FEDERATED LEARNING

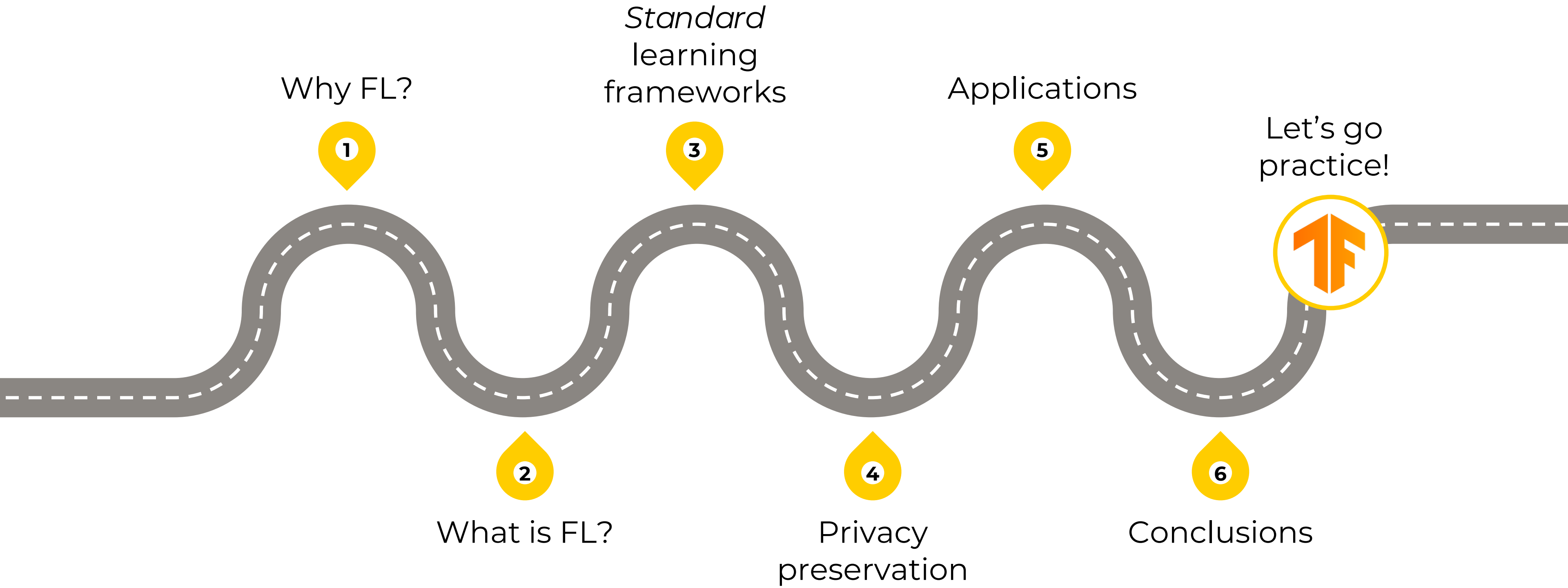
An Overview

Mirko **Polato**, Ph.D.

Assistant Professor @ Dept. of Computer Science, University of Turin

ogr Officine grandi riparazioni – **Opening Future**

Roadmap

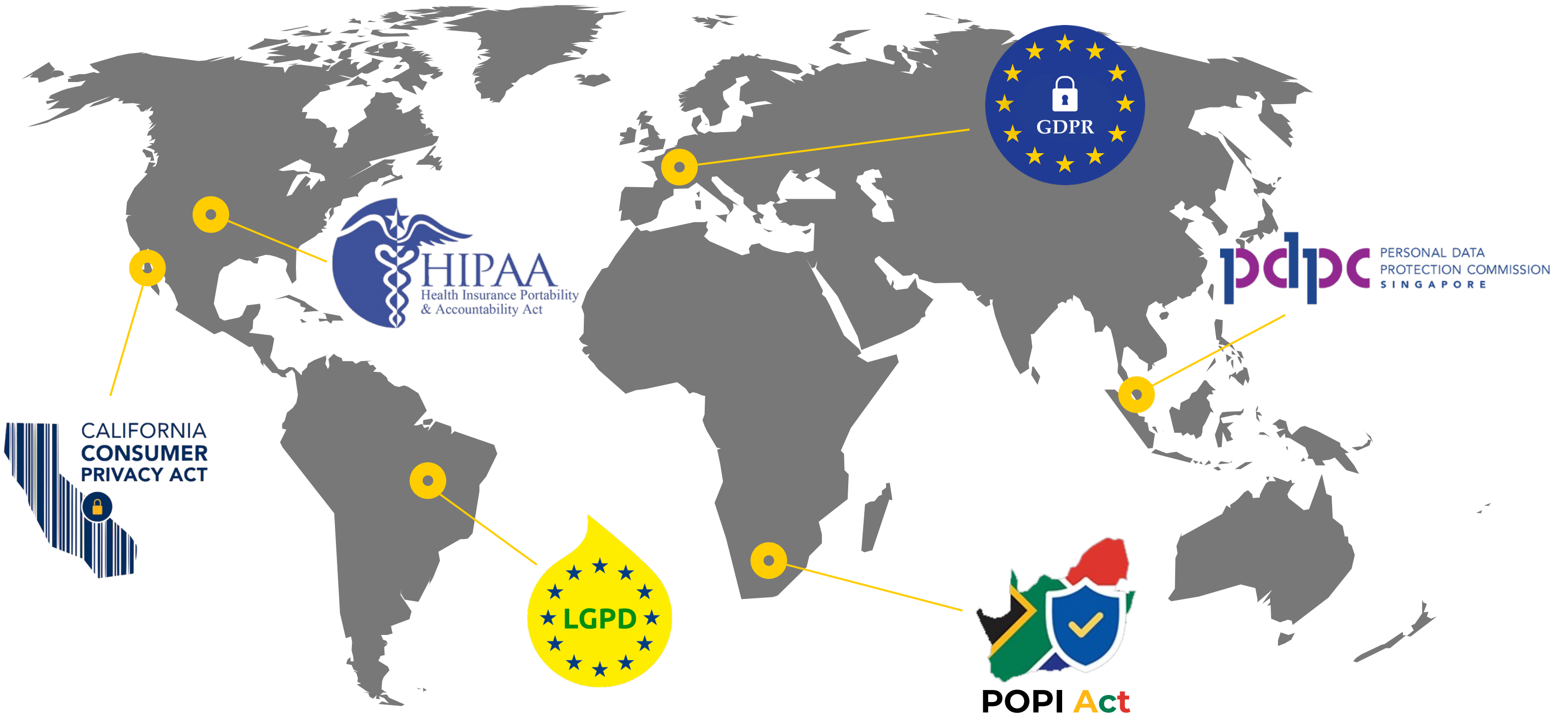

1 Why FL?
2 What is FL?
3 *Standard* learning frameworks
4 Privacy preservation
5 Applications
6 Conclusions
Let's go practice!

1

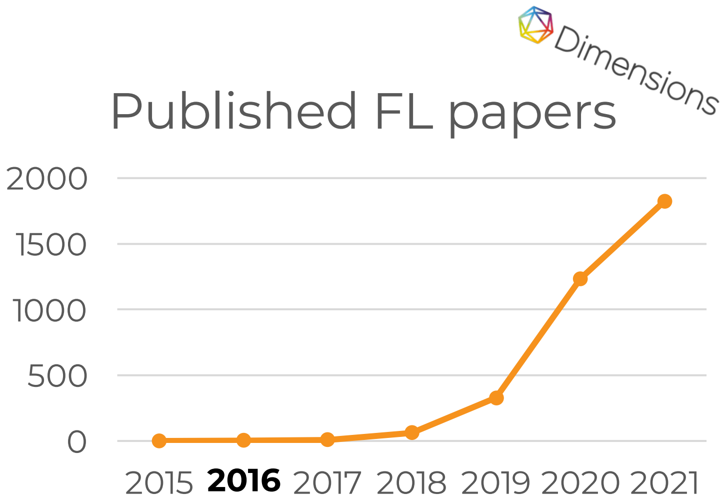
Why FL?

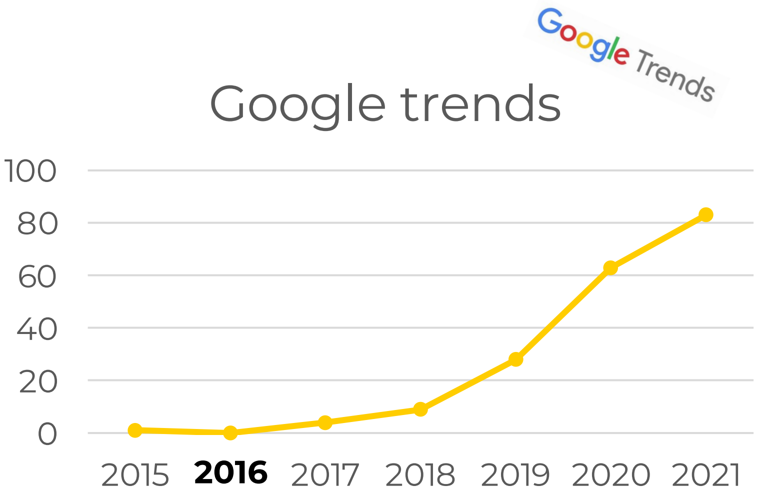
The main reasons behind this new learning paradigm

Data protection regulations



FL is on fire!





