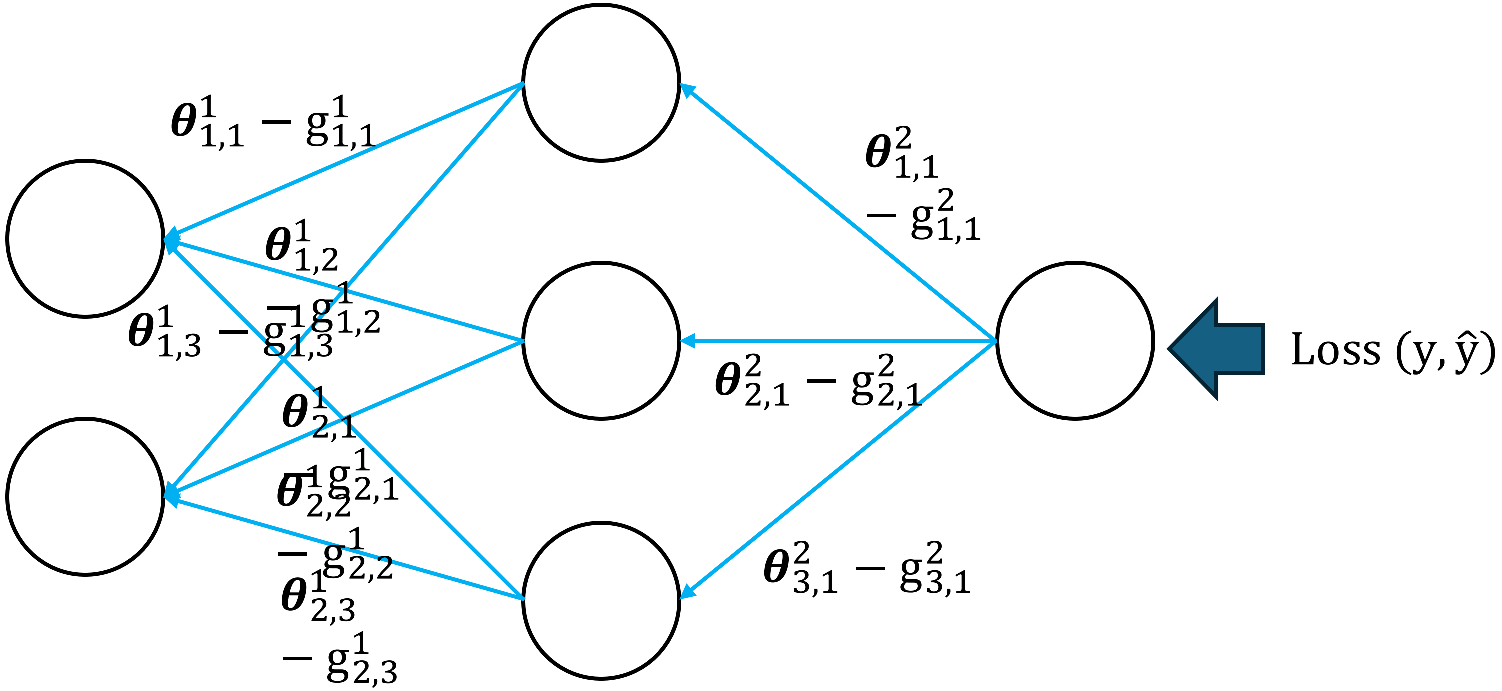
Neural Network: Weight (Parameter)



Weight 1-2
(Parameter)

Weight 2-3
(Parameter)

Neural Network: Gradient

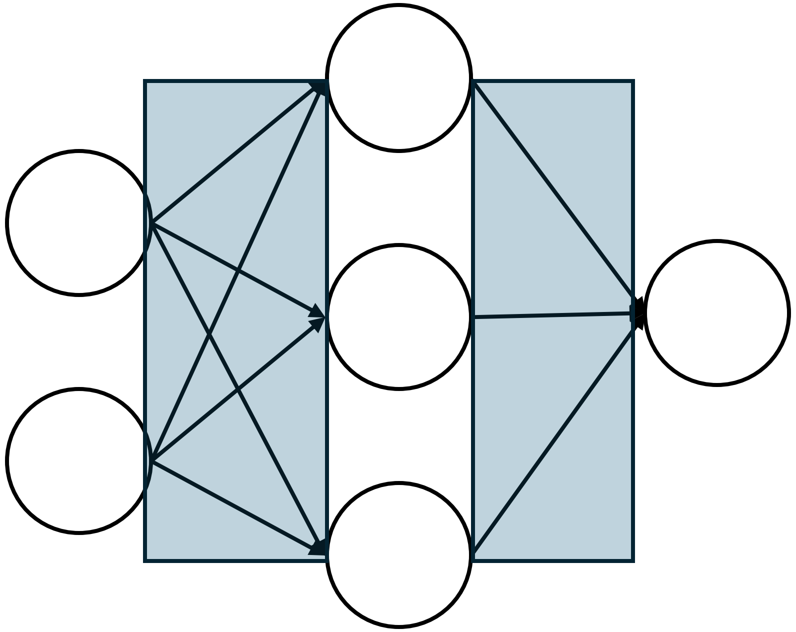

$\boldsymbol{\theta}_{1,1}^{1} - g_{1,1}^{1}$
$\boldsymbol{\theta}_{1,2}^{1} - g_{1,2}^{1}$
$\boldsymbol{\theta}_{1,3}^{1} - g_{1,3}^{1}$
$\boldsymbol{\theta}_{2,1}^{1} - g_{2,1}^{1}$
$\boldsymbol{\theta}_{2,2}^{1} - g_{2,2}^{1}$
$\boldsymbol{\theta}_{2,3}^{1} - g_{2,3}^{1}$
$\boldsymbol{\theta}_{1,1}^{2} - g_{1,1}^{2}$
$\boldsymbol{\theta}_{2,1}^{2} - g_{2,1}^{2}$
$\boldsymbol{\theta}_{3,1}^{2} - g_{3,1}^{2}$
Loss $(y, \hat{y})$

Neural Network

- One neural networks have several
  - Weight matrices (tensors)

$$\begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \theta_{1,3} \\ \theta_{2,1} & \theta_{2,2} & \theta_{2,3} \end{bmatrix} \qquad \begin{bmatrix} \theta_{2,1} \\ \theta_{3,1} \\ \theta_{3,3} \end{bmatrix}$$

Weight 1-2 Weight 2-3



  - Gradients matrices (tensors)

$$\begin{bmatrix} g_{1,1} & g_{1,2} & g_{1,3} \\ g_{2,1} & g_{2,2} & g_{2,3} \end{bmatrix} \qquad \begin{bmatrix} g_{2,1} \\ g_{3,1} \\ g_{3,3} \end{bmatrix}$$

Gradient 1-2 Gradient 2-3

  - We will simply call them weight: θ , gradient: g or (∇F).

Meaning of Training Neural Networks

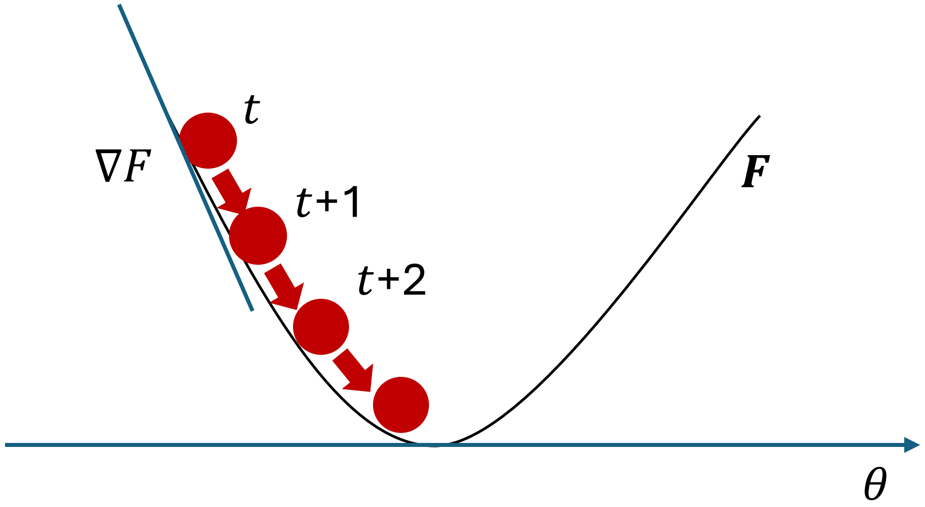
- ERM (Empirical Risk Minimization)
  - $\overline{\boldsymbol{F}}(\boldsymbol{\theta}) = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{F}(\boldsymbol{\theta}; \boldsymbol{\mathcal{D}}_{\boldsymbol{i}}(\boldsymbol{x}, \boldsymbol{y}))$

- Goal: Finding **model parameter** $\boldsymbol{\theta}$ that can minimize $\overline{\boldsymbol{F}}(\boldsymbol{\theta})$ under given dataset $\boldsymbol{\mathcal{D}}$ where objective (loss) function is $\boldsymbol{F}$ (e.g., cross entropy).

- How: Gradient Descent

Gradient Descent

- If F is **differentiable** and satisfies additional conditions such as **convexity** and **smoothness**, we can state the following:
  - Moving in the negative direction of the gradient ($\boldsymbol{\nabla F}$) at parameter $\boldsymbol{\theta}$, with a step size $\boldsymbol{\eta}$, reduces the value of $\overline{\boldsymbol{F}}(\boldsymbol{\theta})$.



GD: $$\theta^{t+1} = \theta^t - \eta\nabla F(\theta^t)$$

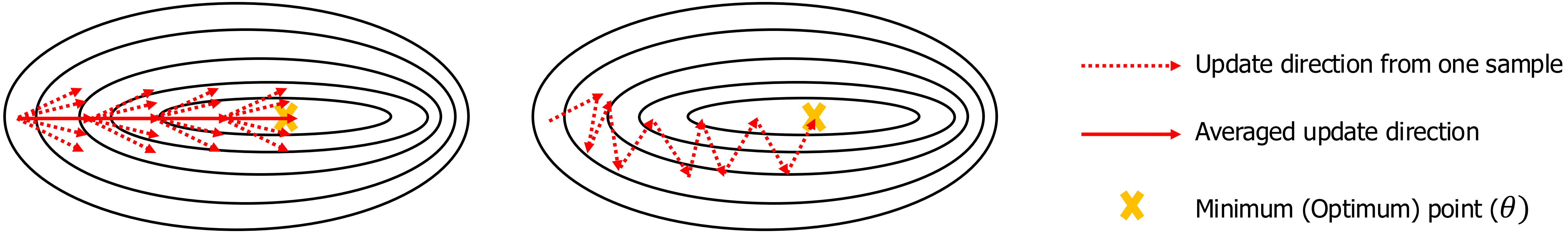
Limitation of GD

- If the number of dataset $\boldsymbol{n}$ is large, each iteration ($\boldsymbol{t}$) can take too long.
  - $\theta^{t+1} = \theta^{t} - \eta\nabla\bar{F}(\theta^{t})$
  - $\theta^{t+1} = \theta^{t} - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla F(\theta^{t};\mathcal{D}_i)$
- Because we have to **average** $\boldsymbol{\nabla F}$ for each dataset $\boldsymbol{\mathcal{D}_i}$.

Stochastic Gradient Descent (SGD)

- How about we update $\boldsymbol{\theta}$ only with **one** sample $\boldsymbol{\mathcal{D}_i}$ for every iteration?
    - $\theta^{t+1} = \theta^{t} - \eta \nabla F(\theta^{t}; \mathcal{D}_i)$ with randomly sampled $\mathcal{D}_i$

- Faster but unstable update


Update direction from one sample
Averaged update direction
Minimum (Optimum) point (θ)

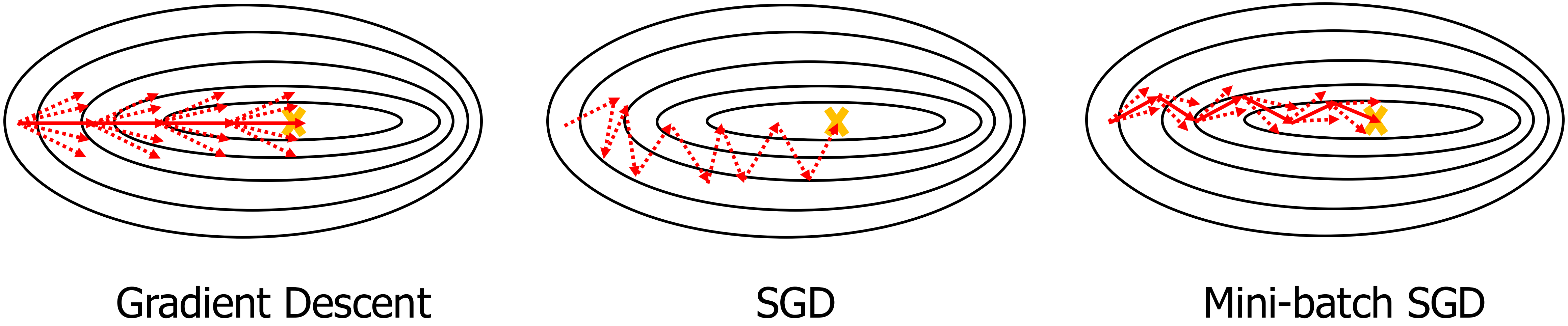
Gradient Descent

SGD

Mini-batch SGD

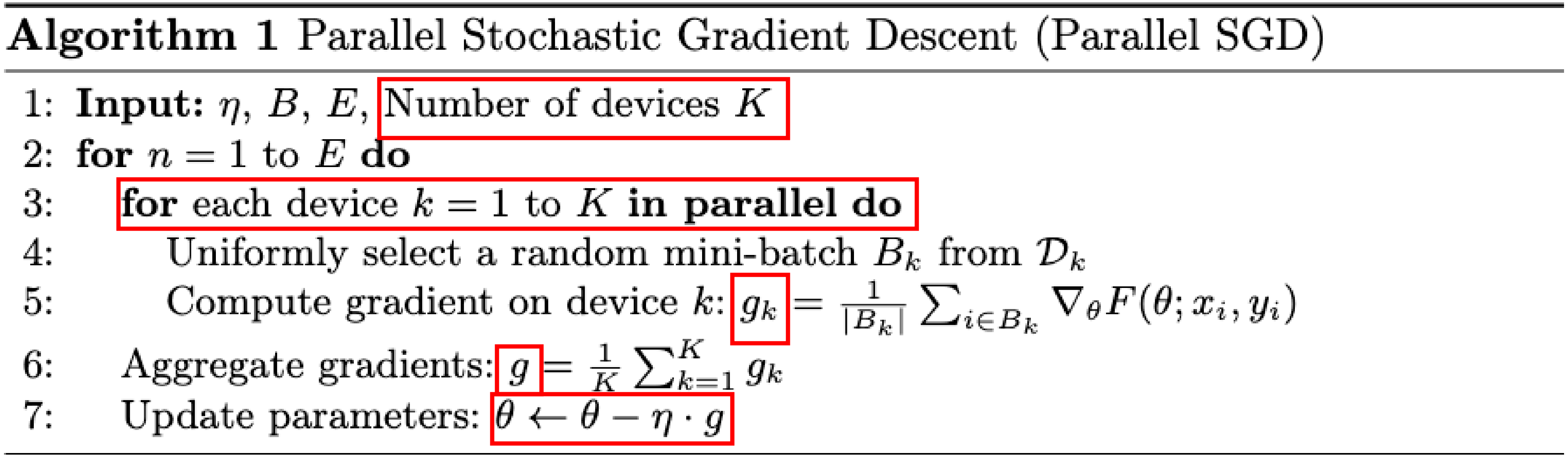
- Sample more than one
  - $\theta^{t+1} = \theta^t - \eta \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla F(\theta^t; \mathcal{D}_i)$ with randomly sampled batch $\boldsymbol{\mathcal{B}}$

- Fast and stable



Parallel SGD

- If hardware limitation is the bottleneck, can we distribute the data across **multiple machines** and calculate the gradients in **parallel**?



