Federated Learning

A gentle introduction to a novel collaborative learning approach

Mirko Polato, Assistant Professor
University of Torino, Department of Computer Science
mirko.polato@unito.it
27.05.2022

TABLE OF CONTENTS

01

Neural Network Refresher

Goal of Machine Learning

- Find a **function** that given an input produces a desired output

TASK	INPUT	OUTPUT
Image Classification		6
Next word(s) prediction		
Playing chess		
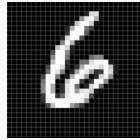
(Deep) Neural Networks

$\mathbf{x}$ $\mathbf{W}^{(1)}$ $\mathbf{z}$ $\mathbf{W}^{(2)}$

$w_{1,1}^{(1)}$

$w_1^{(2)}$

$w_2^{(2)}$

$w_{3,1}^{(1)}$

$\hat{y}$

Neuron (Perceptron)

x_1 w_1

x_2 w_2

w_3

x_3

$$z = \sigma(\sum_{i=1}^{3} w_i x_i) = \sigma(\mathbf{w}^\top \mathbf{x})$$

Activation functions

Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$
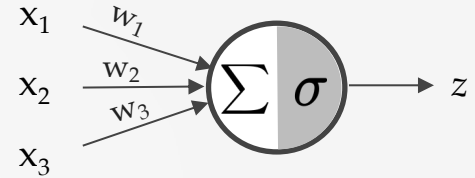
tanh
$\tanh(x)$

ReLU
$\max(0, x)$

Family of functions:

$$f(\mathbf{x}; \theta) = \sigma_2(\mathbf{W}^{(2)} \sigma_1(\mathbf{W}^{(1)} \mathbf{x}))$$

$$\theta \equiv \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}$$
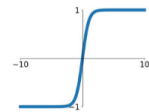
Finding the "best" function
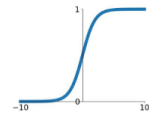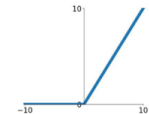
- Given a training dataset containing n input-output pairs $(\mathbf{x}_i, \mathbf{y}_i)$ the goal of deep learning model training is to find a set of parameters $\boldsymbol{\theta}$, such that to maximize (on average) $p(\mathbf{y}=\mathbf{y}_i \mid \mathbf{x}_i)$

- The loss function defines what we want to optimize and it is a function of the model parameters and the training examples

$$\min_{\theta} \mathcal{L}(\mathbf{X}, \mathbf{y}; \theta)$$

where

$$\mathcal{L}(\mathbf{X}, \mathbf{y}; \theta) = \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{x}_i, y_i; \theta)$$

Stochastic Gradient Descent (SGD)