Google publishes the seminal work on FL

2

What is FL?

Definition and overview of the FL taxonomy

Informal definition

Federated Learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective.

"Formal" definition

	Parties	Data	Learning	Performance
Centralized	Server	$\mathcal{D}$	On the server	$\mathcal{P}$
Non-federated	Clients	$\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$	On clients	$\mathcal{P}_i \leq \mathcal{P}$
Federated	Clients Server	$\mathcal{D} = \mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$	Collaborative	$\mathcal{P}'$

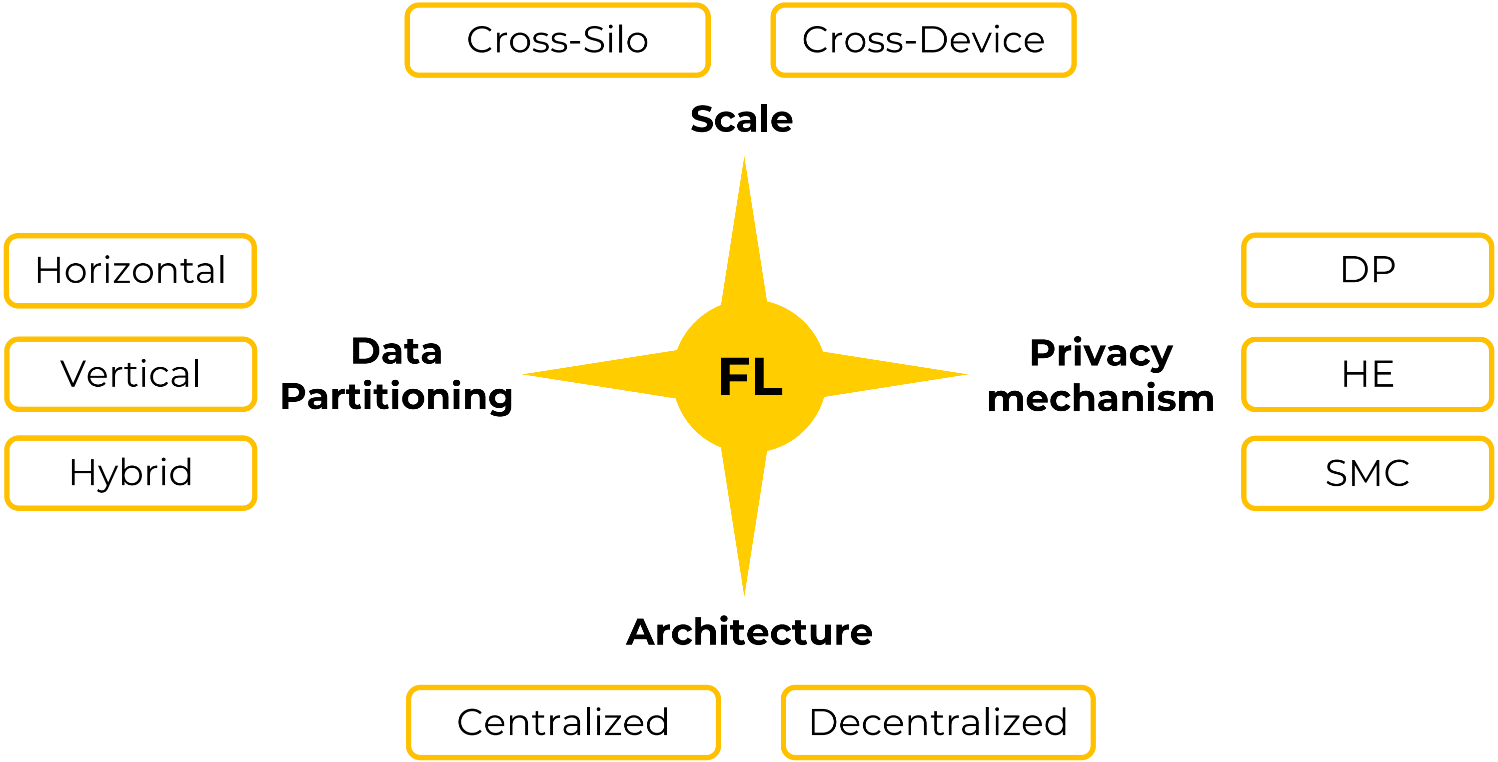
Goals

Clients do not share their data

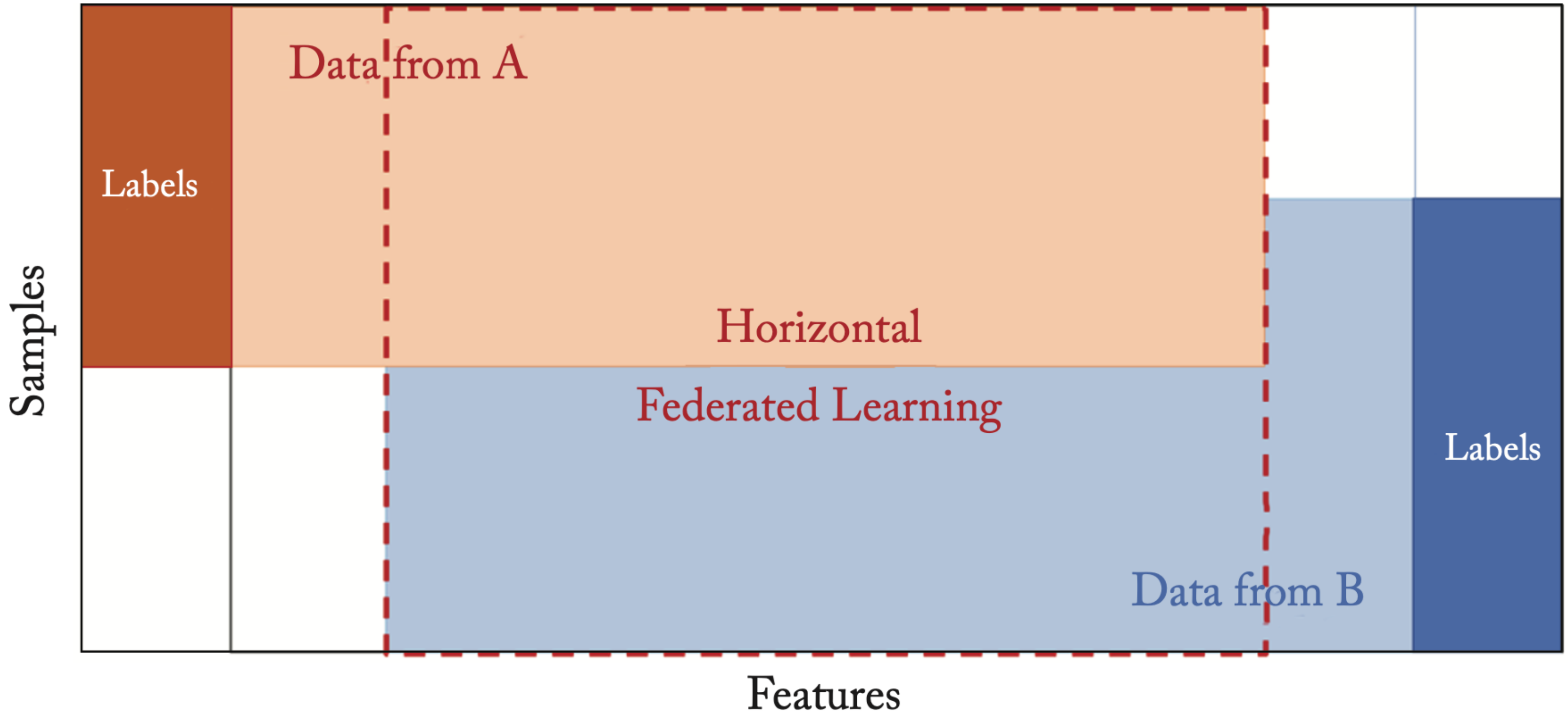
Clients benefit from the federation $\mathcal{P}_i \leq \mathcal{P}'$

The federated model is close to the "ideal" one $\mathcal{P} - \mathcal{P}' \to 0$