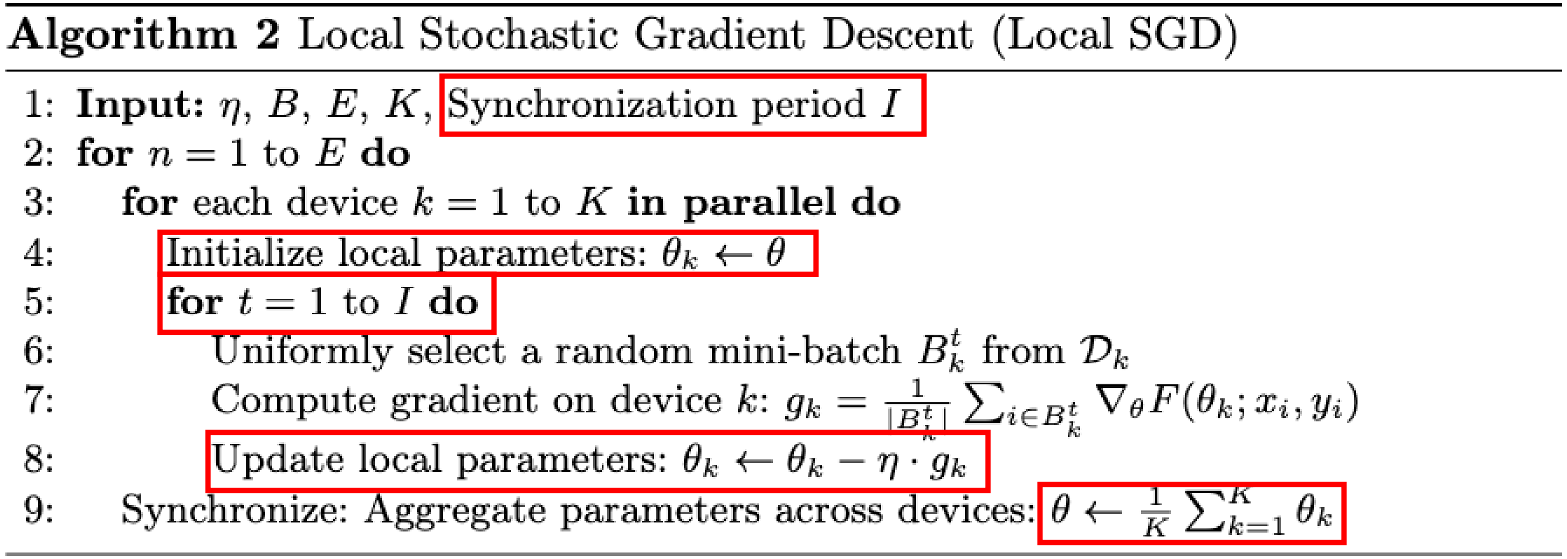
Algorithm 1 Parallel Stochastic Gradient Descent (Parallel SGD)

```
**Input:** $\eta$, $B$, $E$, Number of devices $K$
**for** $n = 1$ to $E$ **do**
    **for** each device $k = 1$ to $K$ **in parallel do**
        Uniformly select a random mini-batch $B_k$ from $\mathcal{D}_k$
        Compute gradient on device $k$: $g_k = \frac{1}{|B_k|} \sum_{i \in B_k} \nabla_\theta F(\theta; x_i, y_i)$
    Aggregate gradients: $g = \frac{1}{K} \sum_{k=1}^{K} g_k$
    Update parameters: $\theta \leftarrow \theta - \eta \cdot g$
```

- Limitation: communication overhead
- Note: It’s a layer-wise aggregation

Local SGD

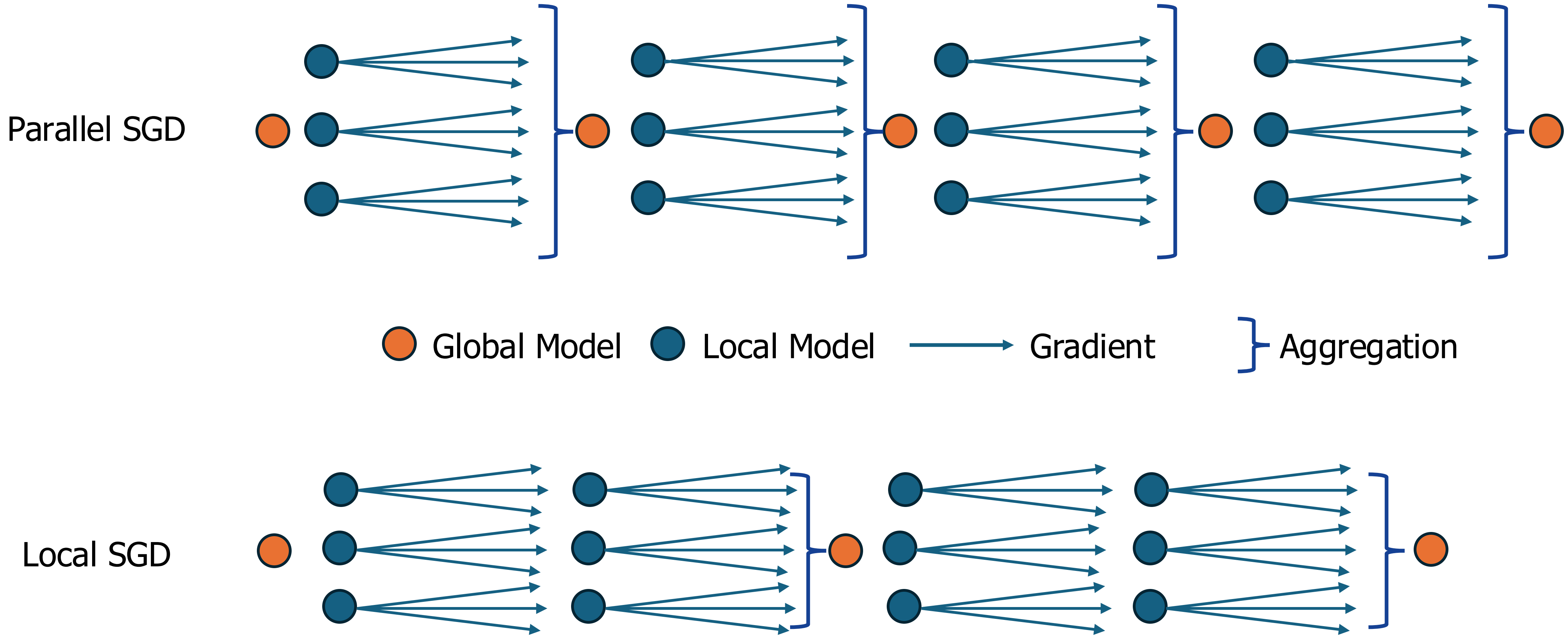
- Communicate less, locally update more.



Algorithm 2 Local Stochastic Gradient Descent (Local SGD)

```
Input: $\eta$, $B$, $E$, $K$, Synchronization period $I$
for $n = 1$ to $E$ do
    for each device $k = 1$ to $K$ in parallel do
        Initialize local parameters: $\theta_k \leftarrow \theta$
        for $t = 1$ to $I$ do
            Uniformly select a random mini-batch $B_k^t$ from $\mathcal{D}_k$
            Compute gradient on device $k$: $g_k = \frac{1}{|B_k^t|} \sum_{i \in B_k^t} \nabla_\theta F(\theta_k; x_i, y_i)$
            Update local parameters: $\theta_k \leftarrow \theta_k - \eta \cdot g_k$
    Synchronize: Aggregate parameters across devices: $\theta \leftarrow \frac{1}{K} \sum_{k=1}^{K} \theta_k$
```

Parallel SGD vs. Local SGD



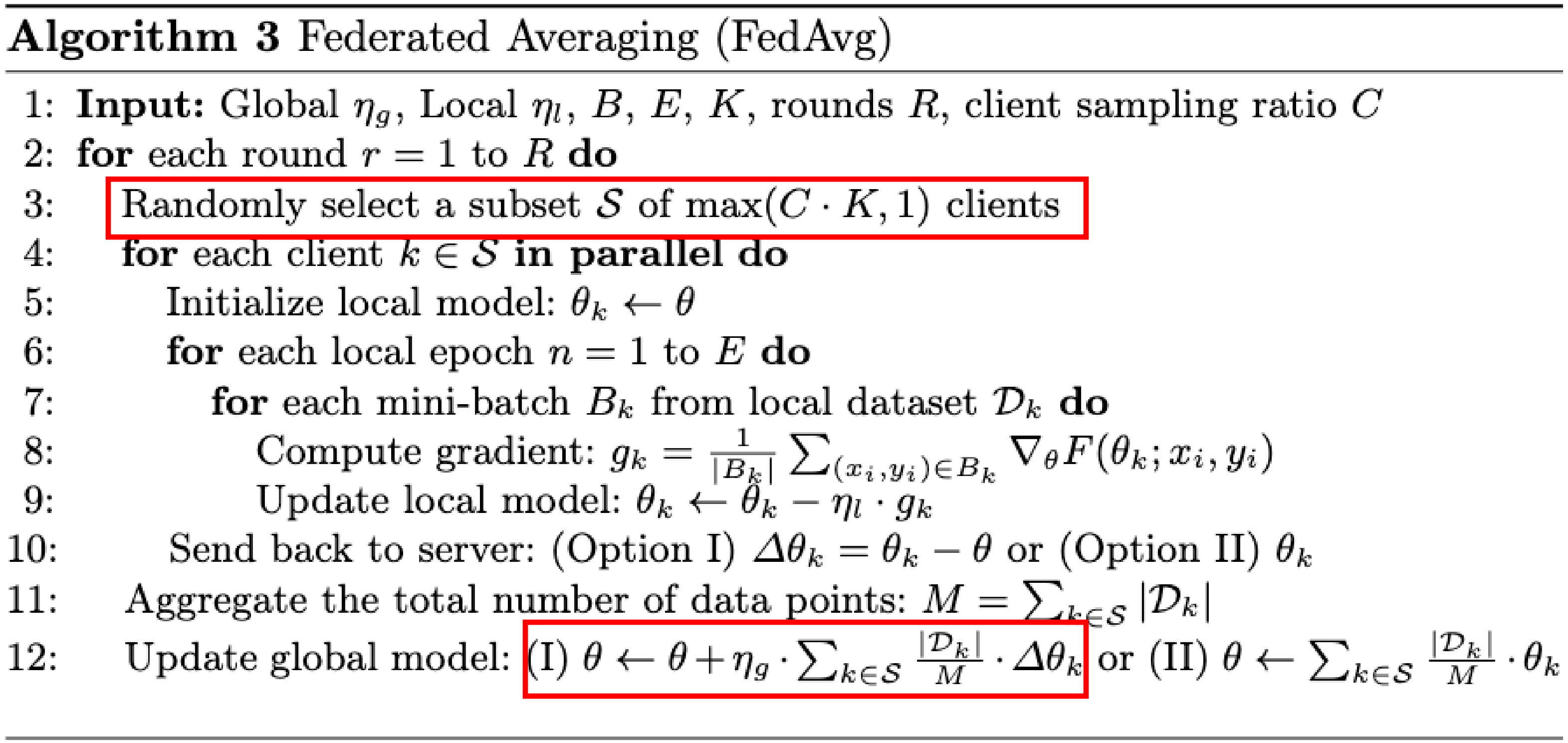
Parallel SGD

Local SGD

● Global Model ● Local Model → Gradient } Aggregation

Federated Averaging (FedAvg)

- With a consideration of “Federated”, we can have:

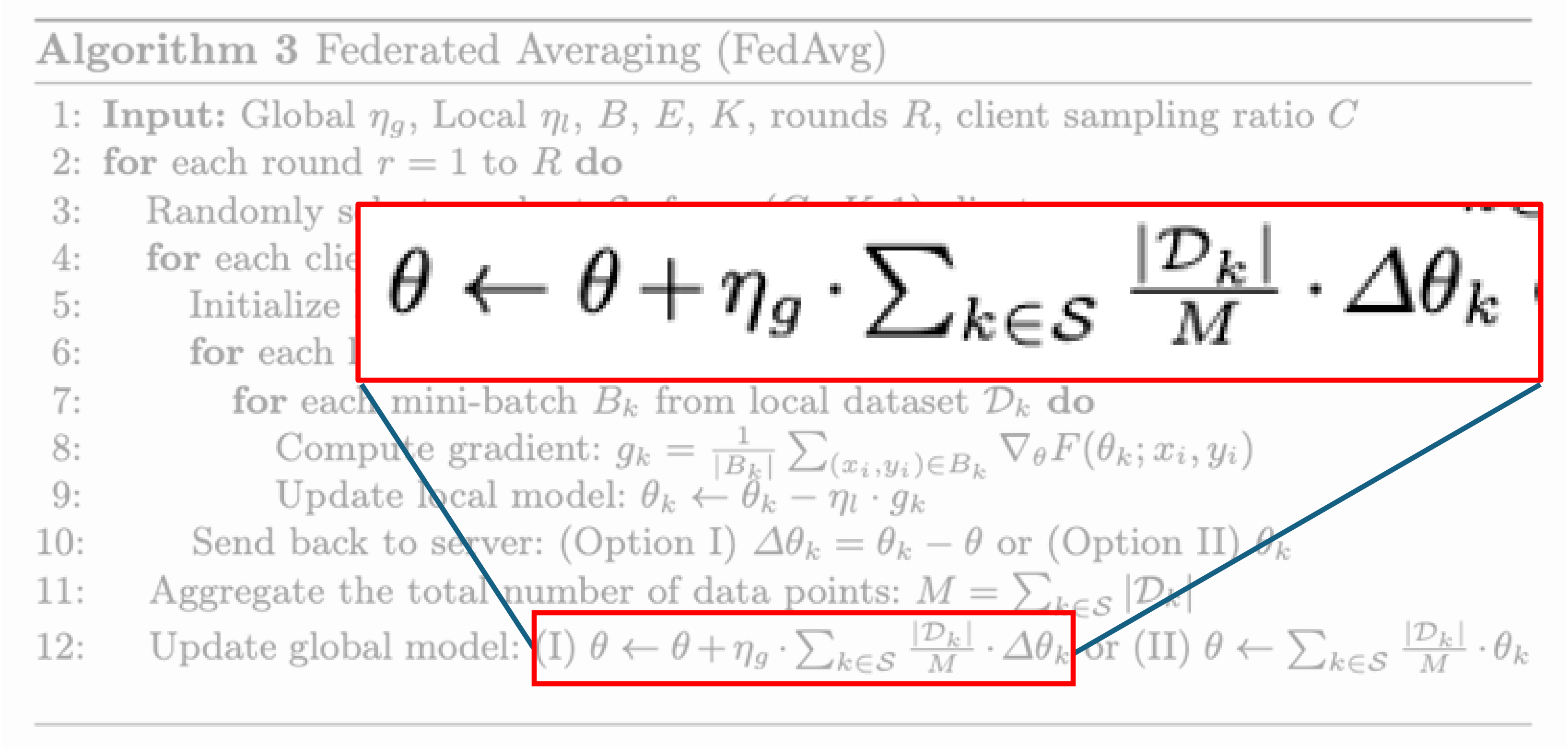


Algorithm 3 Federated Averaging (FedAvg)

```
Input: Global $\eta_g$, Local $\eta_l$, $B$, $E$, $K$, rounds $R$, client sampling ratio $C$
for each round $r = 1$ to $R$ do
    $\boxed{\text{Randomly select a subset } \mathcal{S} \text{ of } \max(C \cdot K, 1) \text{ clients}}$
    for each client $k \in \mathcal{S}$ in parallel do
        Initialize local model: $\theta_k \leftarrow \theta$
        for each local epoch $n = 1$ to $E$ do
            for each mini-batch $B_k$ from local dataset $\mathcal{D}_k$ do
                Compute gradient: $g_k = \frac{1}{|B_k|} \sum_{(x_i, y_i) \in B_k} \nabla_\theta F(\theta_k; x_i, y_i)$
                Update local model: $\theta_k \leftarrow \theta_k - \eta_l \cdot g_k$
        Send back to server: (Option I) $\Delta\theta_k = \theta_k - \theta$ or (Option II) $\theta_k$
    Aggregate the total number of data points: $M = \sum_{k \in \mathcal{S}} |\mathcal{D}_k|$
    Update global model: $\boxed{\text{(I) } \theta \leftarrow \theta + \eta_g \cdot \sum_{k \in \mathcal{S}} \frac{|\mathcal{D}_k|}{M} \cdot \Delta\theta_k}$ or (II) $\theta \leftarrow \sum_{k \in \mathcal{S}} \frac{|\mathcal{D}_k|}{M} \cdot \theta_k$
```

Federated Averaging (FedAvg)

- With a consideration of “Federated”, we can have:



Algorithm 3 Federated Averaging (FedAvg)

```
Input: Global $\eta_g$, Local $\eta_l$, $B$, $E$, $K$, rounds $R$, client sampling ratio $C$
for each round $r = 1$ to $R$ do
    Randomly s
    for each clie
        Initialize
        for each l
            for each mini-batch $B_k$ from local dataset $\mathcal{D}_k$ do
                Compute gradient: $g_k = \frac{1}{|B_k|} \sum_{(x_i, y_i) \in B_k} \nabla_\theta F(\theta_k; x_i, y_i)$
                Update local model: $\theta_k \leftarrow \theta_k - \eta_l \cdot g_k$
        Send back to server: (Option I) $\Delta\theta_k = \theta_k - \theta$ or (Option II) $\theta_k$
    Aggregate the total number of data points: $M = \sum_{k \in \mathcal{S}} |\mathcal{D}_k|$
    Update global model: (I) $\theta \leftarrow \theta + \eta_g \cdot \sum_{k \in \mathcal{S}} \frac{|\mathcal{D}_k|}{M} \cdot \Delta\theta_k$ or (II) $\theta \leftarrow \sum_{k \in \mathcal{S}} \frac{|\mathcal{D}_k|}{M} \cdot \theta_k$
```

$$\theta \leftarrow \theta + \eta_g \cdot \sum_{k \in \mathcal{S}} \frac{|\mathcal{D}_k|}{M} \cdot \Delta\theta_k$$

How does simple averaging even work?