$l(\ ;\boldsymbol{\theta})$

$\boldsymbol{\theta}_0$

incremental step

cost function

gradient

$\boldsymbol{\theta}$

Full Gradient

Starting Point

Optimum

Solution

Stochastic Gradient
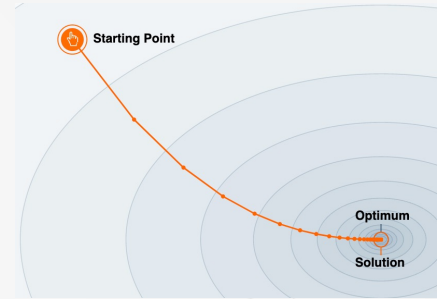
Starting Point

Optimum

Solution
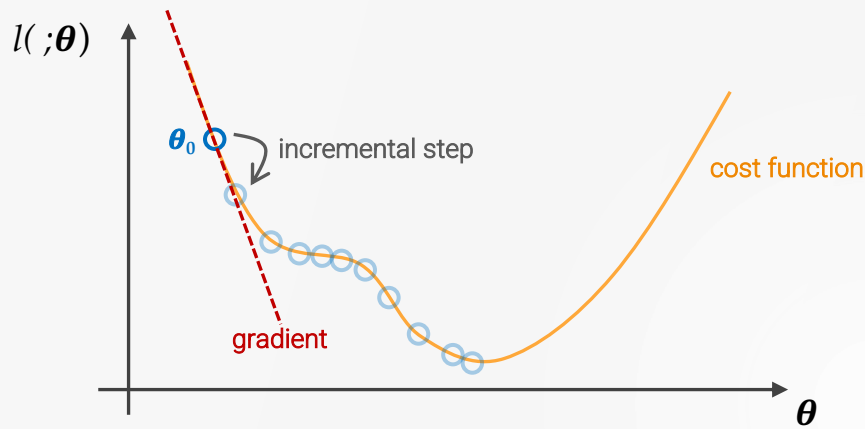
1. Randomly initialize $\boldsymbol{\theta}_0$

2. For t = 1, 2, … do

3. Pick a random training instance $(\mathbf{x}_i\,,\,y_i)$

4. $\theta_t \leftarrow \theta_{t-1} - \gamma \nabla l(\mathbf{x}_i, y_i; \theta_{t-1})$

Learning rate

For efficiency reasons this is usually a mini-batch of examples

02

Introduction to Federated Learning

Motivation

- On one hand, access to (private) data is becoming increasingly challenging
  - Users awareness about privacy of their data
  - Data centralization may be not possible due to legal constraints
  - Legislation, e.g., GDPR, HIPAA…

- On the other hand, machine leaning models, especially deep learning, need a huge amount of data to be properly trained

- We need a privacy-preserving collaborative/distributed learning approach

The biggest obstacle to using advanced data analysis isn't skill base or technology; It's plain old access to the data

Edd Wilder-James, Harvard Business Review

Federated Learning: general idea

Federated Learning (FL) 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.

- Central server, called **aggregator**, orchestrate the learning process
- **Clients** (aka users) own (usually small) amount of private data to be used for training the model
  - **Cross-device FL**: potentially million of clients, relatively small local datasets
  - Cross-silo FL: relatively few clients (<100) with large local datasets

- **Horizontal FL**: each client owns a set of training examples
- Vertical FL: each client owns a subset of the features of (potentially) all the examples

Federated Learning assumptions, goals & desiderata

ASSUMPTIONS
- Model parameters do not contain more information than the raw training data
- The size of the model is *generally* smaller than the size of the raw training data

GOALS
- Confidentiality: clients do not share their data
- Usefulness: clients benefit from the federation