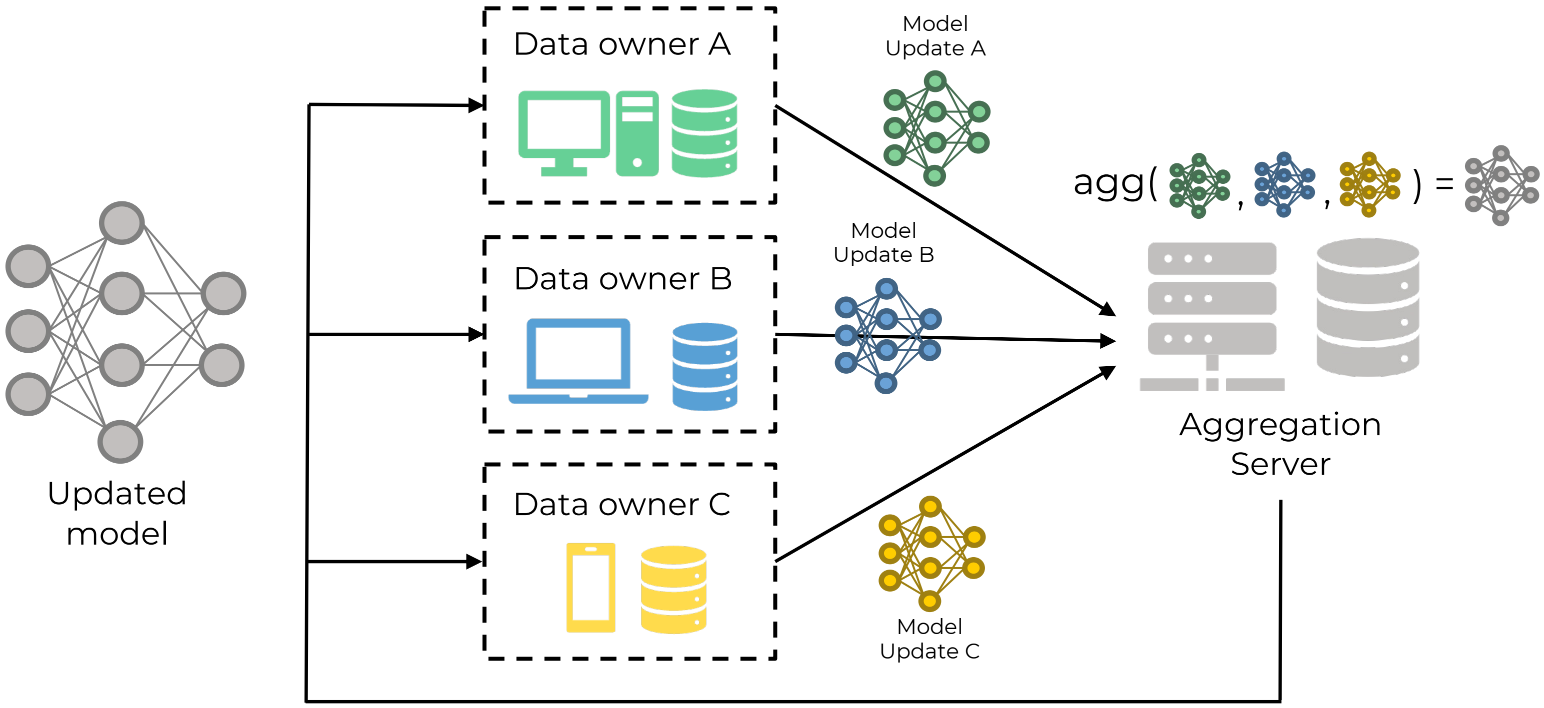
FL settings landscape



Horizontal FL



Standard HFL architecture

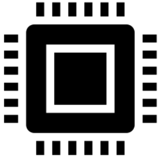


(Cross-device) FL characteristics

Large scale: thousands of devices

Unreliable connections

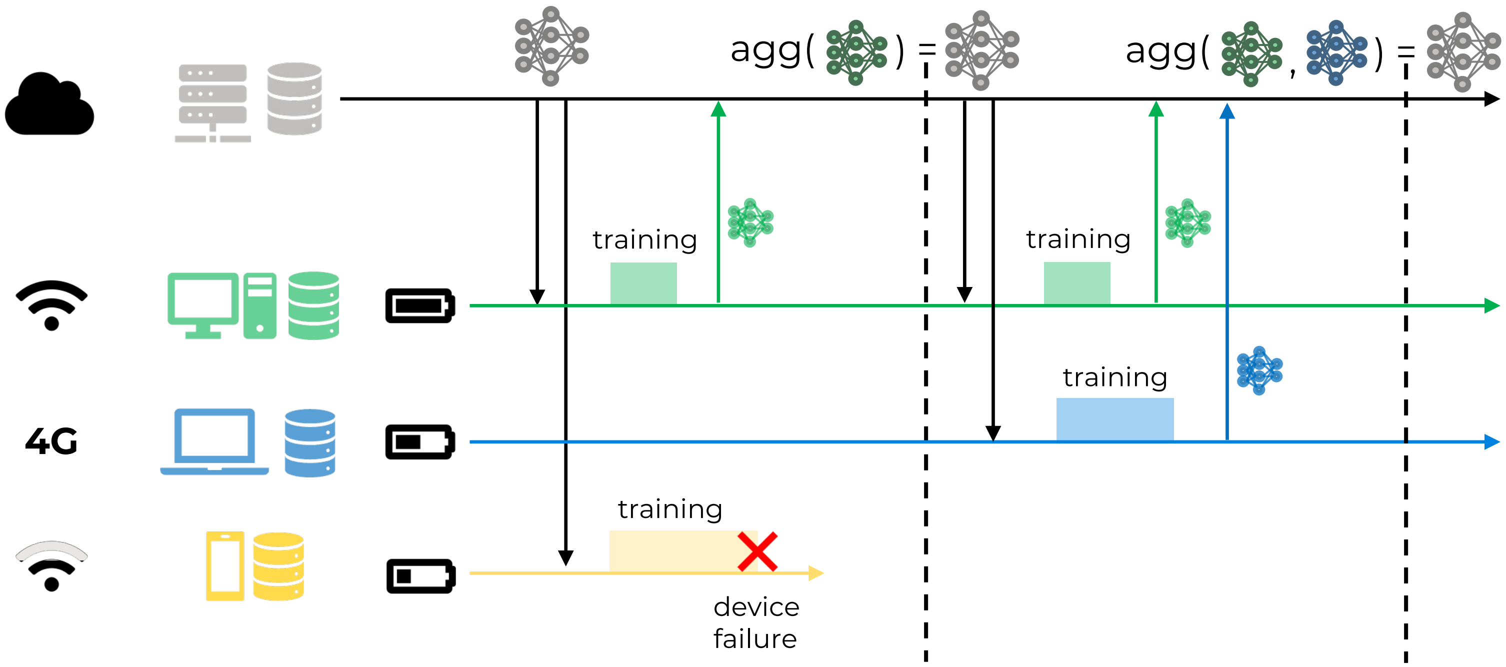
Limited computational capacity



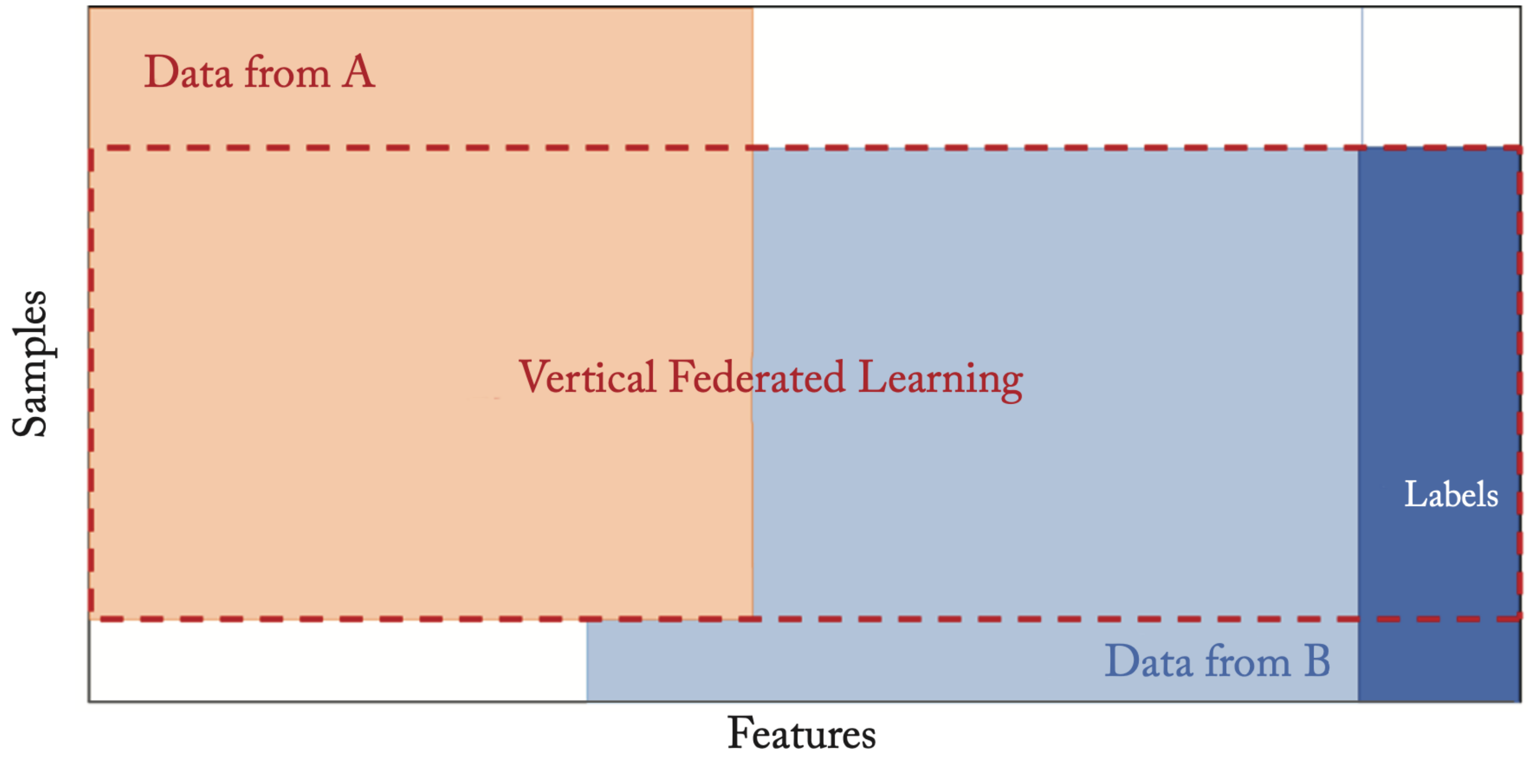
Power consumption

System heterogeneity

System heterogenity