- Global model update can be expressed in GD style.
  - GD: $\theta^{t+1} = \theta^{t} - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla F(\theta^{t}; \mathcal{D}_i)$
  - FedAvg (original) : $\theta^{t+1} = \frac{1}{k} \sum_{i=1}^{k} \theta_k^t$
  - FedAvg (modified) : $\boldsymbol{\theta^{t+1}} = \boldsymbol{\theta^{t}} + \boldsymbol{\eta_g} \boldsymbol{\frac{1}{k} \sum_{i=1}^{k} \Delta\theta_k^t}$ $\textbf{(where } \boldsymbol{\eta_g} = \mathbf{1})$

- Answer:

How does simple averaging even work?

- Global model update can be expressed in GD style.
  - GD: $\theta^{t+1} = \theta^t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla F(\theta^t; \mathcal{D}_i)$
  - ~~FedAvg (original) : $\theta^{t+1} = \frac{1}{k} \sum_{i=1}^{k} \theta_k^t$~~
  - FedAvg (modified) : $\theta^{t+1} = \theta^t + \eta_g \frac{1}{k} \sum_{i=1}^{k} \Delta\theta_k^t$
- Answer: It has always been simple averaging.

Experiments and Analysis

Experiment Goal: Identifying the impact of ..

- Transition from Centralized to Federated Learning

- Hyperparameters in Traditional Machine Learning

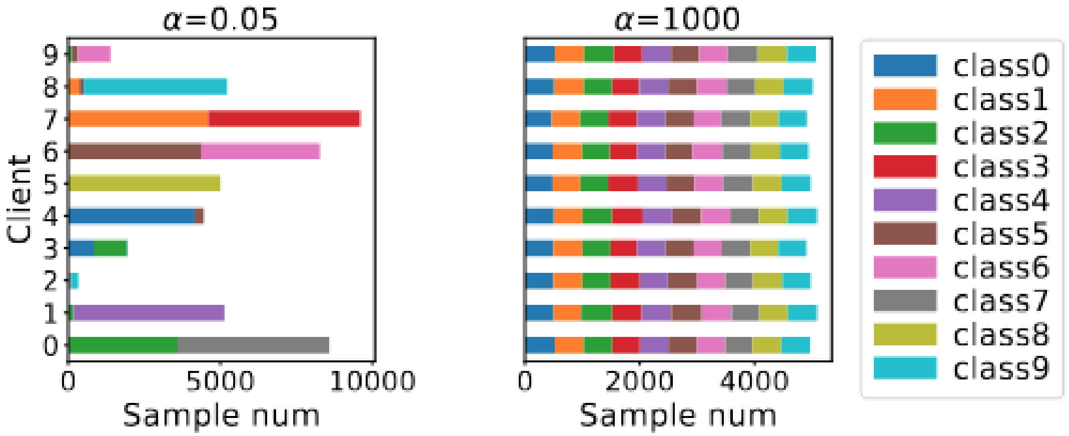
- Federated Learning Environmental Parameters

Experimental Setting

- **Model**: CNN (two convolutional layers with two fully connected layers)
- **Dataset**: CIFAR-10 (50,000 training data, 10,000 test data, 10 labels)
- **Evaluation**
  - Training loss: locally measured
  - Test accuracy/loss: globally measured
- **Hyperparameters**: Epoch, Learning rate, Batch size
- **FL experimental parameters**
  - Total number of client (K), participation ratio, non-IIDness, volume distribution
- **Data Partitioning**
  - Distribute 50,000 training data samples across K clients, ensuring no overlap.

Data Heterogeneity in FL

- If we have control over the local machines:
  - Dataset can be well distributed across machines.
  - **Independent and identically distributed (*i.i.d.*)**
- In FL, we do/should not know the distribution of dataset.
  - In real world, distribution of dataset can be (very) different (**non-*i.i.d.***).

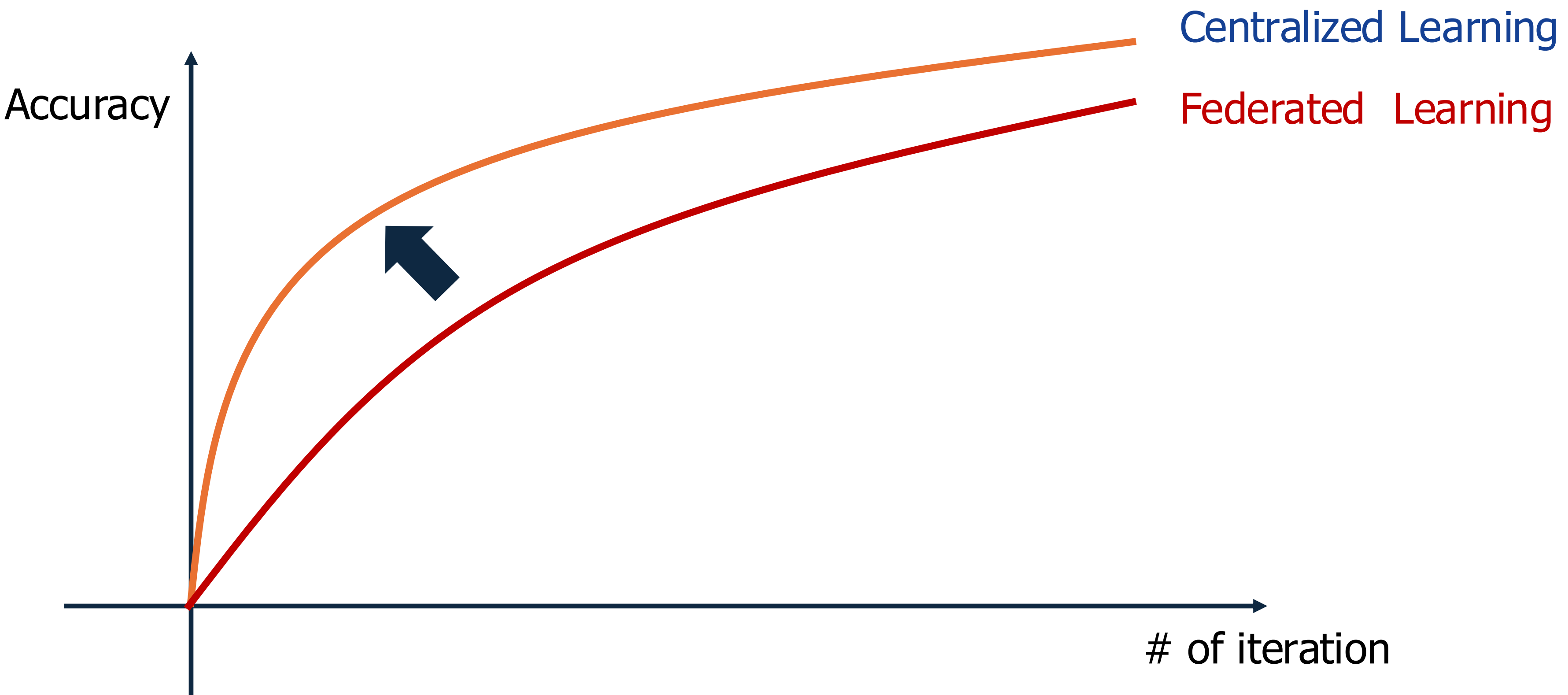


Non- i.i.d.

i.i.d.

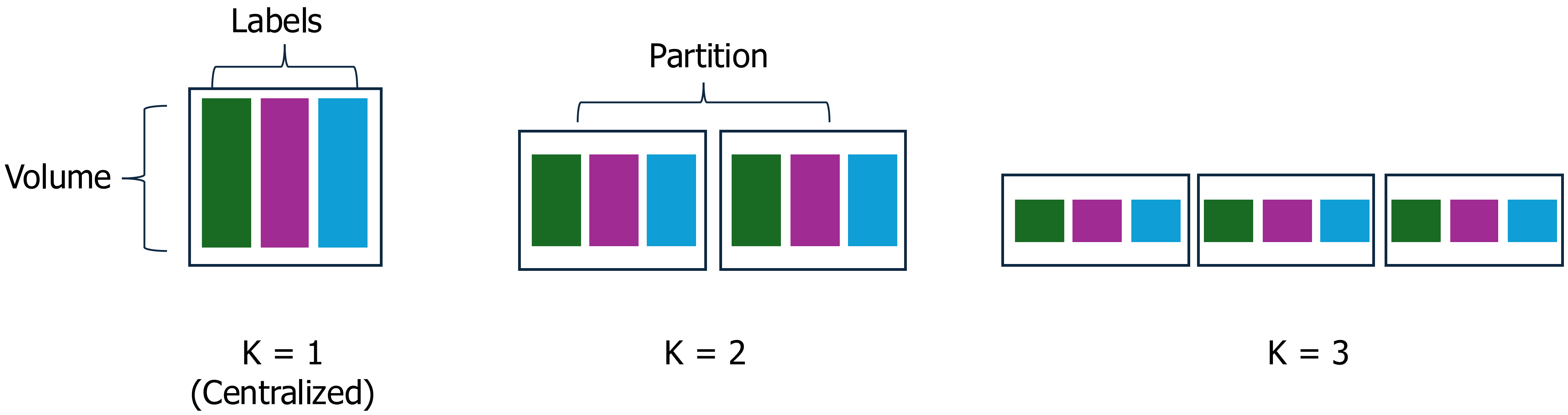
From Centralized to Federated Learning

- Successful FL is determined by how similar **convergence** is to centralize learning when using the same data set.



From Centralized to Federated Learning

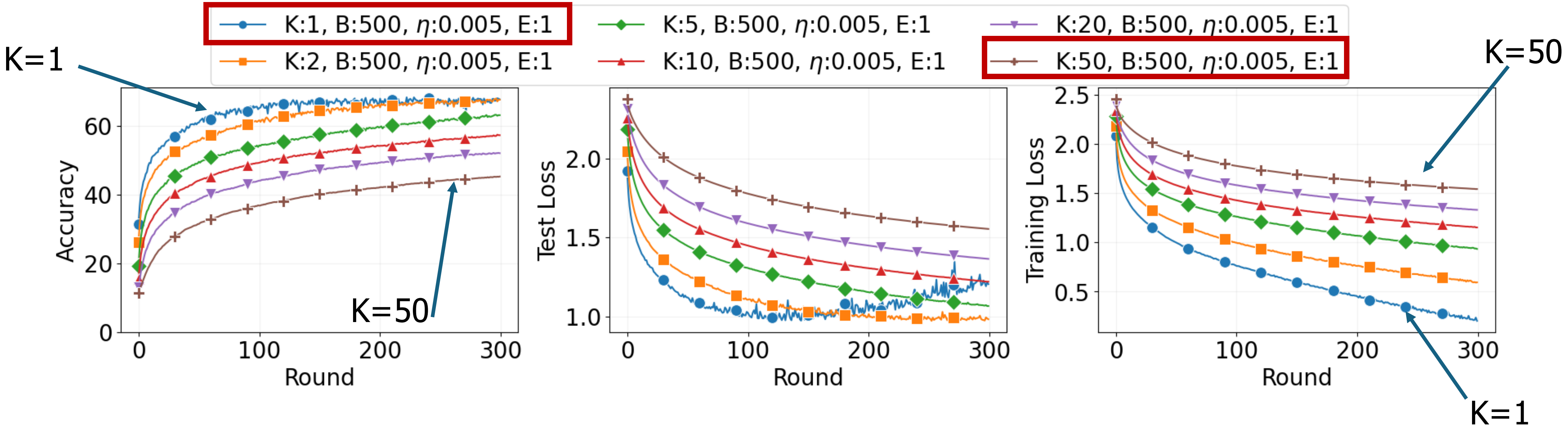
- Question: Does the number of clients influence performance (IID)?



Dataset Partition over K clients

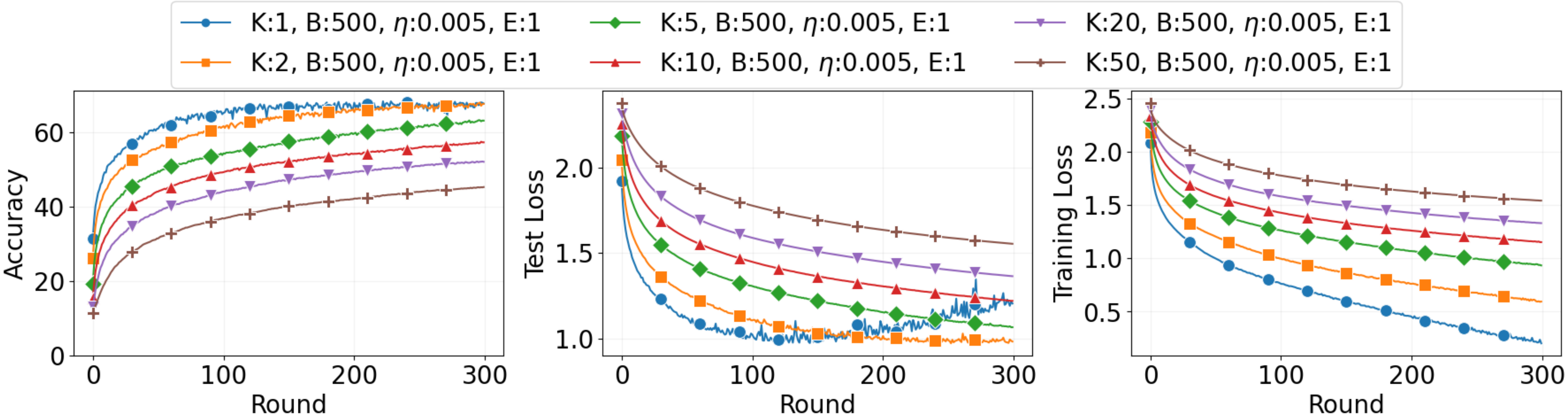
From Centralized to Federated Learning

- Question: Does the number of clients influence performance?
- Answer: Yes..?



From Centralized to Federated Learning

- Question: Does the number of clients influence performance?
- Answer: Yes..? Only if you do not consider the effective update.



From Centralized to Federated Learning

- When the dataset is **IID** and volume is **balanced**, the ***effective update*** per round can be defined:

$$u = \eta \cdot \frac{E \cdot |\mathcal{D}|}{B \cdot K}$$

  - η: learning rate
  - E: local epoch
  - K: number of client
  - B: Batch size
  - $|\mathcal{D}|$: Total number of data

From Centralized to Federated Learning

- When the dataset is **IID** and volume is **balanced**, the ***effective update*** per round can be defined:

$$u = \eta \cdot \frac{E \cdot |\mathcal{D}|}{B \cdot K}$$

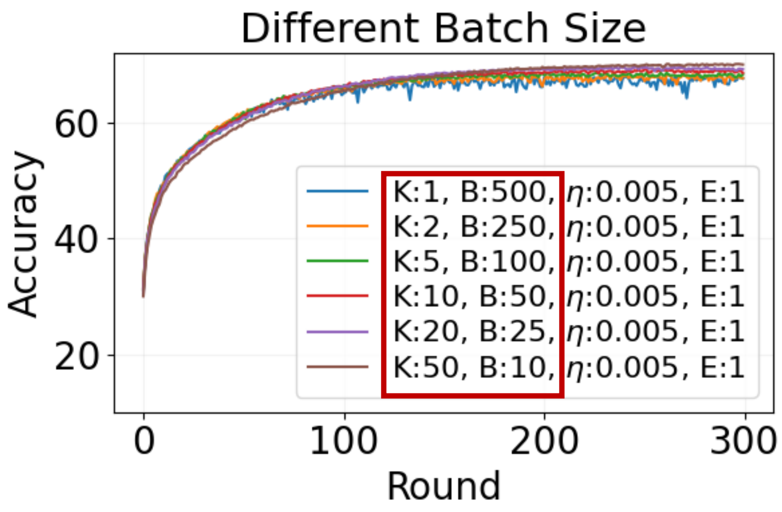
- If everything is fixed except $\boldsymbol{K}$
  - K increases u decreases: Less (smaller) update