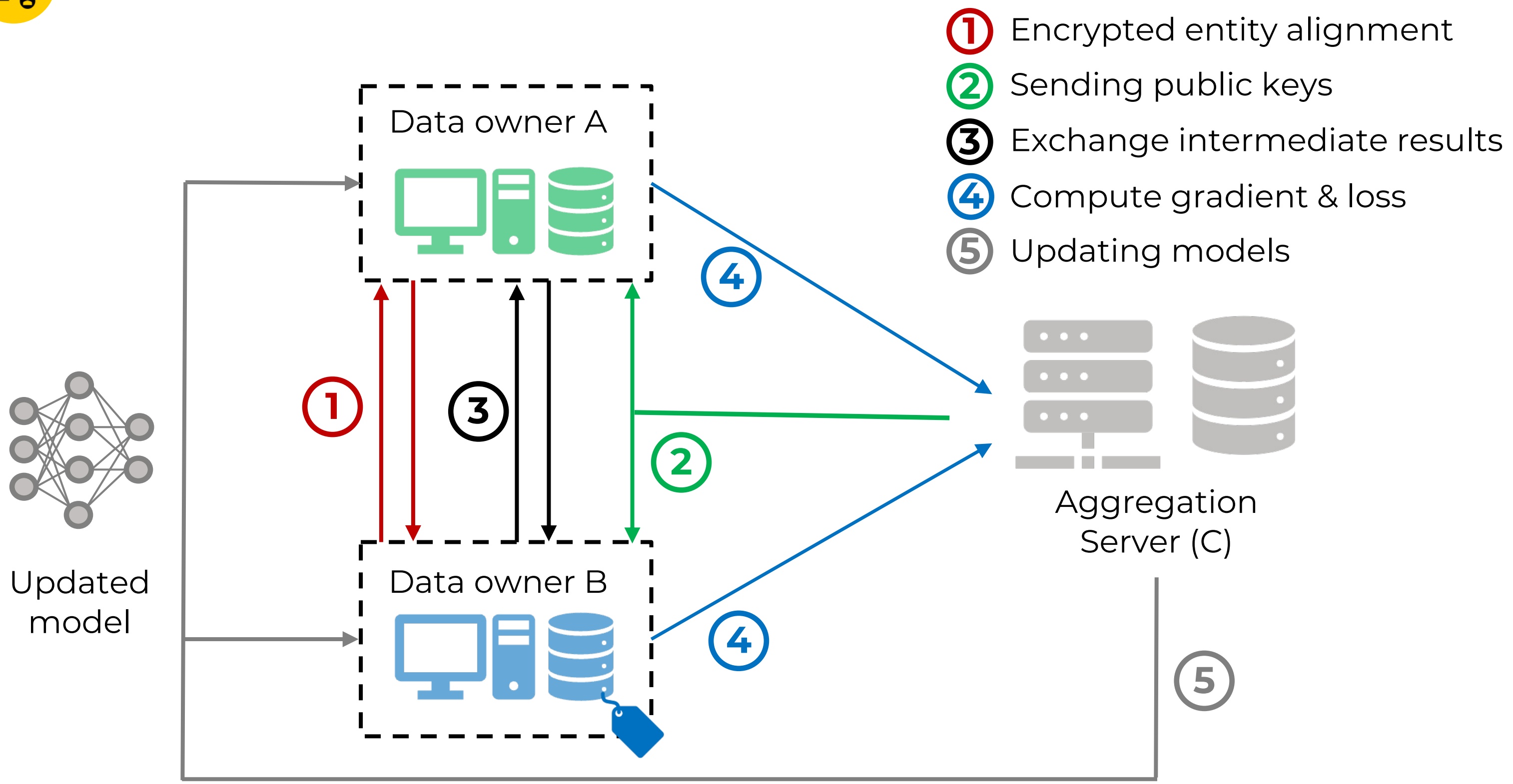


Vertical FL

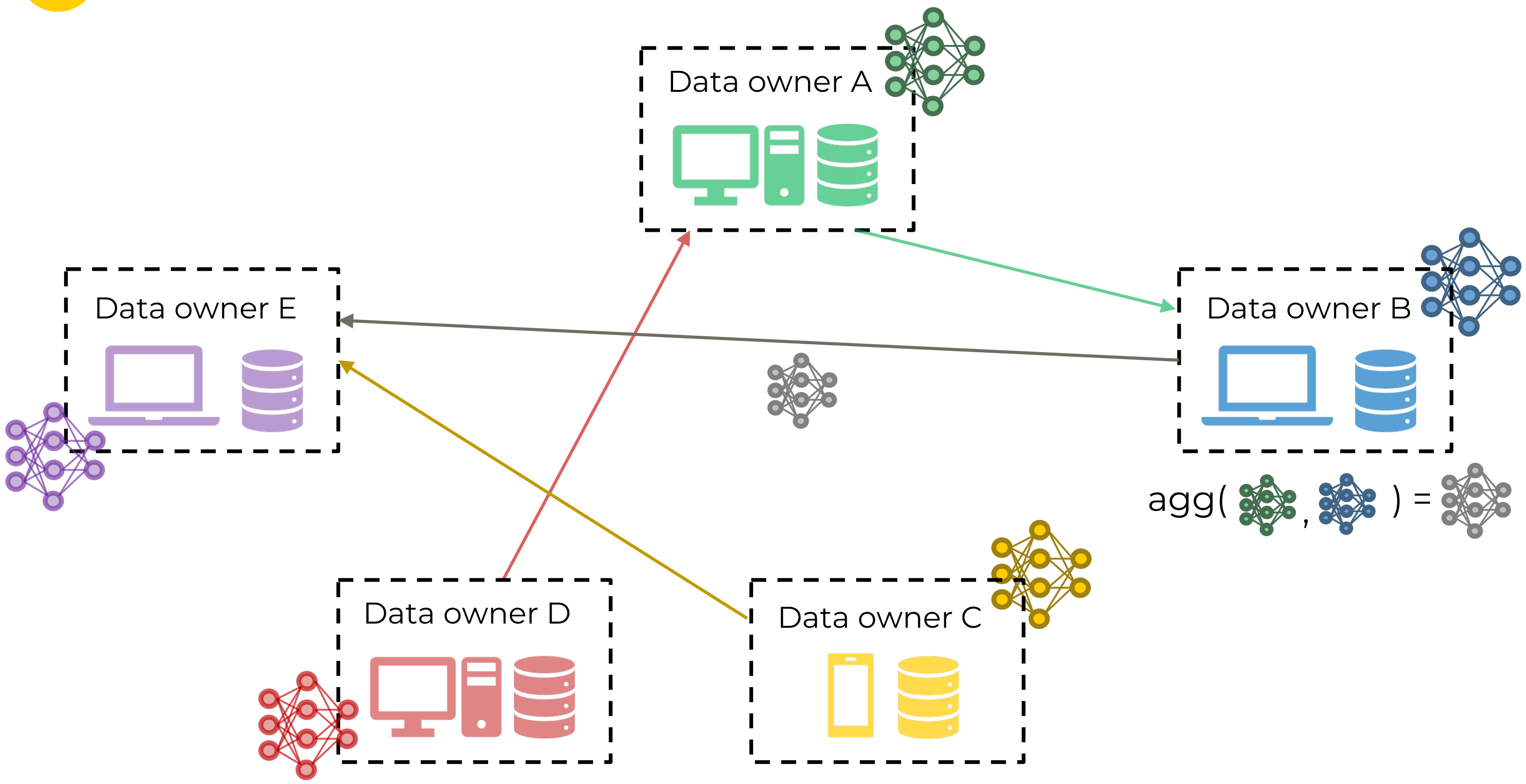


Data from A
Vertical Federated Learning
Labels
Data from B
Samples
Features

VFL Architecture



Decentralized FL



Centralized vs Decentralized

	Centralized	Decentralized
Orchestration	Server	No centralized orchestration*
Topology	Client-Server	Peer-to-peer
Global model	Single	Many
Setup	Centralized	Consensus*

* A central authority might be needed!

Cross-Silo FL

Limited number of collaborators

Big local datasets

Reliable connection/ participation

- **Few organizations** share incentives to train a model without sharing their data
- Or, same organization cannot centralized its data (e.g., legal constraints)

Cross-Silo: incentives

Free-rider problem

Organizations/compatitors may benefit from the federation without contributing as much

- Monetary payout
- Assign FL model with performance commensurate with the contributions (*game theory*)

Cross-Silo vs Cross-Device

	Cross-Silo	Cross-Device
Availability	~Always	Small fraction available
Scale	2-100 clients	Up to 10^{10} clients
Addressability	Direct	No client identifier
Reliability	Few failures	Highly unreliable
Dataset size	Big	Relatively small

Challenges

Hyper parameters tuning

Avoid overfitting

Model debugging

Malicious participants

Non-iidness

System heterogeneity

3

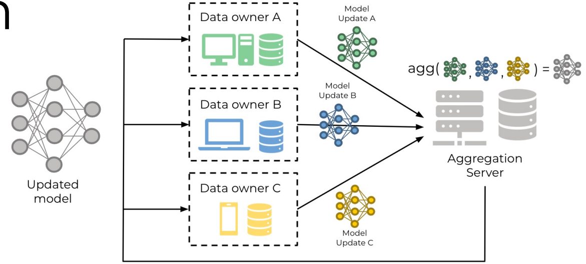
Standard **Learning Frameworks**

Learning in a federation

General Learning Scheme

1. Setup task – Model initialization

2. For each federated round:



   a. **Server**: Broadcast the current global model

   b. In parallel:
      clients update and send back the local models

   c. **Server**: Update global model with the ones received from the users

FedAvg (model averaging)

Aggregation Server

Algorithm: **FedAvg**

initialize model $\bar{w}_0$
for each round t=1,…:
  Broadcast $\bar{w}_{t-1}$
  select C eligible participants
  foreach|| participant p:
    $w_t^p \leftarrow$ LocalUpdate(p)
  $\bar{w}_t \leftarrow$ aggregate($\forall p\ w_t^p$)

Does not guarantee converge

Collaborator/Client

Algorithm: **LocalUpdate**

$w \leftarrow$ global model from Server
for each epoch $s \in 1,\dots,S$:
  for each batch b:
    $g_b \leftarrow$ compute gradient for b
    $w \leftarrow w - \eta g_b$
send w to the Server

Practically works most of the time

Efficient (communication-wise)

Not bound to SGD

FedSgd (gradient averaging)

Aggregation Server

Algorithm: **FedSgd**

```
initialize model w̄_0
for each round t=1,…:
   Broadcast w̄_{t−1}
   select C eligible participants
   foreach|| participant p:
      g_t^p ← LocalUpdate(p)
   g_t ← aggregate(∀p g_t^p)
   w̄_t ← w̄_{t−1} − η g_t
```

Inefficient (communication-wise)

Collaborator/Client

Algorithm: **LocalUpdate**

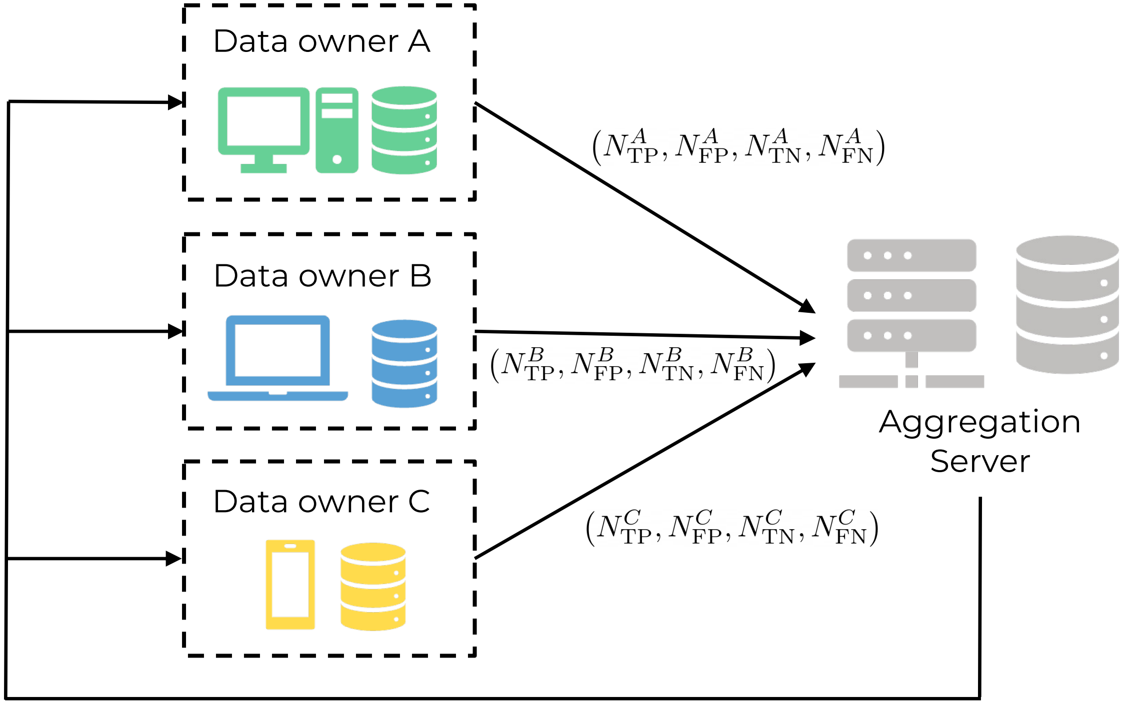
```
w ← global model from Server
select batch b
g_b ← compute gradient for b
send g_b to the Server
```

Guaranteed convergence

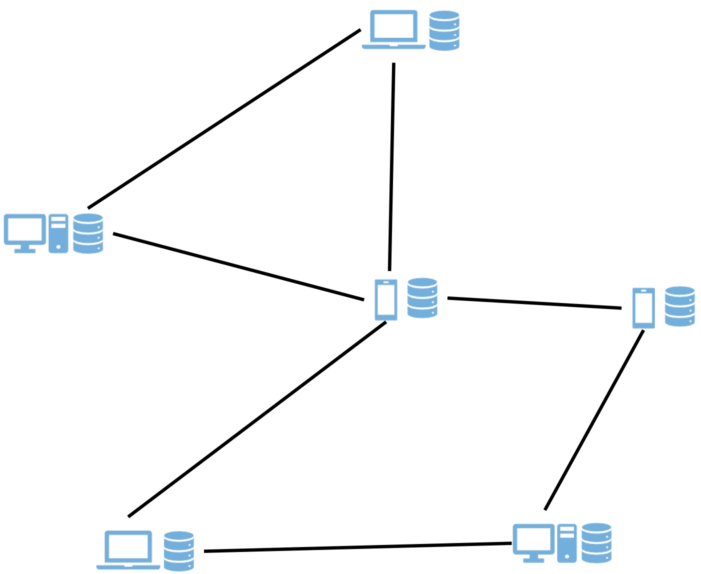
Model evaluation (e.g., classification)

$$\frac{\sum_{k\in\{A,B,C\}} N_{\mathrm{TP}}^{k}}{\sum_{k\in\{A,B,C\}} \left(N_{\mathrm{TP}}^{k} + N_{\mathrm{FN}}^{k}\right)}$$

e.g., global recall



Decentralized FL: Gossip Learning



Algorithm: **Main gossip loop (Push)**

1. initialize local model w
2. loop (forever)
3. wait for a fixed time Δ
4. select neighbor peer p
5. send w to p

Algorithm: **On receive model**

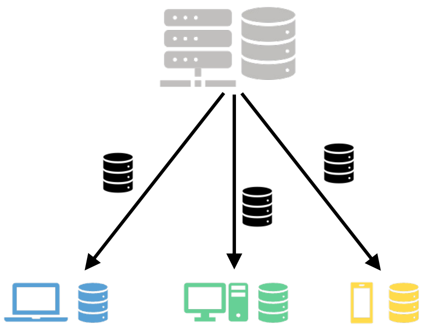
1. receive w_p from p
2. merge(w, w_p)
3. update(w)

e.g., gradient descent step on local data

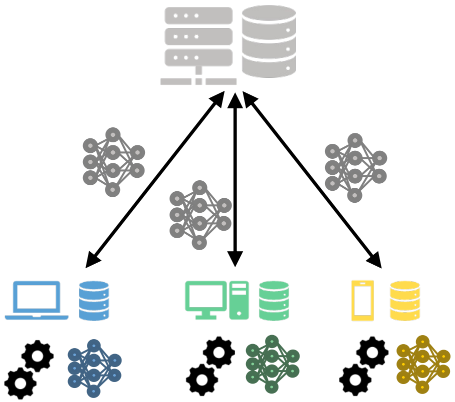
e.g., models averaging

Dealing with non-iidness

Data augmentation



Personalization: It's a feature not a bug!