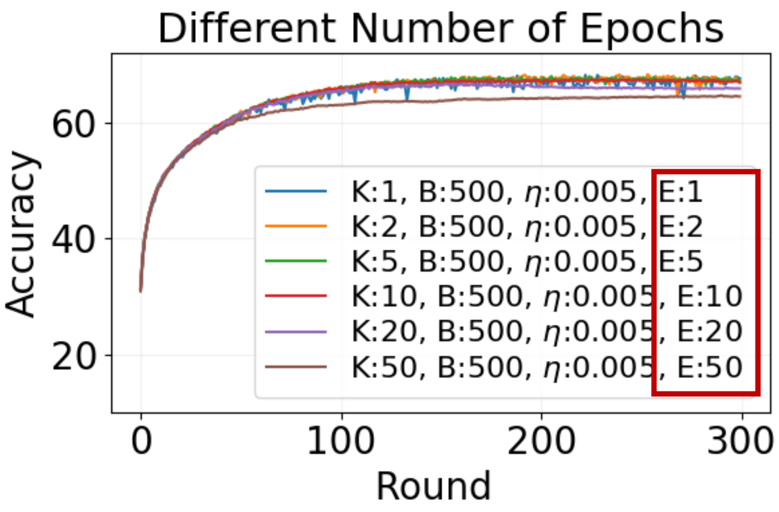
- *Beware of this pitfall!*

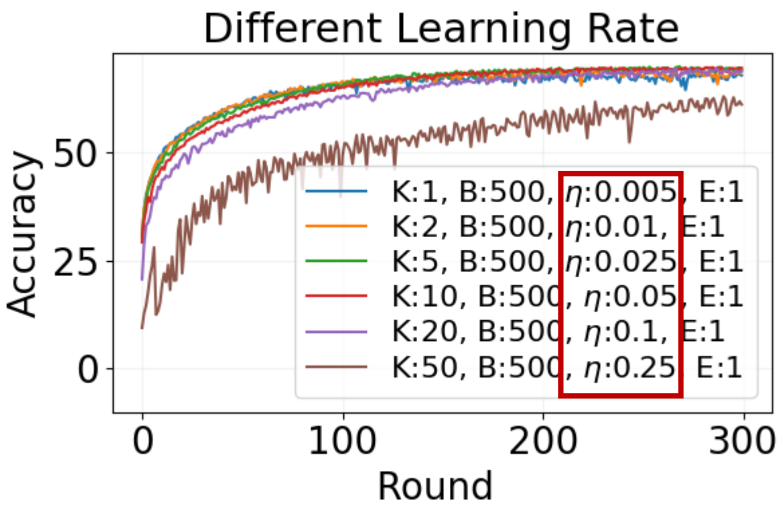
From Centralized to Federated Learning

- Question: Does the number of clients influence performance?
- Answer: It shouldn’t, as long as we have equal **u.**

$$u = \eta \cdot \frac{E \cdot |\mathcal{D}|}{B \cdot K}$$







Impact of Hyperparameters under IID

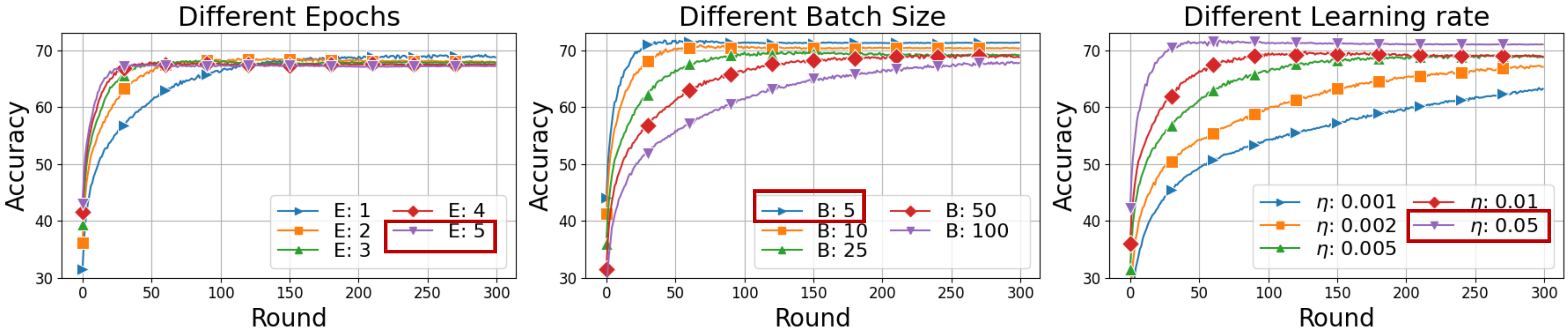
$$u = \eta \cdot \frac{E \cdot |\mathcal{D}|}{B \cdot K}$$

- **Question**: For the faster convergence we need,
  - Epoch: High vs. Low
  - Batch size: High vs. Low
  - Learning rate: High vs. Low

Impact of Hyperparameters under IID

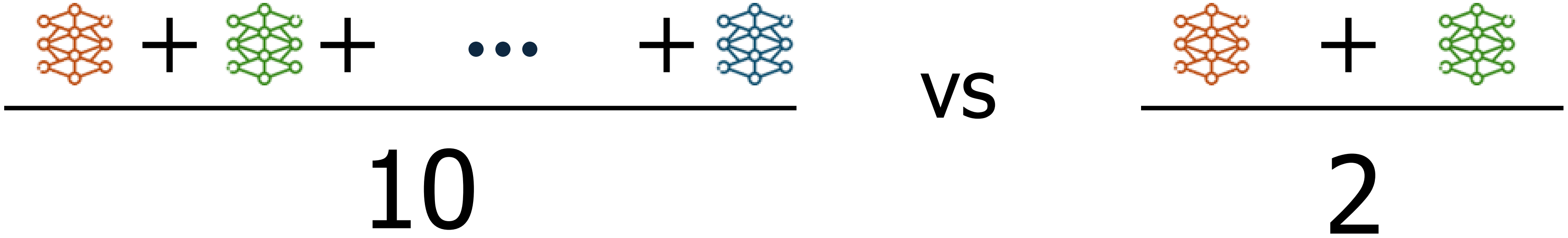
$$u = \eta \cdot \frac{E \cdot |\mathcal{D}|}{B \cdot K}$$

- **Question**: For the faster convergence we need,
  - Epoch: **High** vs. Low
  - Batch size: High vs. **Low**
  - Learning rate: **High** vs. Low



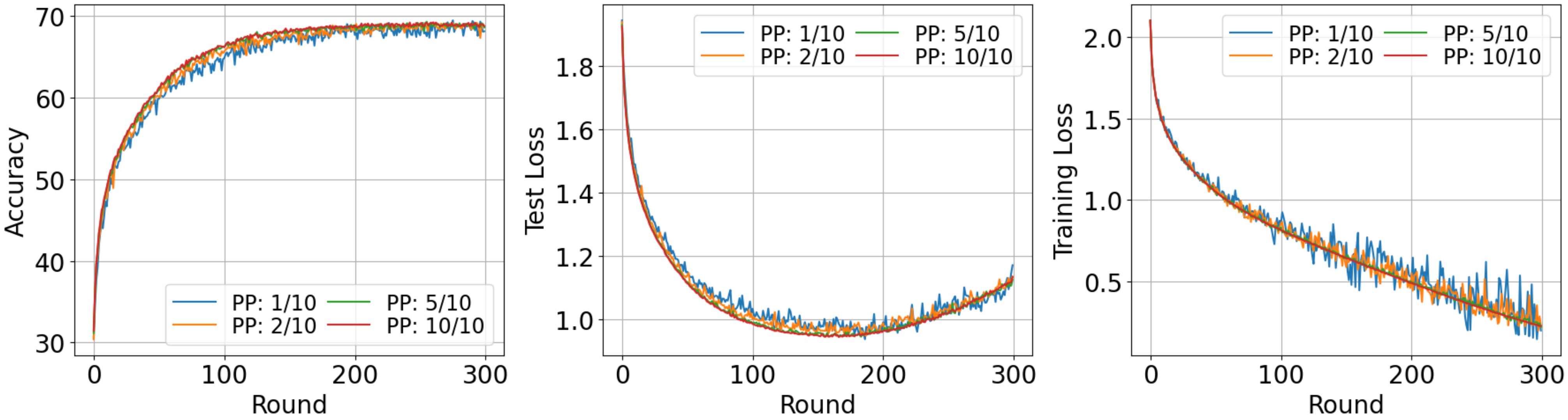
Impact of partial participation (PP) under IID

- **Question:** Does the number of participants influence performance?
- **Answer:**



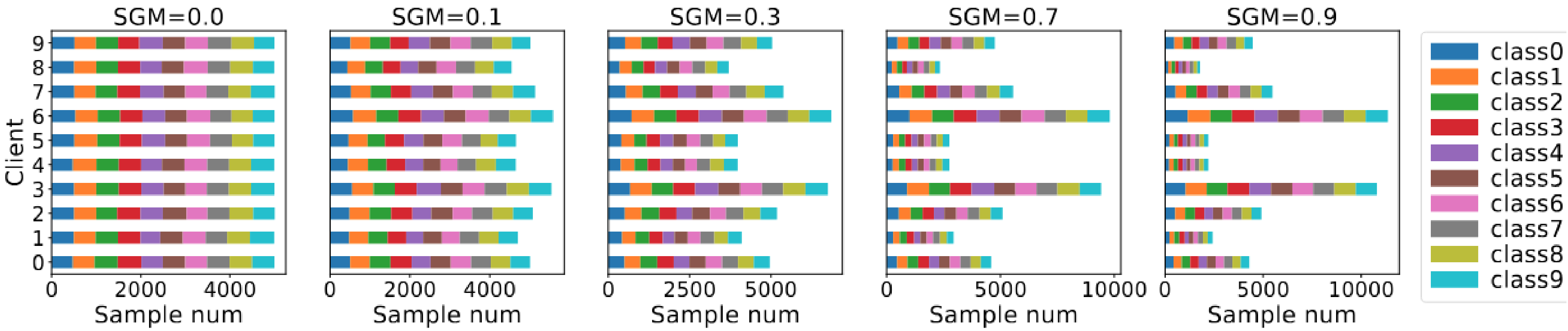
Impact of partial participation (PP) under IID

- **Question**: Does the number of participation influence performance?
- **Answer:** Not that much
- **Interpretation:** Updates ($\Delta\theta_k^t$) from each client are very similar.



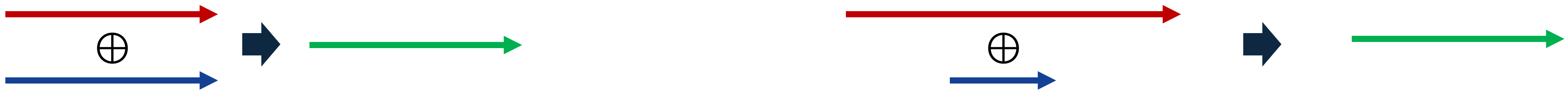
Impact of Volume Imbalance

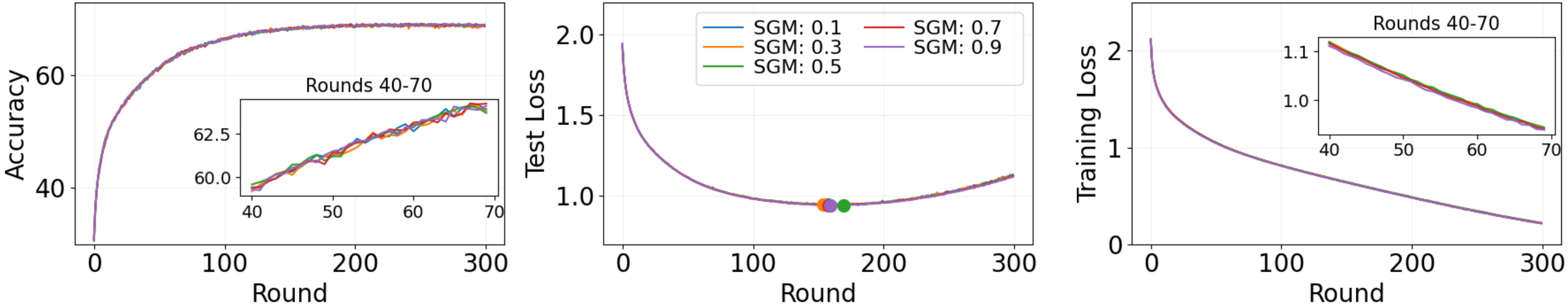
- **Question:** Does volume imbalance impact performance when the label distribution is consistent?



Impact of Volume Imbalance

- **Question**: Does volume imbalance impact performance when the label distribution is consistent?
- **Answer: No!** (for uniform aggregation $\frac{1}{k}\sum_{i=1}^{k}\Delta\theta_k^t$)





Summary of IID experiments

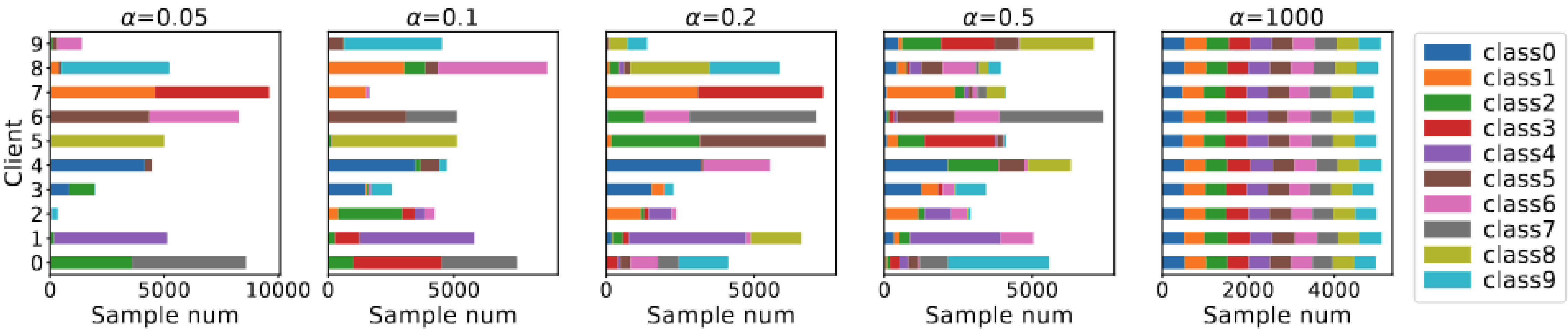
- **In IID Settings:**
  - Higher update amount (number) accelerates convergence.
  - Partial client participation does not impact convergence.
  - Imbalance in data volume does not impact convergence.

- **Takeaway:**
  - When the label distribution is IID, each client update acts like a **vector in the same direction.**

Non-IID Experiments

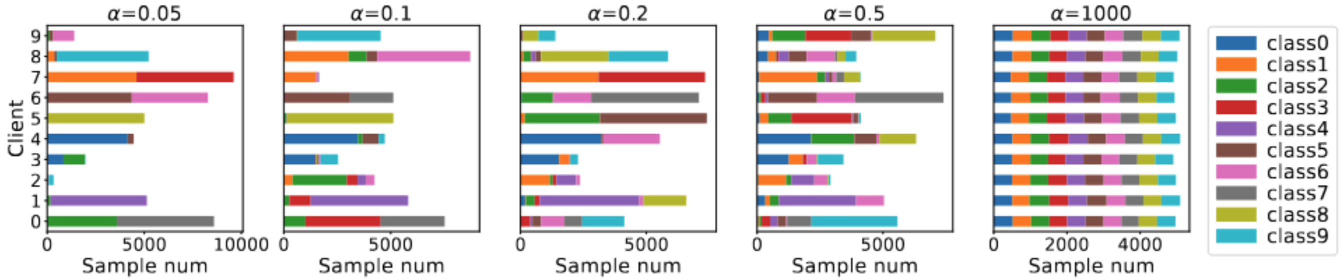
Impact of the level of data heterogeneity

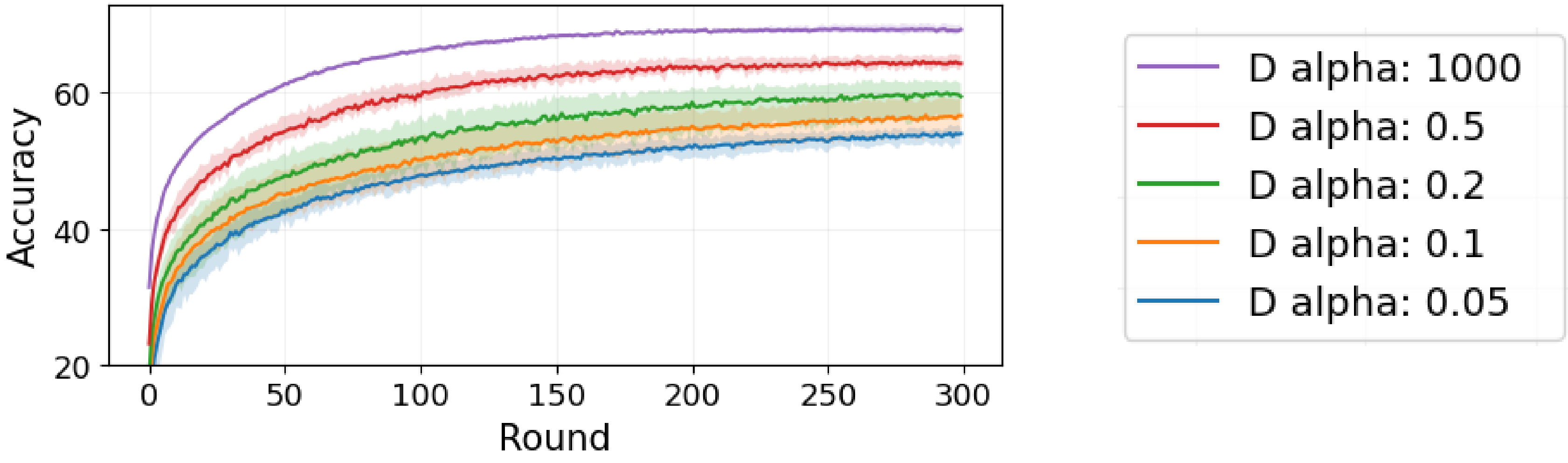
- **Question**: Does the level of data heterogeneity affect convergence?
- **Answer:?**