- Ad hoc learning methods
- Hyper-parameter tuning

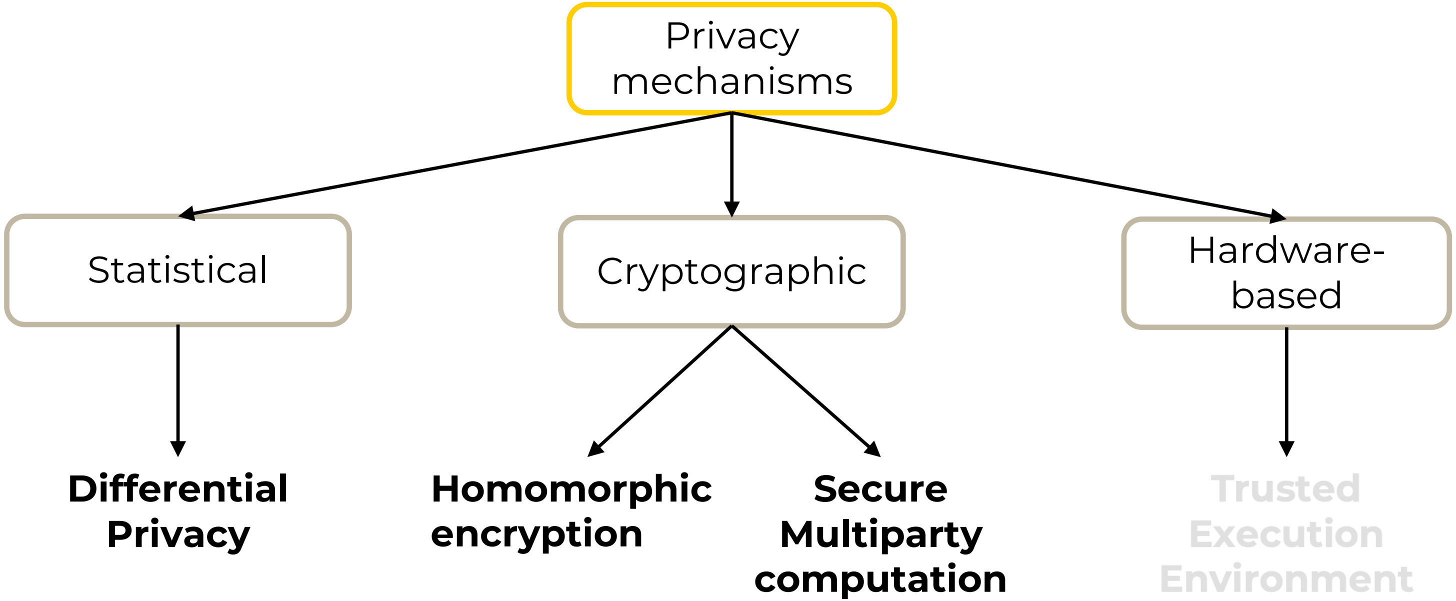
4

Privacy preservation

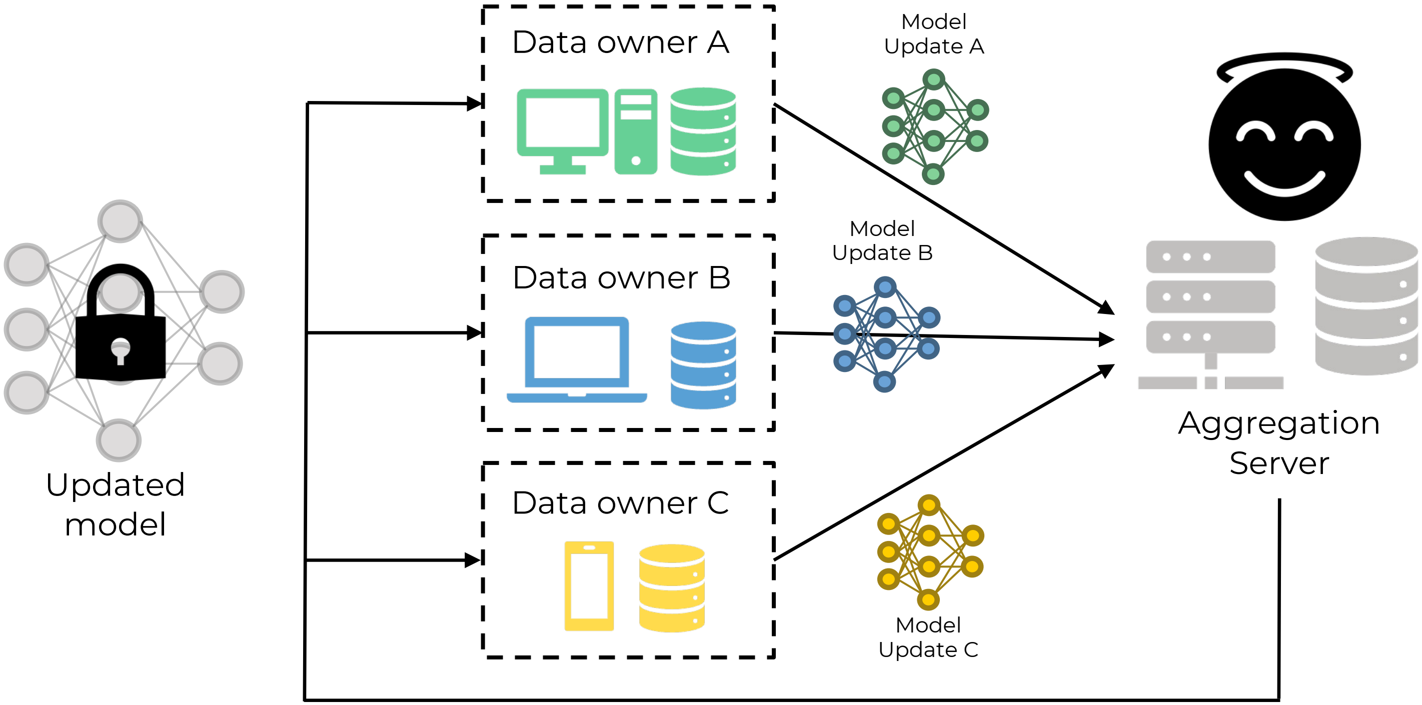
How to improve privacy in FL

FL may be not enough

Gradients/model updates may leak information about the user data!



Global Differential Privacy



FedAvg + Global DP

Aggregation Server

Algorithm: **FedAvg**

Initialize model $\bar{w}_0$
for each round t=1,…:
    Broadcast $\bar{w}_{t-1}$ + noise
    select C eligible participants
    foreach|| participant p:
        $w_t^p \leftarrow$ LocalUpdate(p)
    $\bar{w}_t \leftarrow$ aggregate($\forall p\ w_t^p$)

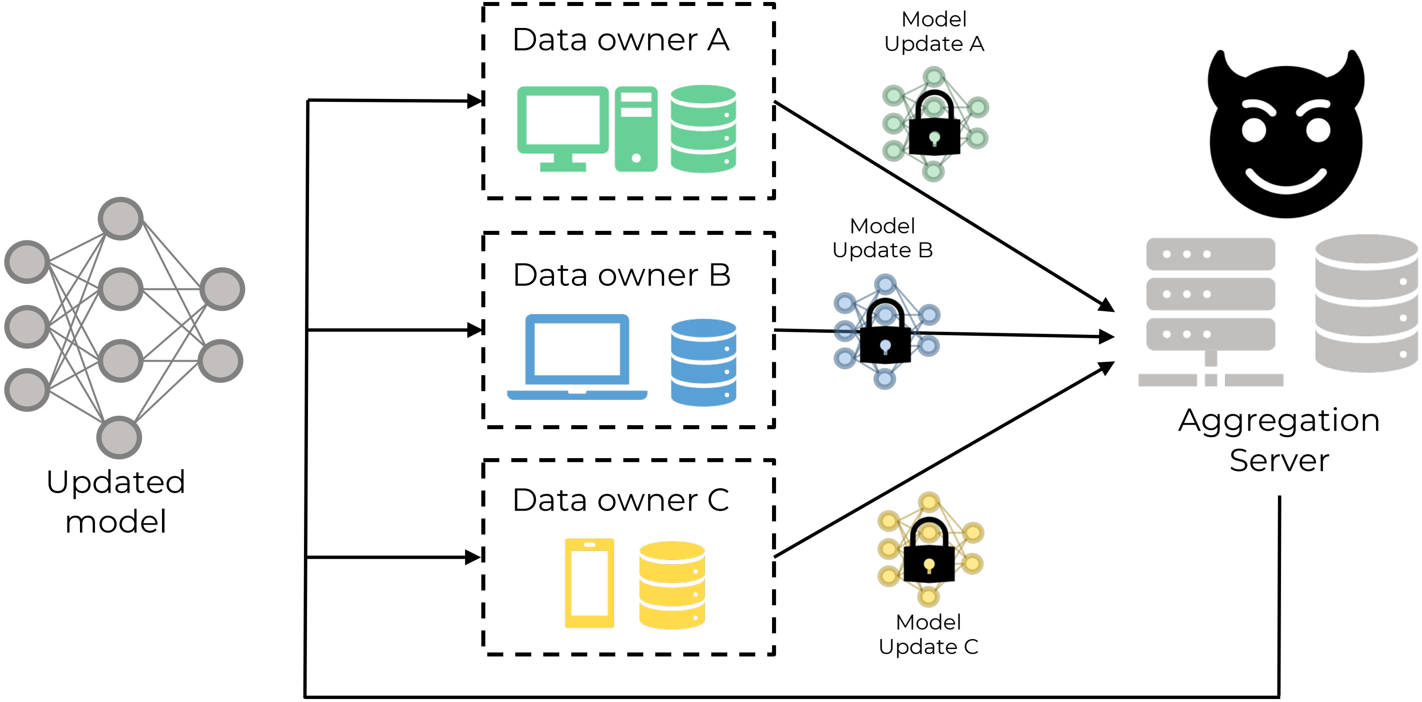
Collaborator/Client

Algorithm: **LocalUpdate**

$w \leftarrow$ global model from Server
for each epoch $s \in 1, \dots, S$:
    for each batch b:
        $g_b \leftarrow$ compute gradient for b
        $w \leftarrow w - \eta g_b$
send w to the Server

Not safe with honest-but-curious servers

Local Differential Privacy



FedAvg + Local DP

Aggregation Server

Algorithm: **FedAvg**

```
1.  Initialize model w̄_0
2.  for each round t=1,…:
3.     Broadcast w̄_{t-1}
4.     select C eligible participants
5.     foreach|| participant p:
6.        w_t^p ← LocalUpdate(p)
7.     w̄_t ← aggregate(∀p w_t^p)
```