Impact of the level of data heterogeneity

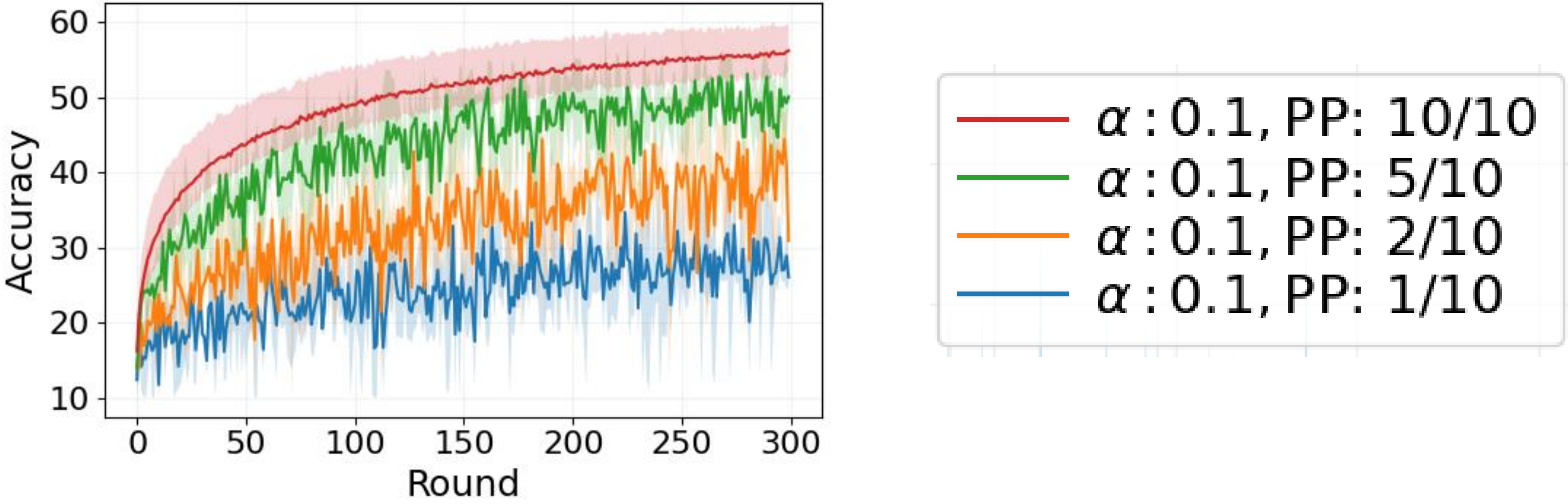
- **Question**: Does the level of data heterogeneity affect convergence?
- **Answer: YES.**





Impact of partial participation (Non-IID)

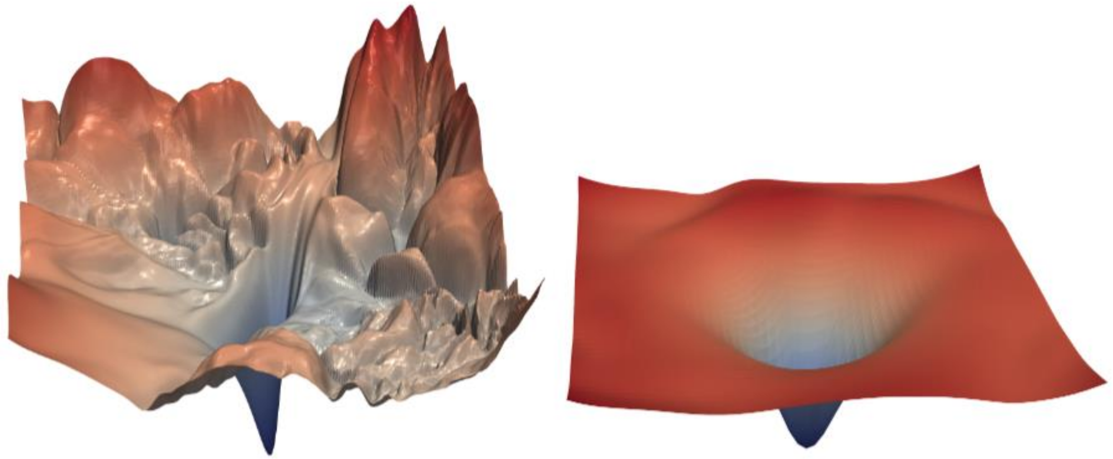
- **Question**: Does partial participation affect convergence?
- **Answer: A lot.**



Understanding FL from Loss Landscape Perspective

Loss Landscape

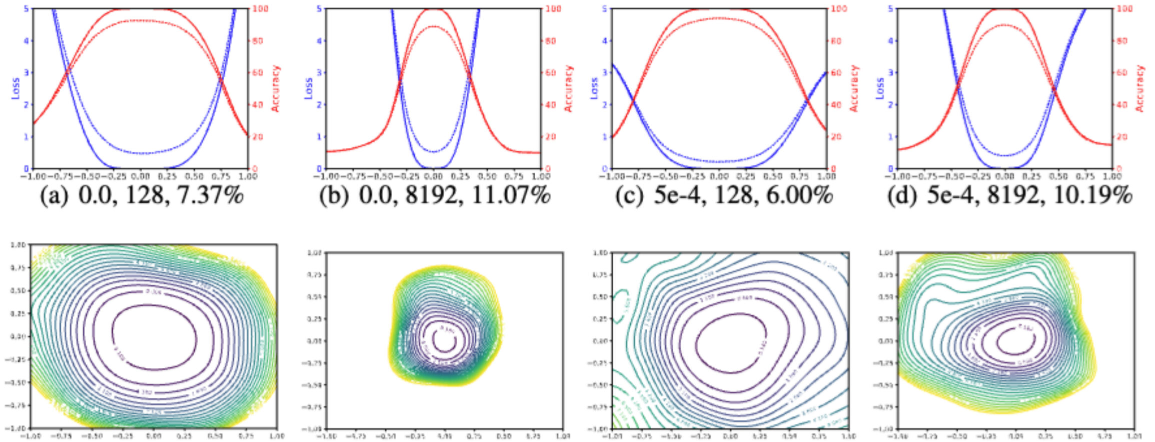
- The **loss landscape** is a visualization of how a model's error (loss F) changes across its parameter (θ) space.



(a) without skip connections

(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.



(a) 0.0, 128, 7.37%

(b) 0.0, 8192, 11.07%

(c) 5e-4, 128, 6.00%

(d) 5e-4, 8192, 10.19%

(e) 0.0, 128, 7.37%

(f) 0.0, 8192, 11.07%

(g) 5e-4, 128, 6.00%

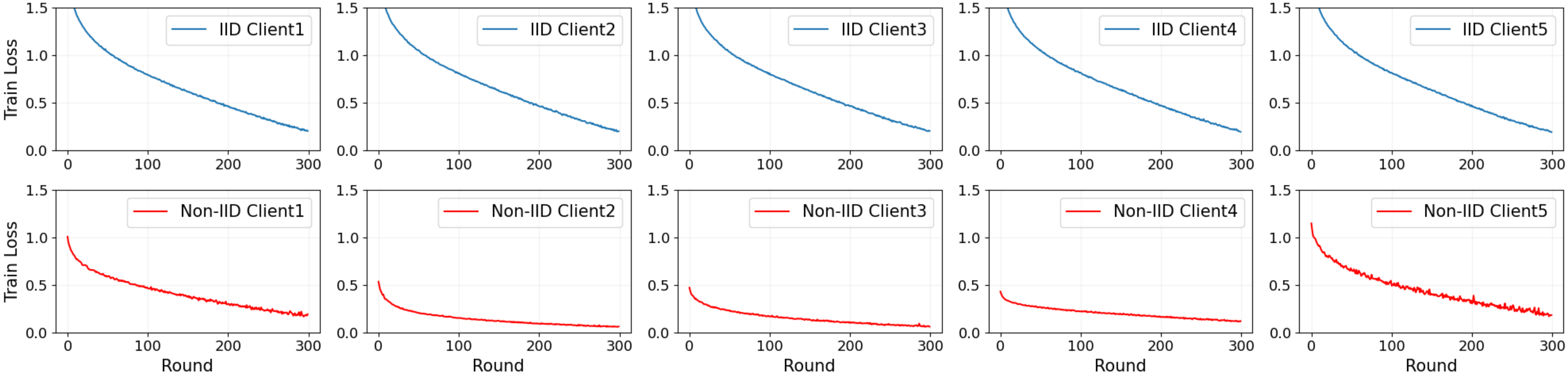
(h) 5e-4, 8192, 10.19%

Figure 3: The 1D and 2D visualization of solutions obtained using SGD with different weight decay and batch size. The title of each subfigure contains the weight decay, batch size, and test error.

Li, Hao, et al. "Visualizing the loss landscape of neural nets." Advances in neural information processing systems 31 (2018).

Client-wise Training Loss values

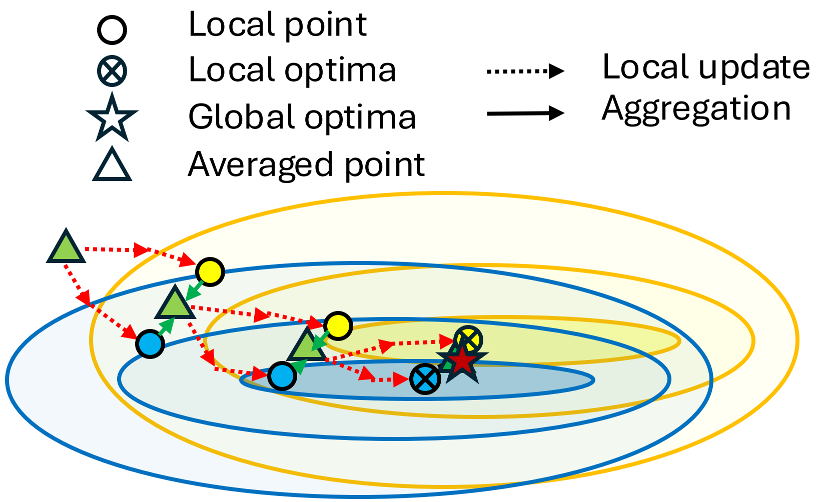
- In our experiment, we observed that under IID conditions, each client exhibits similar loss values.
- Conversely, in non-IID settings, the loss values vary significantly across clients.



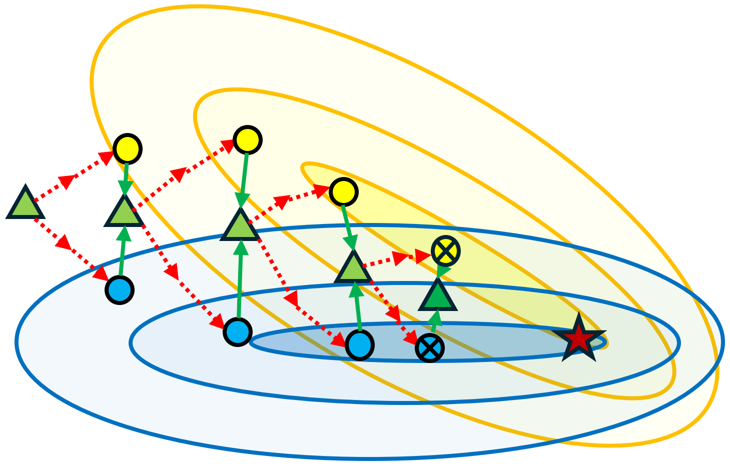
FL from Loss Landscape Perspective

- IID
  - Similar loss landscape
  - Similar update direction
  - Overlapping optimal region

- Non-IID
  - Distinct loss landscape
  - Distinct update direction (client drift)
  - Divergent optimal region



IID



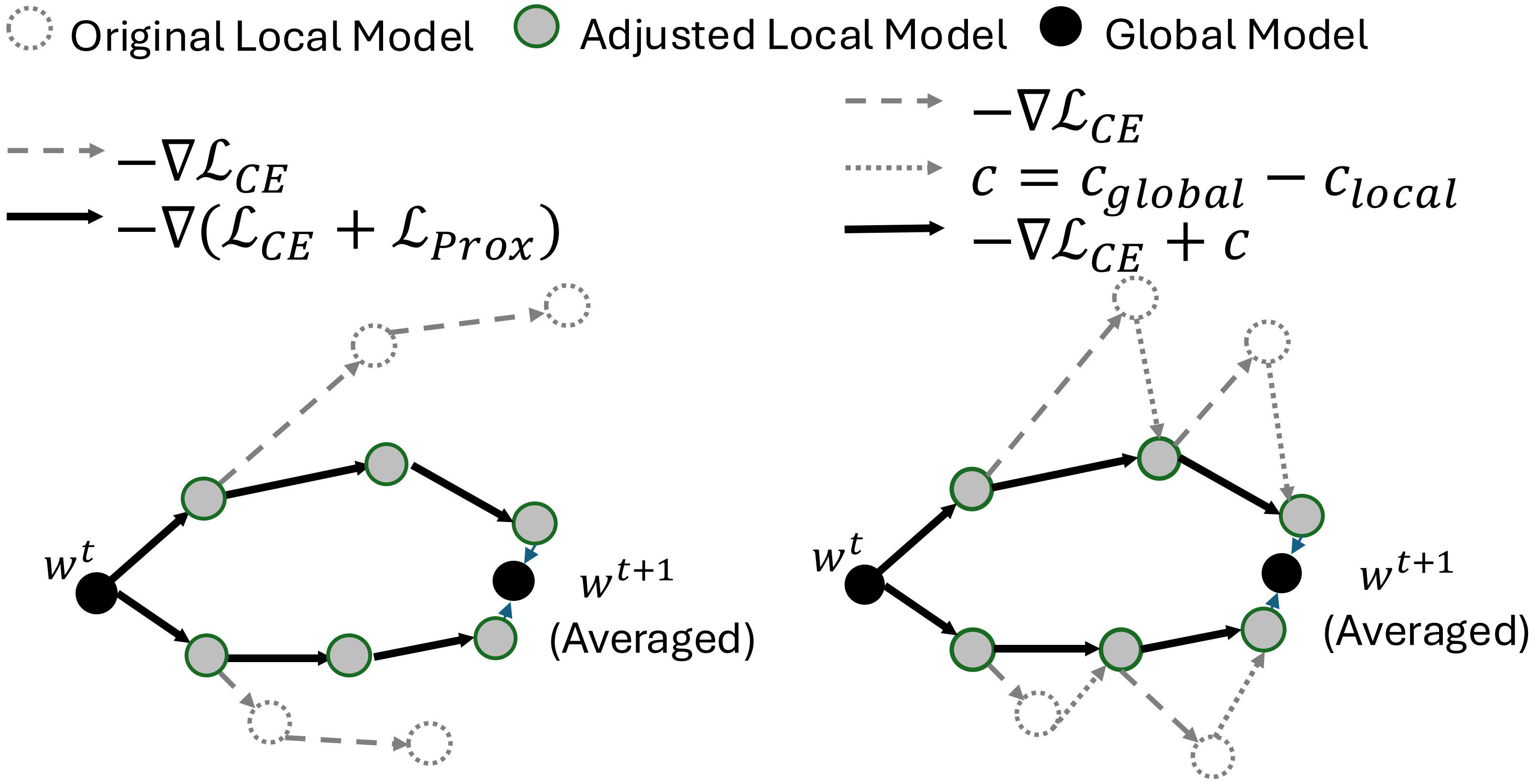
Non-IID

How to improve FL under Non-IID condition

- **Adjusting the update paths**: Make them have similar direction
    - Explicit update alignment
- **Modifying the loss landscape**: Make them follow the similar landscape
    - Implicit update alignment
- Or both

- We need some creativity!