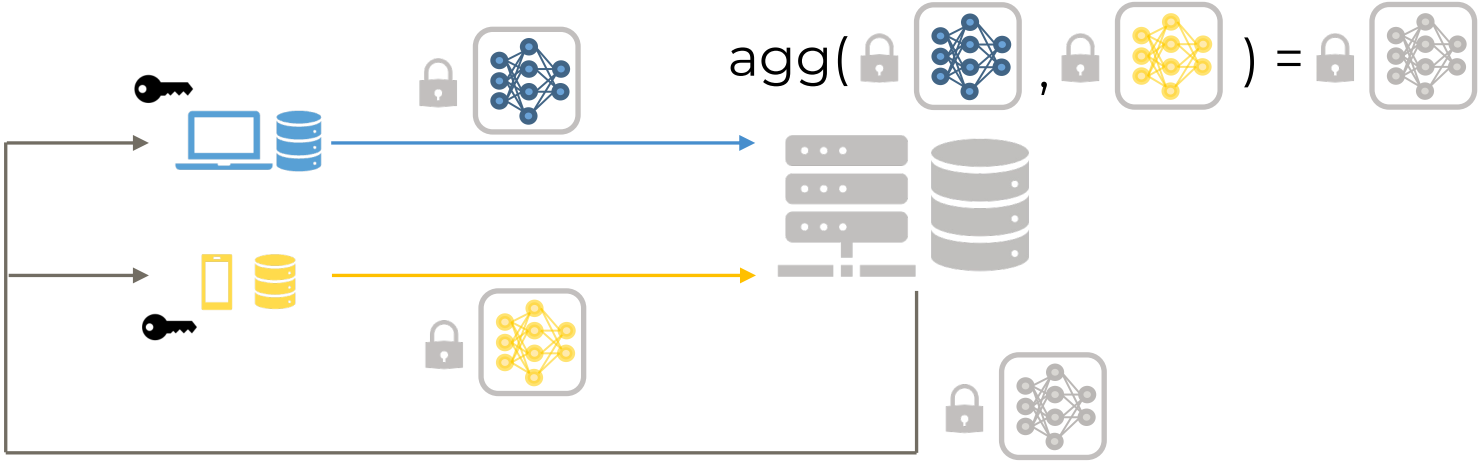
Collaborator/Client

Algorithm: **LocalUpdate**

```
1.  w ← global model from Server
2.  for each epoch s ∈ 1,…,S:
3.     for each batch b:
4.        g_b ← compute gradient for b
5.        w ← w − η g_b
6.  send (w + noise) to the Server
```

It may affect the performance

Homomorphic Encryption



FedAvg + HE

Aggregation Server

Algorithm: **FedAvg**

Initialize model $\overline{w}_0$
for each round t=1,…:
  Broadcast $\overline{w}_{t-1}$
  select C eligible participants
  foreach|| participant p:
    $\mathrm{enc}(w_t^p) \leftarrow$ LocalUpdate(p)
  $\overline{w}_t \leftarrow$ aggregate_he($\forall p\ \mathrm{enc}(w_t^p)$)

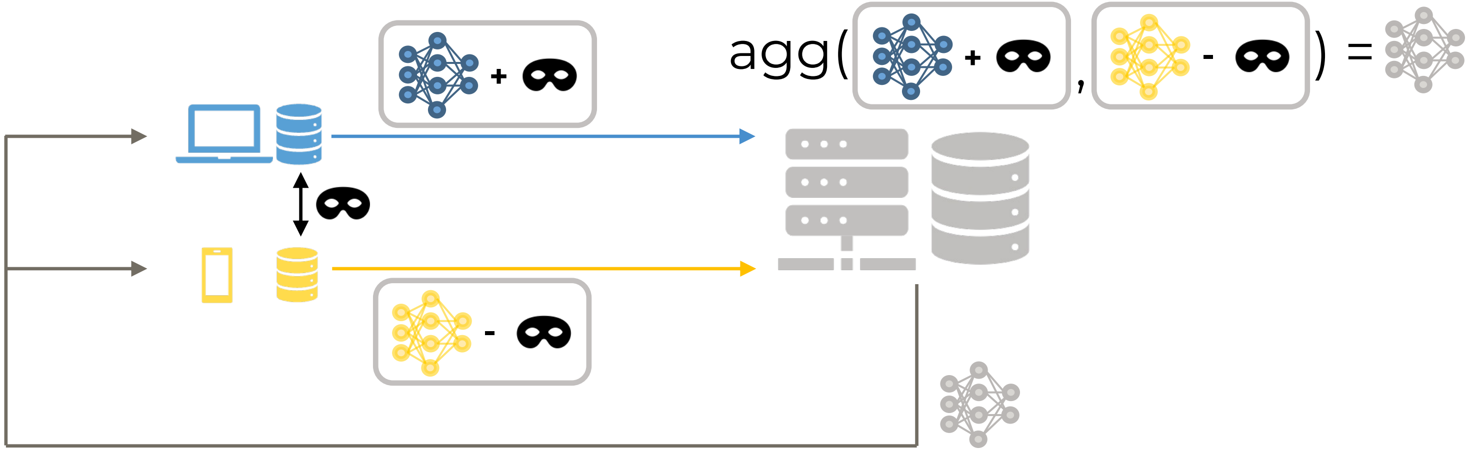
Collaborator/Client

Algorithm: **LocalUpdate**

$w_{enc} \leftarrow$ global model from Server
$w \leftarrow$ decrypt(w_{enc})
for each epoch $s \in 1, \dots, S$:
  for each batch b:
    $g_b \leftarrow$ compute gradient for b
    $w \leftarrow w - \eta g_b$
send encrypt(w) to the Server

Computationally expensive

Secure Multiparty Computation



FedAvg + SMC

Aggregation Server

Algorithm: **FedAvg**

initialize model $\overline{w}_0$
for each round t=1,…:
  Broadcast $\overline{w}_{t-1}$
  select C eligible participants
  foreach|| participant p:
    $\text{mask}(w_t^p) \leftarrow$ LocalUpdate(p)
  $\overline{w}_t \leftarrow$ agg_smc($\forall p$ $\text{mask}(w_t^p)$)

Weigh down the communication protocol

Collaborator/Client

Algorithm: **OneTimePadAgreement**

For each active client p:
  agree on perturbation s_p

Algorithm: **LocalUpdate**

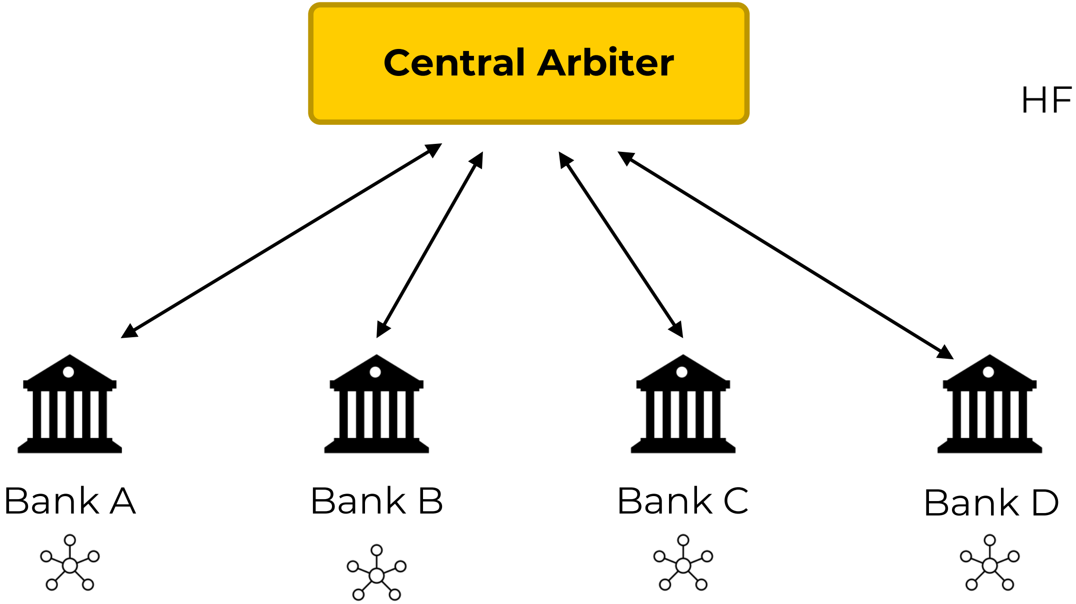
$w \leftarrow$ global model from Server
for each epoch $s \in 1,\dots,S$:
  for each batch b:
    $g_b \leftarrow$ compute gradient for b
    $w \leftarrow w - \eta g_b$
send mask(w, $\forall p$ s_p) to the Server

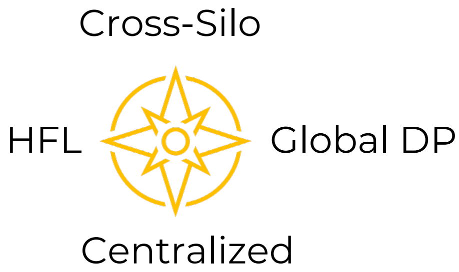
5

Applications

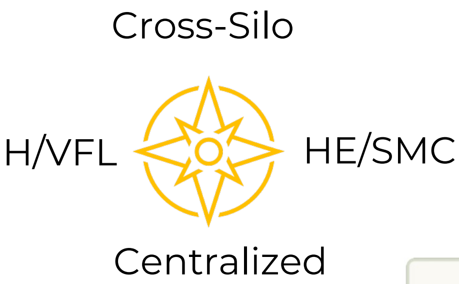
Some use cases and real-world examples of FL