Beyond FedAvg



○ Original Local Model ● Adjusted Local Model ● Global Model

$-\nabla\mathcal{L}_{CE}$

$-\nabla(\mathcal{L}_{CE} + \mathcal{L}_{Prox})$

$-\nabla\mathcal{L}_{CE}$

$c = c_{global} - c_{local}$

$-\nabla\mathcal{L}_{CE} + c$

w^t

w^{t+1} (Averaged)

w^t

w^{t+1} (Averaged)

FedProx

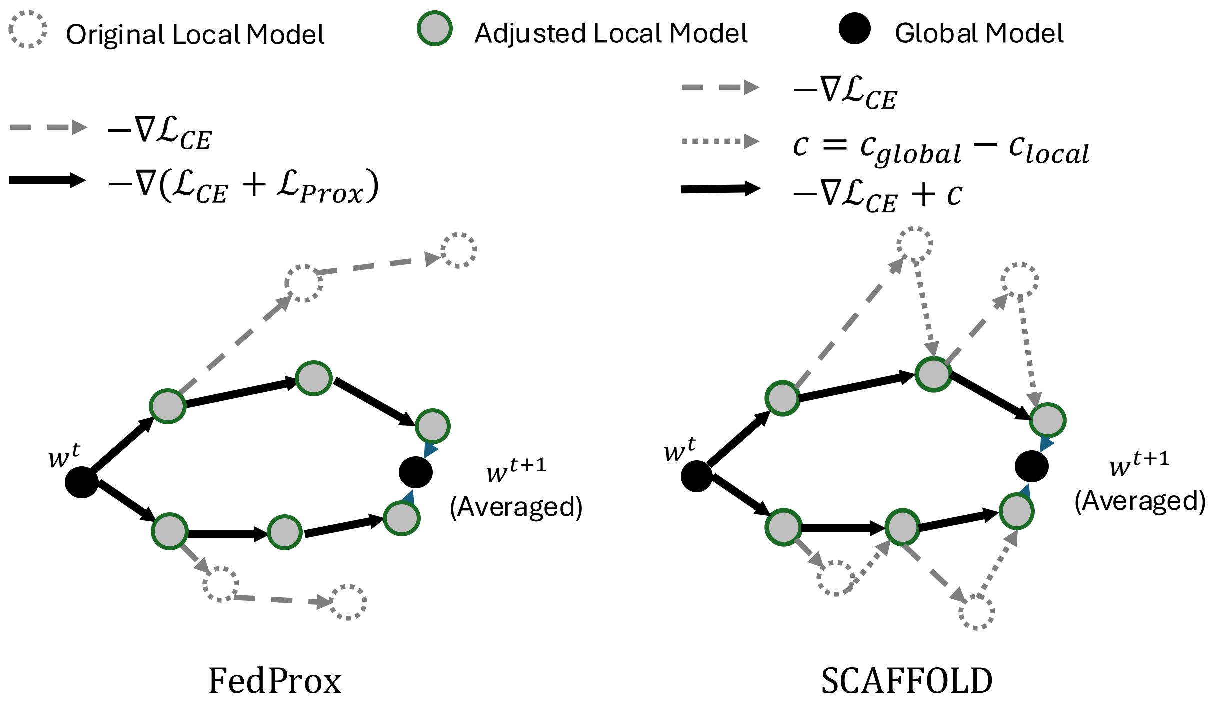
SCAFFOLD

Li, Tian, et al. "Federated optimization in heterogeneous networks." Proceedings of Machine learning and systems 2 (2020): 429-450.

Karimireddy, Sai Praneeth, et al. "Scaffold: Stochastic controlled averaging for federated learning." International conference on machine learning. PMLR, 2020.

Beyond FedAvg

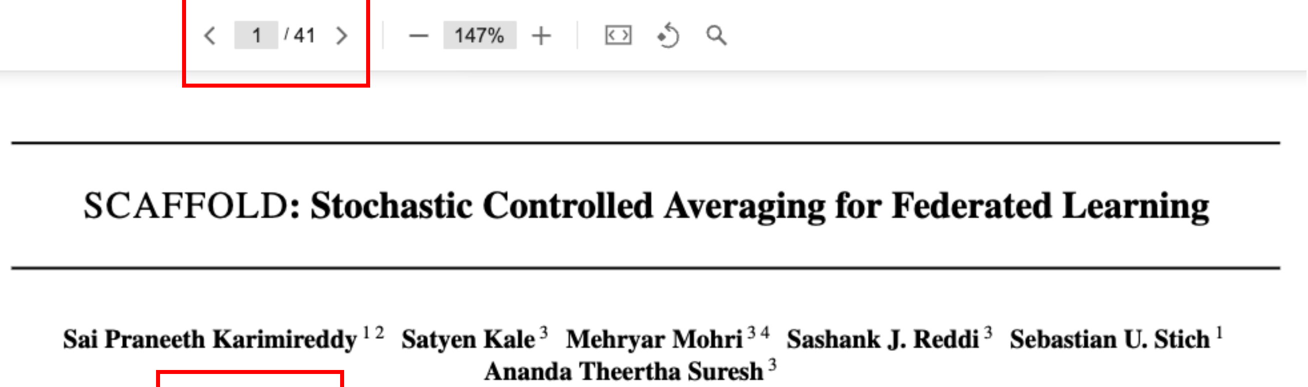
- FedProx
  - $F(\theta) = \mathcal{L}_{CE}(\theta) + \frac{\mu}{2}\|\theta - \theta^t\|_2$
  - $\mathcal{L}_{CE}$: Cross-Entropy loss
  - θ^t: Global model parameter
  - θ : Current local model parameter
- SCAFFOLD
  - $-\nabla F(\theta)$ = $-\nabla\mathcal{L}_{CE} + c$
  - $c = c_{global} - c_{local}$
  - c: control variate (gradient matrix)



Original Local Model
Adjusted Local Model
Global Model
$-\nabla\mathcal{L}_{CE}$
$-\nabla(\mathcal{L}_{CE} + \mathcal{L}_{Prox})$
$-\nabla\mathcal{L}_{CE}$
$c = c_{global} - c_{local}$
$-\nabla\mathcal{L}_{CE} + c$
w^t
w^{t+1}
(Averaged)
w^t
w^{t+1}
(Averaged)
FedProx
SCAFFOLD

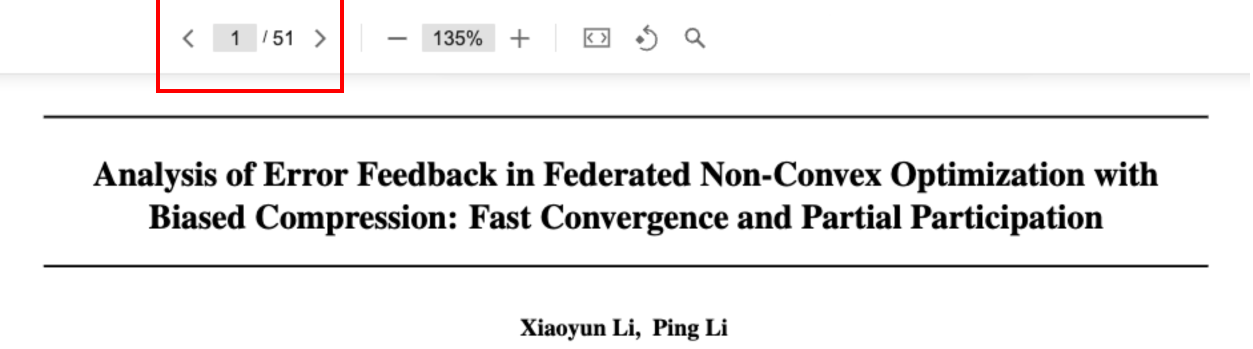
To Design New FL Algorithms

- We also need some math!
  - Convergence Analysis



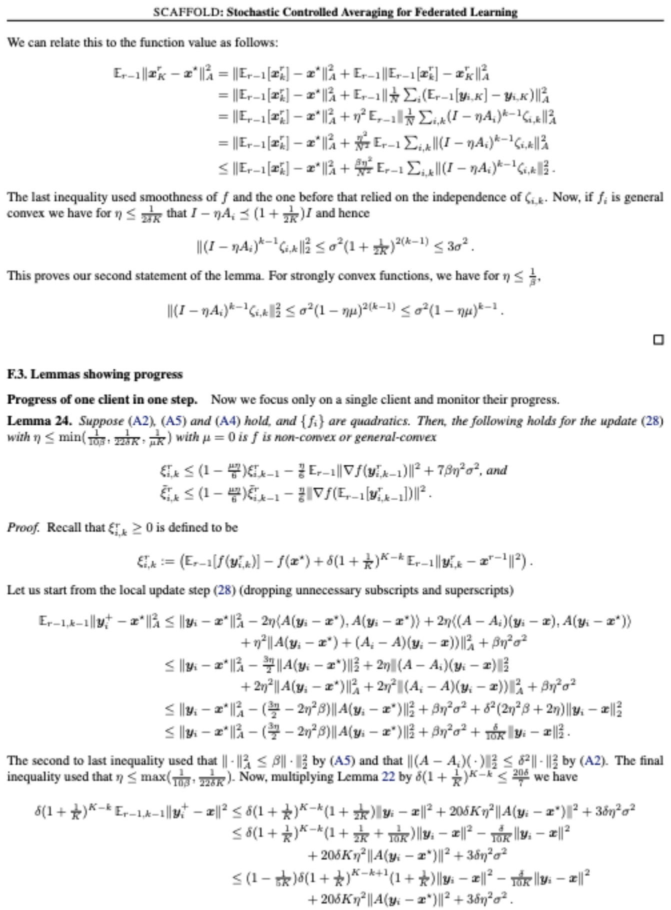
SCAFFOLD: **Stochastic Controlled Averaging for Federated Learning**

Sai Praneeth Karimireddy [1 2] Satyen Kale [3] Mehryar Mohri [3 4] Sashank J. Reddi [3] Sebastian U. Stich [1]
Ananda Theertha Suresh [3]



Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression: Fast Convergence and Partial Participation

Xiaoyun Li, Ping Li



SCAFFOLD: **Stochastic Controlled Averaging for Federated Learning**

We can relate this to the function value as follows:

$$\begin{aligned}\mathbb{E}_{r-1}\|\boldsymbol{x}_K^r - \boldsymbol{x}^\star\|_A^2 &= \|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}^\star\|_A^2 + \mathbb{E}_{r-1}\|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}_K^r\|_A^2\\ &= \|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}^\star\|_A^2 + \mathbb{E}_{r-1}\|\tfrac{1}{N}\textstyle\sum_i(\mathbb{E}_{r-1}[\boldsymbol{y}_{i,K}] - \boldsymbol{y}_{i,K})\|_A^2\\ &= \|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}^\star\|_A^2 + \eta^2\,\mathbb{E}_{r-1}\|\tfrac{1}{N}\textstyle\sum_{i,k}(I-\eta A_i)^{k-1}\zeta_{i,k}\|_A^2\\ &= \|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}^\star\|_A^2 + \tfrac{\eta^2}{N^2}\,\mathbb{E}_{r-1}\textstyle\sum_{i,k}\|(I-\eta A_i)^{k-1}\zeta_{i,k}\|_A^2\\ &\le \|\mathbb{E}_{r-1}[\boldsymbol{x}_k^r] - \boldsymbol{x}^\star\|_A^2 + \tfrac{\beta\eta^2}{N^2}\,\mathbb{E}_{r-1}\textstyle\sum_{i,k}\|(I-\eta A_i)^{k-1}\zeta_{i,k}\|_2^2\,.\end{aligned}$$

The last inequality used smoothness of f and the one before that relied on the independence of $\zeta_{i,k}$. Now, if f_i is general convex we have for $\eta \le \frac{1}{2\delta K}$ that $I - \eta A_i \preceq (1 + \frac{1}{2K})I$ and hence

$$\|(I-\eta A_i)^{k-1}\zeta_{i,k}\|_2^2 \le \sigma^2(1+\tfrac{1}{2K})^{2(k-1)} \le 3\sigma^2\,.$$

This proves our second statement of the lemma. For strongly convex functions, we have for $\eta \le \frac{1}{\beta}$,

$$\|(I-\eta A_i)^{k-1}\zeta_{i,k}\|_2^2 \le \sigma^2(1-\eta\mu)^{2(k-1)} \le \sigma^2(1-\eta\mu)^{k-1}\,.$$

□

F.3. Lemmas showing progress

Progress of one client in one step. Now we focus only on a single client and monitor their progress.

Lemma 24. *Suppose (A2), (A5) and (A4) hold, and $\{f_i\}$ are quadratics. Then, the following holds for the update (28) with $\eta \le \min(\frac{1}{10\beta}, \frac{1}{22\delta K}, \frac{1}{\mu K})$ with $\mu = 0$ is f is non-convex or general-convex*

$$\begin{aligned}\xi_{i,k}^r &\le (1 - \tfrac{\mu\eta}{6})\xi_{i,k-1}^r - \tfrac{\eta}{6}\,\mathbb{E}_{r-1}\|\nabla f(\boldsymbol{y}_{i,k-1}^r)\|^2 + 7\beta\eta^2\sigma^2, \textit{ and}\\ \tilde{\xi}_{i,k}^r &\le (1 - \tfrac{\mu\eta}{6})\tilde{\xi}_{i,k-1}^r - \tfrac{\eta}{6}\|\nabla f(\mathbb{E}_{r-1}[\boldsymbol{y}_{i,k-1}^r])\|^2\,.\end{aligned}$$

Proof. Recall that $\xi_{i,k}^r \ge 0$ is defined to be

$$\xi_{i,k}^r := \big(\mathbb{E}_{r-1}[f(\boldsymbol{y}_{i,k}^r)] - f(\boldsymbol{x}^\star) + \delta(1+\tfrac{1}{K})^{K-k}\,\mathbb{E}_{r-1}\|\boldsymbol{y}_{i,k}^r - \boldsymbol{x}^{r-1}\|^2\big)\,.$$

Let us start from the local update step (28) (dropping unnecessary subscripts and superscripts)

$$\begin{aligned}\mathbb{E}_{r-1,k-1}\|\boldsymbol{y}_i^+ - \boldsymbol{x}^\star\|_A^2 &\le \|\boldsymbol{y}_i - \boldsymbol{x}^\star\|_A^2 - 2\eta\langle A(\boldsymbol{y}_i - \boldsymbol{x}^\star), A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\rangle + 2\eta\langle (A - A_i)(\boldsymbol{y}_i - \boldsymbol{x}), A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\rangle\\ &\quad + \eta^2\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star) + (A_i - A)(\boldsymbol{y}_i - \boldsymbol{x}))\|_A^2 + \beta\eta^2\sigma^2\\ &\le \|\boldsymbol{y}_i - \boldsymbol{x}^\star\|_A^2 - \tfrac{3\eta}{2}\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|_2^2 + 2\eta\|(A - A_i)(\boldsymbol{y}_i - \boldsymbol{x})\|_2^2\\ &\quad + 2\eta^2\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|_A^2 + 2\eta^2\|(A_i - A)(\boldsymbol{y}_i - \boldsymbol{x}))\|_A^2 + \beta\eta^2\sigma^2\\ &\le \|\boldsymbol{y}_i - \boldsymbol{x}^\star\|_A^2 - (\tfrac{3\eta}{2} - 2\eta^2\beta)\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|_2^2 + \beta\eta^2\sigma^2 + \delta^2(2\eta^2\beta + 2\eta)\|\boldsymbol{y}_i - \boldsymbol{x}\|_2^2\\ &\le \|\boldsymbol{y}_i - \boldsymbol{x}^\star\|_A^2 - (\tfrac{3\eta}{2} - 2\eta^2\beta)\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|_2^2 + \beta\eta^2\sigma^2 + \tfrac{\delta}{10K}\|\boldsymbol{y}_i - \boldsymbol{x}\|_2^2\,.\end{aligned}$$

The second to last inequality used that $\|\cdot\|_A^2 \le \beta\|\cdot\|_2^2$ by (A5) and that $\|(A - A_i)(\cdot)\|_2^2 \le \delta^2\|\cdot\|_2^2$ by (A2). The final inequality used that $\eta \le \max(\frac{1}{10\beta}, \frac{1}{22\delta K})$. Now, multiplying Lemma 22 by $\delta(1+\frac{1}{K})^{K-k} \le \frac{20\delta}{7}$ we have

$$\begin{aligned}\delta(1+\tfrac{1}{K})^{K-k}\,\mathbb{E}_{r-1,k-1}\|\boldsymbol{y}_i^+ - \boldsymbol{x}\|^2 &\le \delta(1+\tfrac{1}{K})^{K-k}(1+\tfrac{1}{2K})\|\boldsymbol{y}_i - \boldsymbol{x}\|^2 + 20\delta K\eta^2\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|^2 + 3\delta\eta^2\sigma^2\\ &\le \delta(1+\tfrac{1}{K})^{K-k}(1+\tfrac{1}{2K}+\tfrac{1}{10K})\|\boldsymbol{y}_i - \boldsymbol{x}\|^2 - \tfrac{\delta}{10K}\|\boldsymbol{y}_i - \boldsymbol{x}\|^2\\ &\quad + 20\delta K\eta^2\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|^2 + 3\delta\eta^2\sigma^2\\ &\le (1 - \tfrac{1}{5K})\delta(1+\tfrac{1}{K})^{K-k+1}(1+\tfrac{1}{K})\|\boldsymbol{y}_i - \boldsymbol{x}\|^2 - \tfrac{\delta}{10K}\|\boldsymbol{y}_i - \boldsymbol{x}\|^2\\ &\quad + 20\delta K\eta^2\|A(\boldsymbol{y}_i - \boldsymbol{x}^\star)\|^2 + 3\delta\eta^2\sigma^2\,.\end{aligned}$$

Karimireddy, Sai Praneeth, et al. "Scaffold: Stochastic controlled averaging for federated learning." International conference on machine learning. PMLR, 2020.

Li, Xiaoyun, and Ping Li. "Analysis of error feedback in federated non-convex optimization with biased compression: Fast convergence and partial participation." International Conference on Machine Learning. PMLR, 2023.

FL frameworks

- Flower (https://github.com/adap/flower)
- PySyft (https://github.com/OpenMined/PySyft)
- TFF (https://github.com/google-parfait/tensorflow-federated)
- But you can also implement from the scratch.
  - https://github.com/thejungwon/GC-Fed

Summary

- **Federated Learning (FL)** preserves data privacy by keeping the raw data on each client device.
  Instead of transmitting the data, clients share their **locally trained model parameters** with a central server.
- **Key Challenge:**
  Performance often degrades in **non-IID settings** (when client data distributions differ).
- **Reason:**
  Each client's training tends to move the model toward its own **local optimum (client drift)**, causing misalignment when aggregating updates.

Challenges other than optimization

- Incentive Mechanism
  - How to measure contribution
  - How to distribute reward
- Decentralized Federated Learning
  - No central authority (server)
    - How do they agree on global model?
    - How can they measure the performance?
- Communication Efficiency
  - Quantization (Compression)
  - Asynchronous Federated Learning

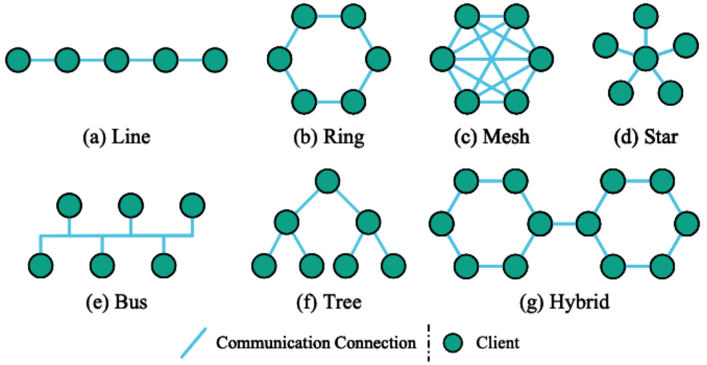
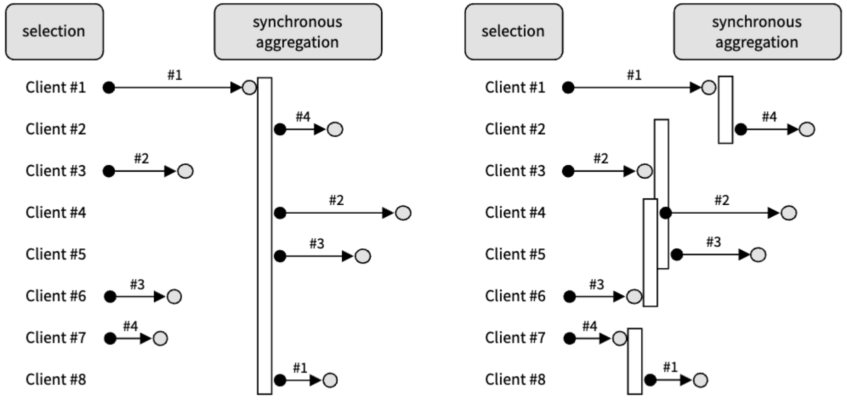


Fig. 4. Illustration of communication network topology.



(a) **synchronous** federated learning

(b) **asynchronous** federated learning

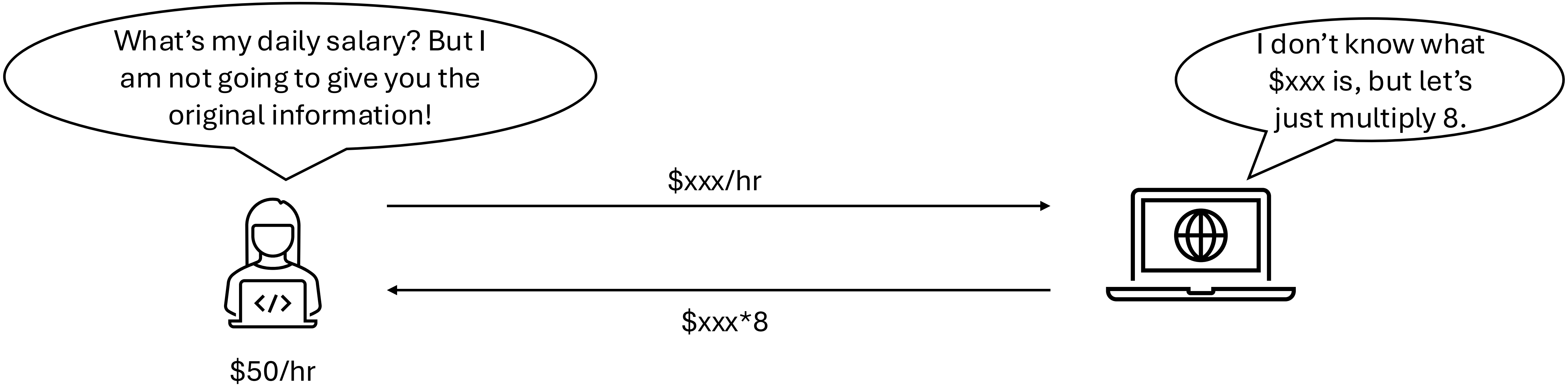
Beltrán, Enrique Tomás Martínez, et al. "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges." *IEEE Communications Surveys & Tutorials* (2023).

Su, Ningxin, and Baochun Li. "How asynchronous can federated learning be?." 2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS). IEEE, 2022.

Secure Multi-Party Computation with Secret Sharing

Motivation

- Without disclosing my input values, can a third party perform the computation on my behalf?
  - Can I ask a question to ChatGPT without revealing the both question and answer?



What's my daily salary? But I am not going to give you the original information!

I don't know what $xxx is, but let's just multiply 8.

$xxx/hr

$xxx*8

$50/hr

Basic concept

- Secure Multi-party Computation (SMPC)
  - A cryptographic technique that allows multiple parties to **jointly compute** a function over their inputs while keeping **those inputs private**.

Techniques:

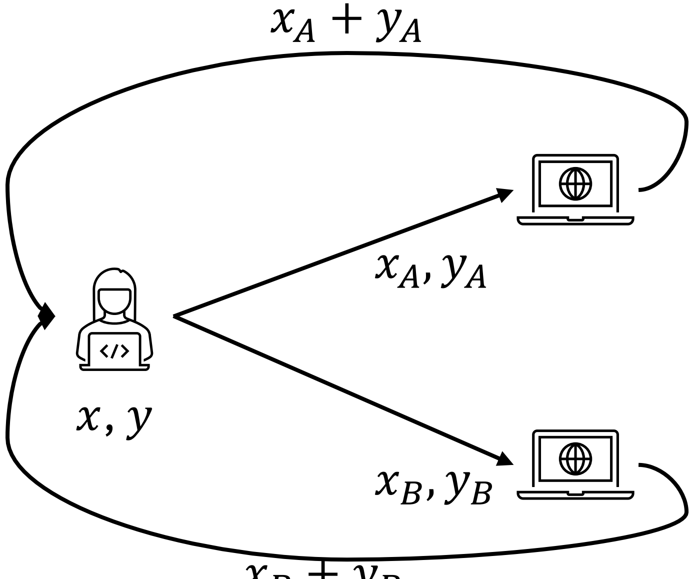
- Homomorphic Encryption (HE)
- Garbled Circuits (GC)
- Oblivious Transfer (OT)
- Trusted Execution Environment (TEE)
- **Secret Sharing (SS)**

Key Feature

- To support SMPC, we need to design operations that can be performed on encrypted (or similarly protected) data instead of on plain data.
- For example,
  - A+B = Dec(Enc(A) ⊕ Enc (B))
  - ⊕: can’t be just +
  - 3 + 5 ≠ Dec(0xabc331.. + 0xac08431..)

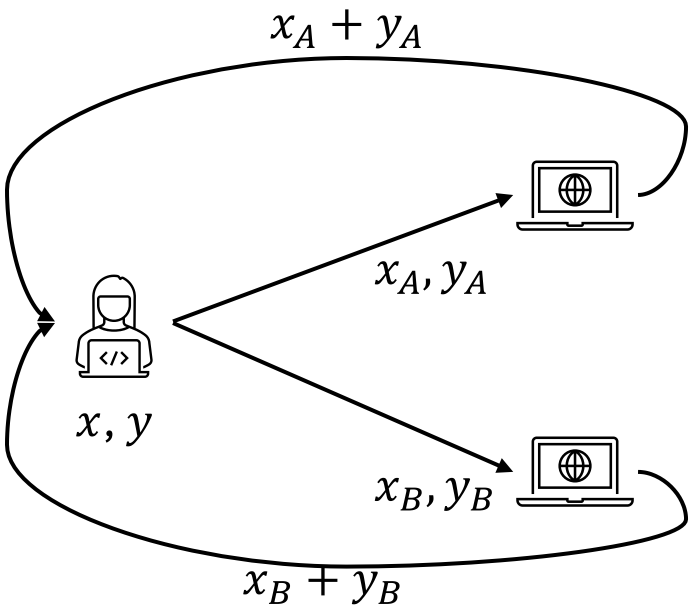
Secret Sharing

- In Secret Sharing, share doesn’t mean ‘to share openly,’ but rather refers to a ‘share’ as in a **portion or stake of the secret**.
- The main idea is to distribute **pieces of the original data** (secret shares) to multiple computing parties, then collect the results of their computations to acquire the final result.



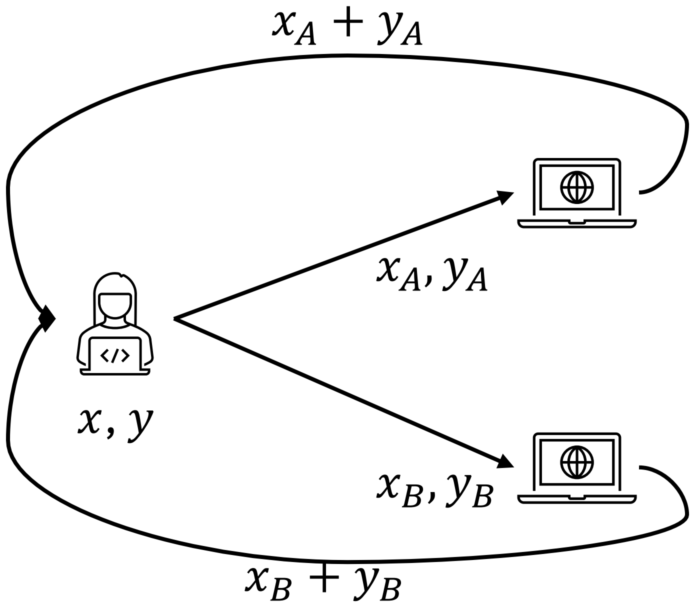
Addition (2-party) using Additive Secret Sharing

- Goal: Alice wants to perform $\boldsymbol{x} + \boldsymbol{y}$ with 3rd-party computers (A and B).
- Alice holds $\boldsymbol{x} = \mathbf{5}$ and $\boldsymbol{y} = \mathbf{6}$.
    - Generate secret shares for x
        - $\boldsymbol{x_A} = 3$ (random pick)
        - $\boldsymbol{x_B} = x - \boldsymbol{x_A} = 5 - 3 = 2$
    - Generate secret shares for y
        - $\boldsymbol{y_A} = 4$ (random pick)
        - $\boldsymbol{y_B} = y - \boldsymbol{y_A} = 6 - 4 = 2$
- Server A calculates : $\boldsymbol{x_A} + \boldsymbol{y_A} = \mathbf{3} + \mathbf{4} = \mathbf{7}$
- Server B calculates : $\boldsymbol{x_B} + \boldsymbol{y_B} = \mathbf{2} + \mathbf{2} = \mathbf{4}$
- Server A and B return **7** and **4** to Alice
- Alice can calculate **7+4 = 11**



Addition (2-party) using Additive Secret Sharing

- Neither *Server A* nor *Server B* can know the original X and Y unless they share the information with each other.



- How can we make more secure?

Addition (3-party)

- Goal: Alice wants to perform $x + y$ with 3rd-party computers (A, B and C).
- Alice holds $x = 5$ and $y = 6$.
  - Secret shares for x
    - $\boldsymbol{x_A} = 3$ (random pick)
    - $\boldsymbol{x}_{\mathrm{B}} = 1$ (random pick)
    - $\boldsymbol{x}_{\mathrm{C}} = x - \boldsymbol{x_A} - \boldsymbol{x}_{\mathrm{B}} = 5 - 3 - 1 = 1$
  - Secret shares for y
    - $\boldsymbol{y_A} = 4$ (random pick)
    - $\boldsymbol{y_B} = 5$ (random pick)
    - $\boldsymbol{y}_{\mathrm{C}} = y - \boldsymbol{y_A} - \boldsymbol{y}_{\mathrm{B}} = 6 - 4 - 5 = -3$
- Server A calculates : $\boldsymbol{x_A + y_A = 3 + 4 = 7}$
- Server B calculates : $\boldsymbol{x_B + y_B = 1 + 5 = 6}$
- Server C calculates : $\boldsymbol{x_C + y_C = 1 + (-3) = -2}$
- Server A , B, C return **7, 6, $\boldsymbol{-2}$** to Alice
- Alice can calculate **7+ 6 - 2 = 11**

Why Additive Secret Sharing Works

- For Addition and Subtraction
  - $[x] = (\mathrm{s}_1, \mathrm{s}_2, \dots, \mathrm{s}_\mathrm{n})$
  - $[y] = (\mathrm{t}_1, \mathrm{t}_2, \dots, \mathrm{t}_\mathrm{n})$
  - $x + y = (\sum_\mathrm{i} \mathrm{s}_\mathrm{i} + \sum_\mathrm{i} \mathrm{t}_\mathrm{i}) = \sum_\mathrm{i}(\mathrm{s}_\mathrm{i} + \mathrm{t}_\mathrm{i})$

- How about multiplication?
  - $[x] = (\mathrm{s}_1, \mathrm{s}_2, \dots, \mathrm{s}_\mathrm{n})$
  - $[y] = (\mathrm{t}_1, \mathrm{t}_2, \dots, \mathrm{t}_\mathrm{n})$
  - $(\sum_\mathrm{i} \mathrm{s}_\mathrm{i} \cdot \sum_\mathrm{i} \mathrm{t}_\mathrm{i}) \neq \sum_\mathrm{i}(\mathrm{s}_\mathrm{i} \cdot \mathrm{t}_\mathrm{i})$

Multiplication

- Goal: Alice wants to perform $x \times y$ with 3rd-party computers.
- Alice holds $x = 5$ and $y = 6$.
  - Secret shares for x
    - $\boldsymbol{x_A} = 3)$ random pick)
    - $\boldsymbol{x_B} = x - \boldsymbol{x_A} = 5 - 3 = 2$
  - Secret shares for y
    - $\boldsymbol{y_A} = 4)$ random pick)
    - $\boldsymbol{y_B} = y - \boldsymbol{y_A} = 6 - 4 = 2$

- Challenge: We cannot simply compute like addition.
  - $[x] = (\mathrm{s}_1, \mathrm{s}_2, \dots, \mathrm{s}_\mathrm{n})$
  - $[y] = (\mathrm{t}_1, \mathrm{t}_2, \dots, \mathrm{t}_\mathrm{n})$
  - $(\sum_\mathrm{i} \mathrm{s}_\mathrm{i} \cdot \sum_\mathrm{i} \mathrm{t}_\mathrm{i}) \neq \sum_\mathrm{i}(\mathrm{s}_\mathrm{i} \cdot \mathrm{t}_\mathrm{i})$

	A	B
x	3	2
y	4	2
a		
b		
c		
e		
f		

Beaver Triplet

- Precomputed and shared
  - By trusted dealer or with other technique (HE, OT)
  - Input independent
- Beaver Triplet : a, b, c
  - a = 10 (random pick)
  - b = 15 (random pick)
  - c = a× b = 150
- Additive sharing again
  - $\mathrm{a}_{\boldsymbol{A}} = 3, \mathrm{a}_{\mathrm{B}} = 7$
  - $\mathrm{b}_{\boldsymbol{A}} = 5, \mathrm{b}_{\mathrm{B}} = 10$
  - $\mathrm{c}_{\boldsymbol{A}} = 50, \mathrm{c}_{\mathrm{B}} = 100$

	A	B
x	3	2
y	4	2
a	3	7
b	5	10
c	50	100
e		
f		

Beaver, Donald. "Efficient multiparty protocols using circuit randomization." Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991.