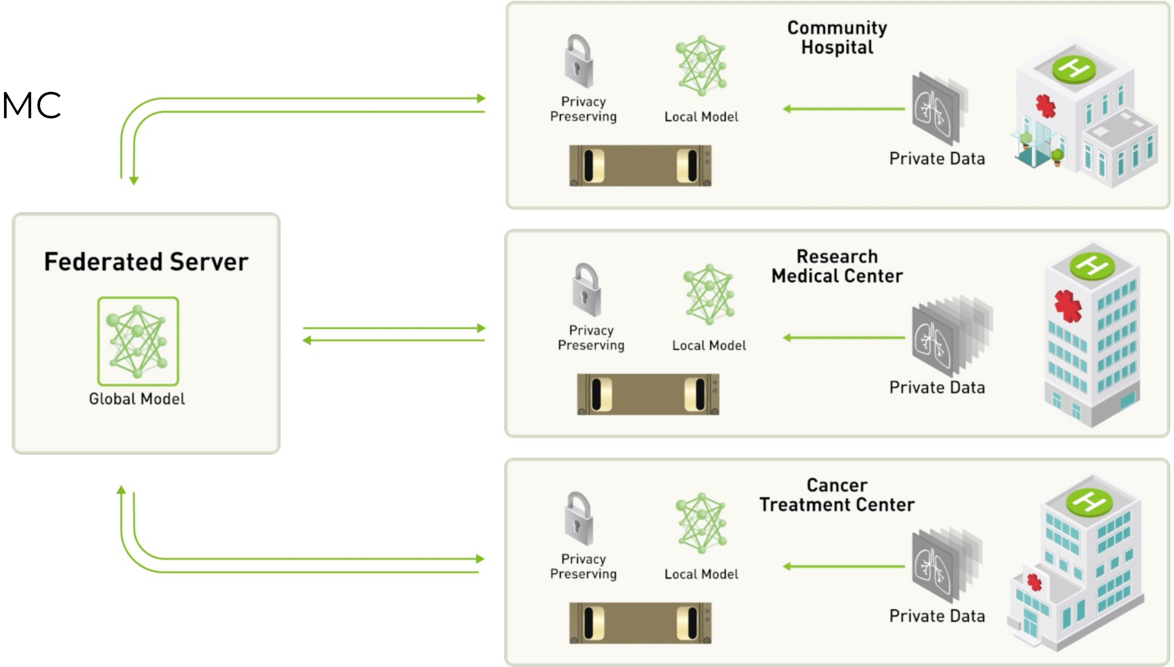
Use case 1
Anti-money laundering



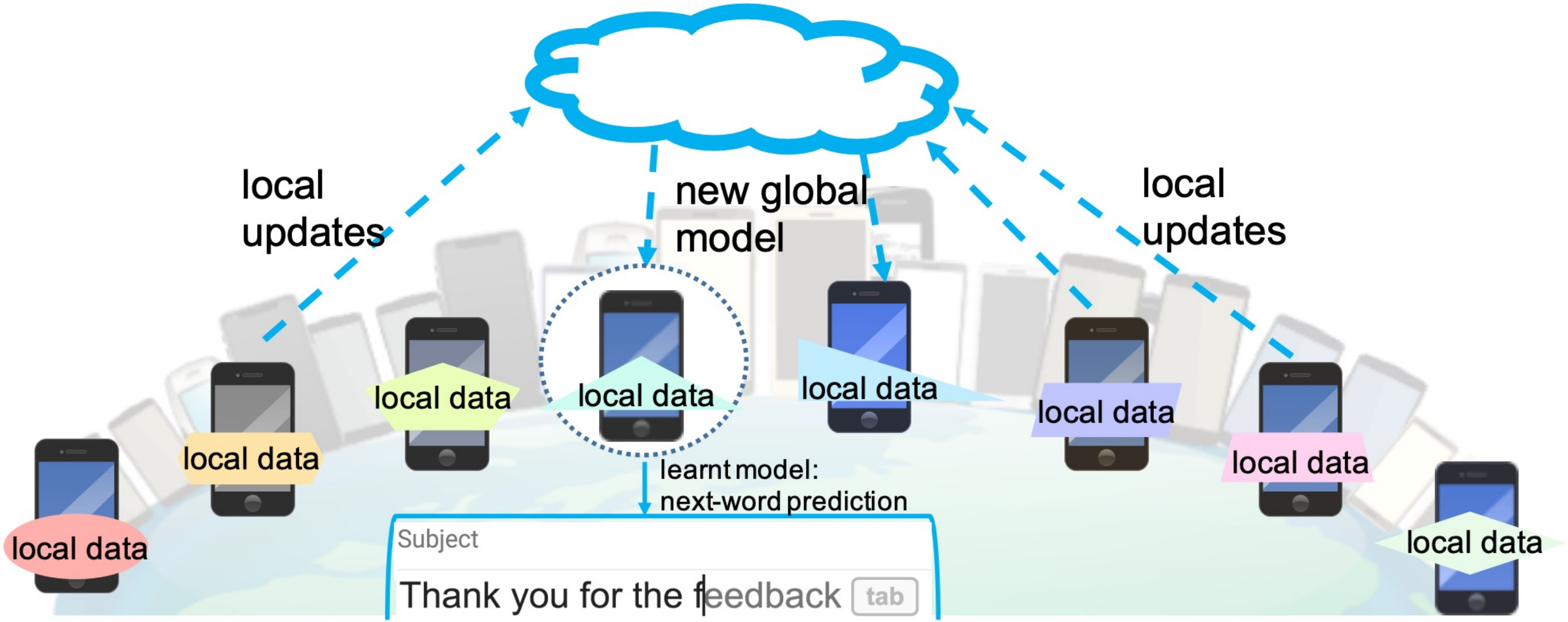


Use case 2
Medical diagnosis

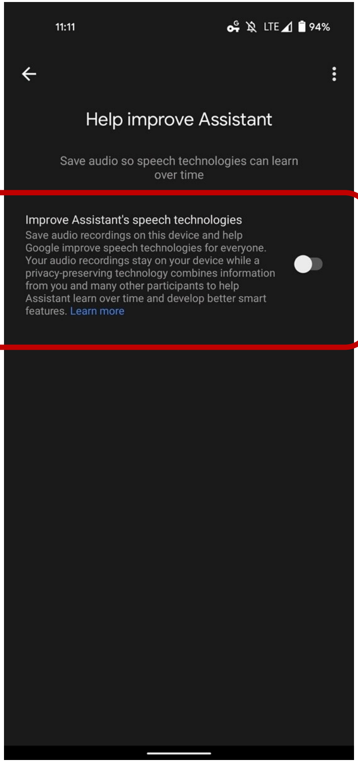


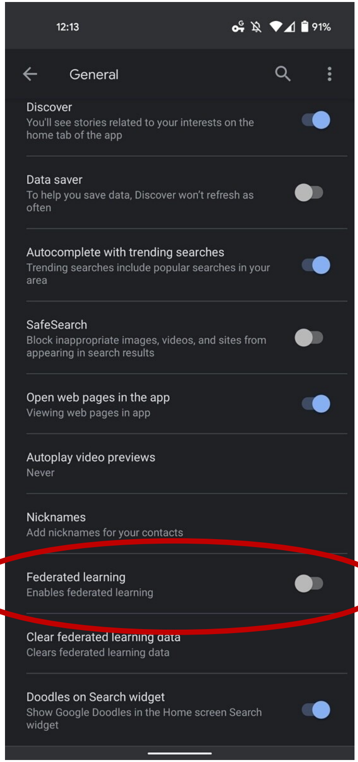


Google's GBoard



Google's "Hey Google!" recognition





6

Conclusions

The glorious “take home message”

What we did not cover

- Attacks to FL systems
- Federated Transfer Learning
- Improve communication efficiency, e.g., model quantization
- Fairness

Take home message ☺

- FL is a “novel” yet **interesting framework for privacy-preseving ML**
- FL methods must be designed considering the **communication-computation-privacy-effectiveness trade-off**
- FL is still in its infancy and there are **many open problems**

Thanks!

Any Questions?

The only stupid question is the one you were afraid to ask but never did.

Richard Sutton

Resources

- Li, et al. 'A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection', 2021. http://arxiv.org/abs/1907.09693.
- Kairouz, et al. 'Advances and Open Problems in Federated Learning'. Foundations and Trends in Machine Learning 14, 2021. https://doi.org/10.1561/2200000083.
- Yang et al. 'Federated Learning'. Synthesis Lectures on Artificial Intelligence and Machine Learning 13, 2019. https://doi.org/10.2200/S00960ED2V01Y201910AIM043.
- Li, et al. 'Federated Learning: Challenges, Methods, and Future Directions', 2019. https://arxiv.org/abs/1908.07873
- Bonawitz, et al. 'Practical Secure Aggregation for Federated Learning on User-Held Data'. 2016. http://arxiv.org/abs/1611.04482.
- McMahan, et al. 'Communication-efficient learning of deep networks from decentralized data', 2016. https://arxiv.org/abs/1602.05629