Multiplication

- Calculate (e, f)
  - $e = x - a$
    - $e_A = x_A - a_A = 3 - 3 = 0$
    - $e_B = x_B - a_B = 2 - 7 = -5$
  - $f = y - b$
    - $f_A = y_A - b_A = 4 - 5 = -1$
    - $f_B = y_B - b_B = 2 - 10 = -8$
- Open e and f (extra communication)
  - $e = e_A + e_B = -5$
  - $f = f_A + f_B = -9$

	A	B
x	3	2
y	4	2
a	3	7
b	5	10
c	50	100
e	-5	
f	-9	

Multiplication

- $x \times y = c + e \times b + f \times a + e \times f$
  - We can verify that substituting a, b, c, e, and f based on x and y respectively is equivalent.

- Server A
  - $\mathrm{z_A} = \mathrm{c_A} + e \times \mathrm{b_A} + f \times \mathrm{a_A} + e \times f$
  - $\mathrm{z_A} = 50 + (-5) \cdot 5 + (-9) \cdot 3 + (-5) \cdot (-9)$
  - $\mathrm{z_A} = 50-25-27+45=43$
- Server B
  - $\mathrm{z_B} = \mathrm{c_A} + e \times \mathrm{b_A} + f \times \mathrm{a_A}$
  - $\mathrm{z_B} = 100+(-5)\cdot 10+(-9)\cdot 7$
  - $\mathrm{z_B} = 100-50-63=-13$
- $\mathrm{z_A} + \mathrm{z_B} = 30$
  - $x = 5$ and $y = 6$

	A	B
x	3	2
y	4	2
a	3	7
b	5	10
c	50	100
e	-5	
f	-9	

Other Operations

Operation	Core Method	Example
Addition	Additive Sharing	X + Y
Subtraction	Additive Sharing	X - Y
Multiplication	Beaver Triplet	X * Y
Division	Reciprocal via Newton–Raphson (or Goldschmidt)	X / Y
Comparison (a>b)	Yao’s Garbled Circuits / Bit-Decomposition	X > Y, max(X, Y), min(X, Y)
Sigmoid	Polynomial / Taylor Approximation	$\sigma(x) \approx 0.5 + 0.25x - 0.0208x^3$ …
Vector, Matrix, Tensor	Parallelization of above operation (not one by one)	[1,2,3] + [4,5,6]
…	…	…

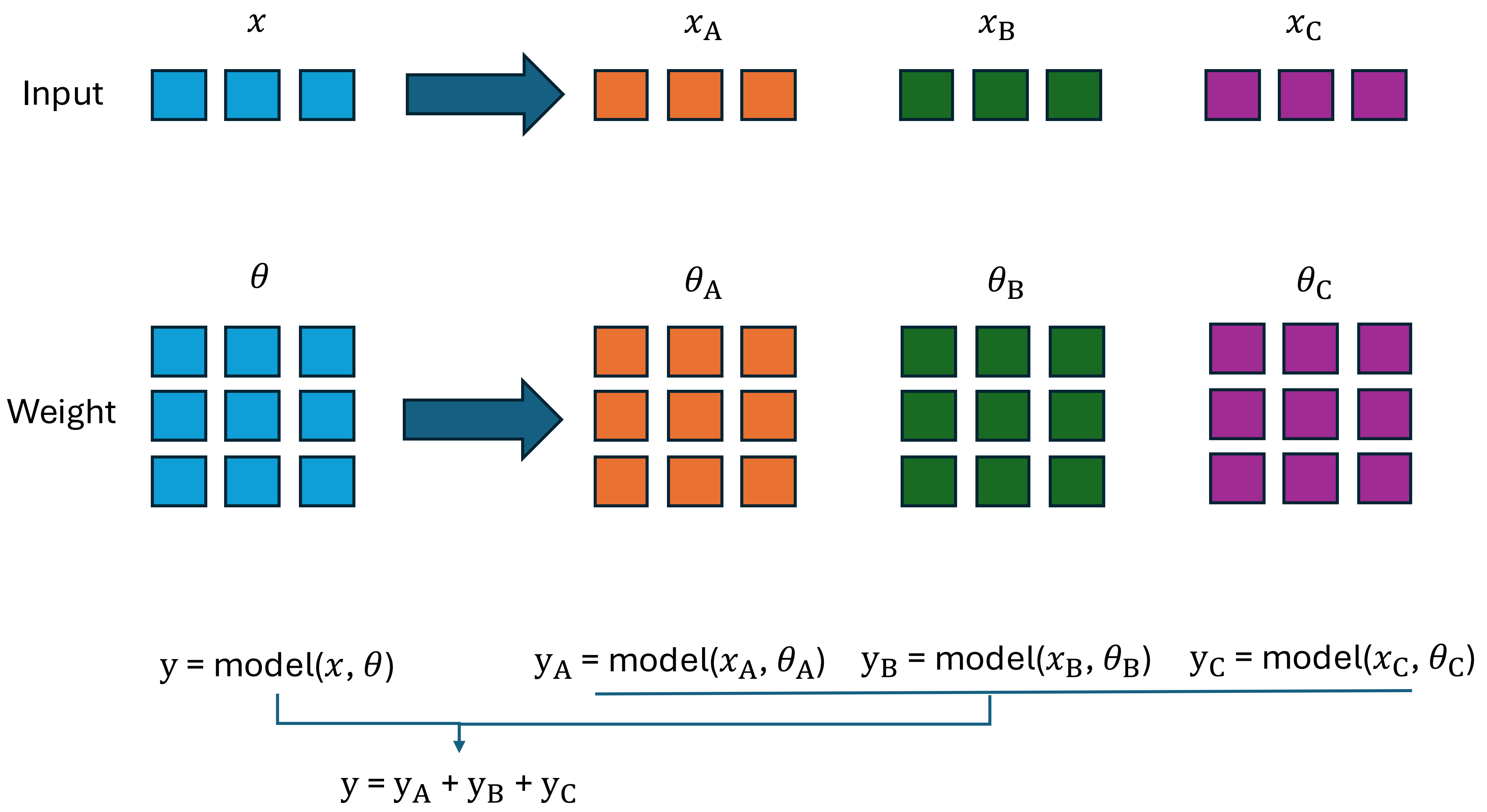
Inference with SMPC

- Using Secret Sharing, we can do pretty much all the operation.

- Even inference with neural network!

- Meaning of inference:
  - Data: x
  - Model parameter: θ
  - Inference: $\mathrm{y} = \mathrm{model}(\theta, \mathrm{x}) = \theta\mathrm{x} \ + \mathrm{b}$

Neural Network with Secret Sharing



x $\quad x_{\mathrm{A}}$ $\quad x_{\mathrm{B}}$ $\quad x_{\mathrm{C}}$

Input

θ $\quad \theta_{\mathrm{A}}$ $\quad \theta_{\mathrm{B}}$ $\quad \theta_{\mathrm{C}}$

Weight

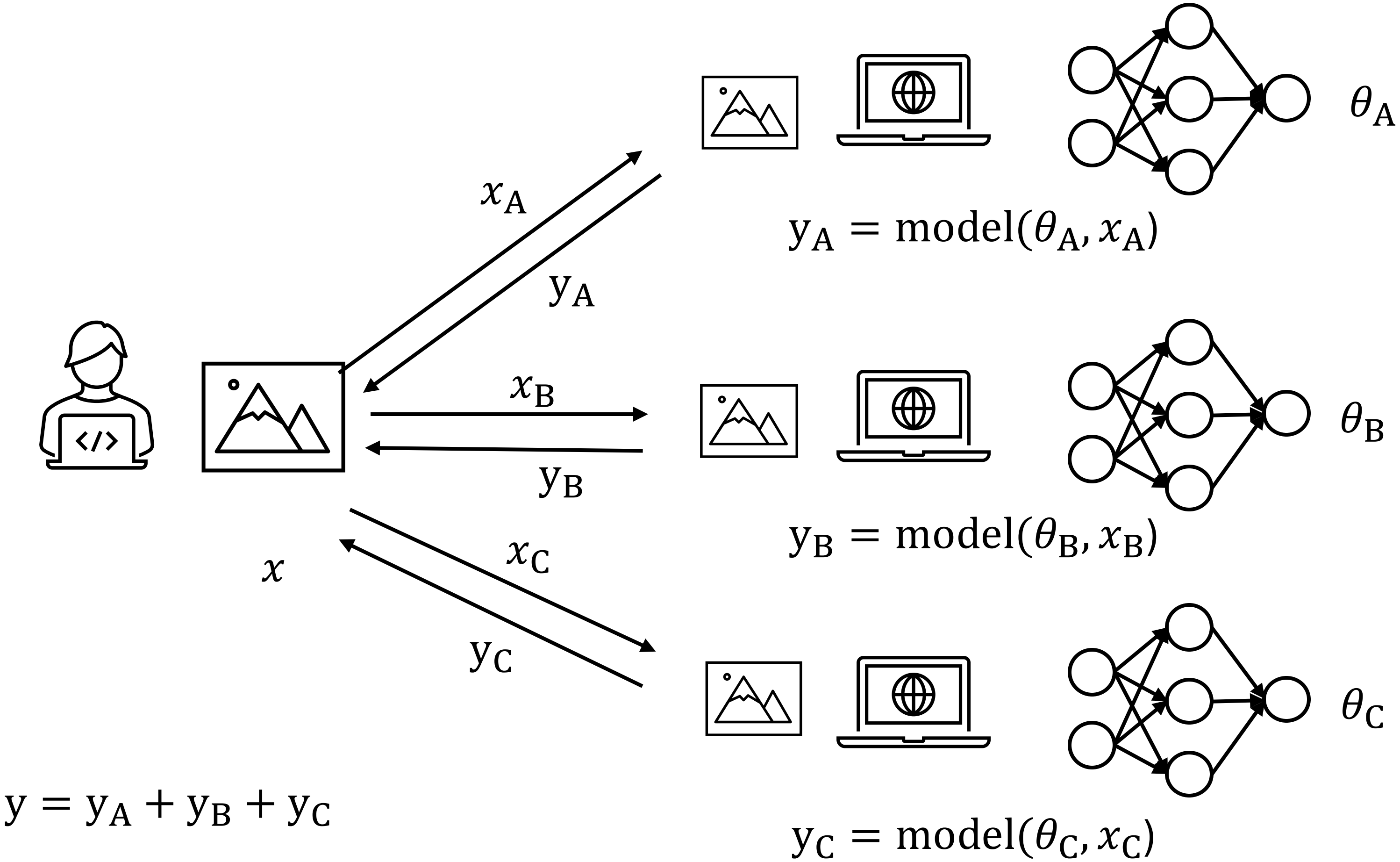
$\mathrm{y} = \mathrm{model}(x, \theta)$

$\mathrm{y}_{\mathrm{A}} = \mathrm{model}(x_{\mathrm{A}}, \theta_{\mathrm{A}})$ $\quad \mathrm{y}_{\mathrm{B}} = \mathrm{model}(x_{\mathrm{B}}, \theta_{\mathrm{B}})$ $\quad \mathrm{y}_{\mathrm{C}} = \mathrm{model}(x_{\mathrm{C}}, \theta_{\mathrm{C}})$

$\mathrm{y} = \mathrm{y}_{\mathrm{A}} + \mathrm{y}_{\mathrm{B}} + \mathrm{y}_{\mathrm{C}}$

Scenario 1 : Data Privacy

User can get the inference result without revealing the data.



$\mathrm{y}_{\mathrm{A}} = \mathrm{model}(\theta_{\mathrm{A}}, x_{\mathrm{A}})$

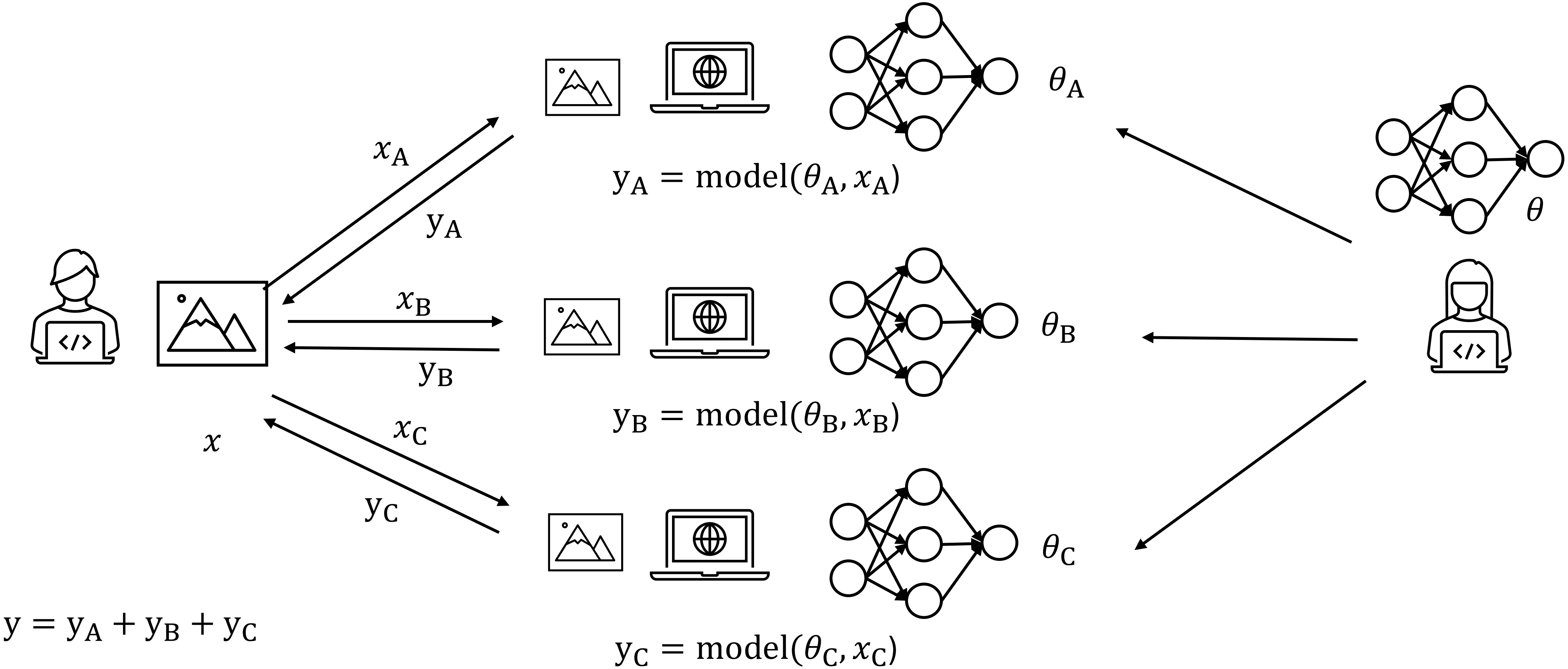
$\mathrm{y}_{\mathrm{B}} = \mathrm{model}(\theta_{\mathrm{B}}, x_{\mathrm{B}})$

$\mathrm{y}_{\mathrm{C}} = \mathrm{model}(\theta_{\mathrm{C}}, x_{\mathrm{C}})$

$\mathrm{y} = \mathrm{y}_{\mathrm{A}} + \mathrm{y}_{\mathrm{B}} + \mathrm{y}_{\mathrm{C}}$

Scenario 2: Data and Model Privacy

Data providers and model providers can collaborate without disclosing their data or models.


θ_A
x_A
$y_A = \text{model}(\theta_A, x_A)$
y_A
θ
x_B
θ_B
y_B
$y_B = \text{model}(\theta_B, x_B)$
x_C
x
y_C
θ_C
$y = y_A + y_B + y_C$
$y_C = \text{model}(\theta_C, x_C)$

SMPC Frameworks

- Crypten: https://github.com/facebookresearch/CrypTen
- SyMPC: https://github.com/OpenMined/SyMPC
- But you can also implement from the scratch…

Challenges

- Communication and computation overhead

Batch Size	SMPC w\ CPU (s)	SMPC w\ GPU (s)	Centralized w\ CPU (s)
1	7.70	23.0	1.20
2	9.79	23.0	1.27
8	27.72	28.0	1.32
16	52.99	33.0	1.29
64	228.06	69.0	1.33

  - Imagine training with SMPC…
- Relatively loose security assumption
  - Semi-honest (honest-but-curious)

Summary

- With **Federated Learning**, we can train models without directly sharing datasets.
  By applying **Secure Multi-Party Computation (SMPC)** with **Secret Sharing**, it is even possible to run model inference without exposing either the data or the model itself.
- That said, there are still some drawbacks compared to centralized approaches.
- However, these limitations can be mitigated through algorithmic improvements, and as hardware performance continues to advance, the trade-offs may eventually become negligible.

Thank you!