DESIDERATA
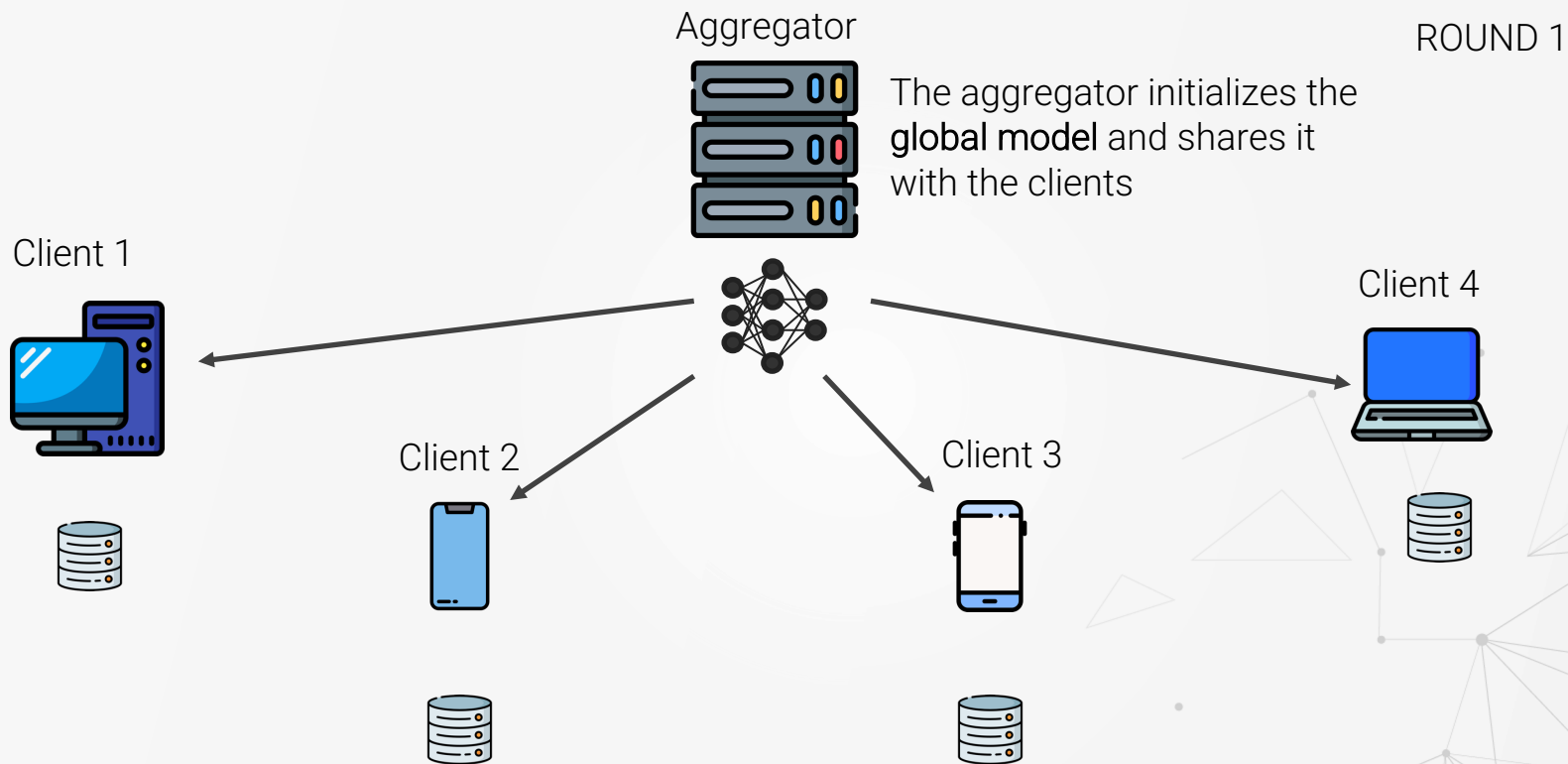- The federated model is close to the "ideal" one

Federated Learning: major challenges

- **Non-IID**: the data generated by each user are quite different

- **Unbalanced**: some users produce significantly more data than others

- **Massively distributed**: mobile device owners ≫ avg # training samples on each device

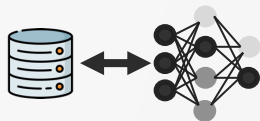- **Limited communication**: unstable mobile network connections
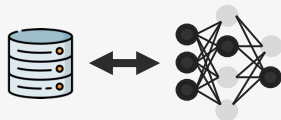
FL: the general protocol

Aggregator

ROUND 1

The aggregator initializes the **global model** and shares it with the clients

Client 1

Client 4

Client 2

Client 3

FL: the general protocol

The clients update the model using their own private data

Aggregator

ROUND 1

Client 1

Client 4

Client 2

Client 3

FL: the general protocol

Aggregator

ROUND 1

The clients send the (local) updated model to the aggregator

Client 1

Client 4

Client 2

Client 3

FL: the general protocol

Aggregator

The aggregator updates the global model aggregating the received ones

ROUND 1

Client 1

Client 4

Client 2

Client 3

FL: the general protocol

Aggregator

The aggregator sends the updated **global model** to the clients

Client 1

Client 4

Client 2

Client 3

and so on....
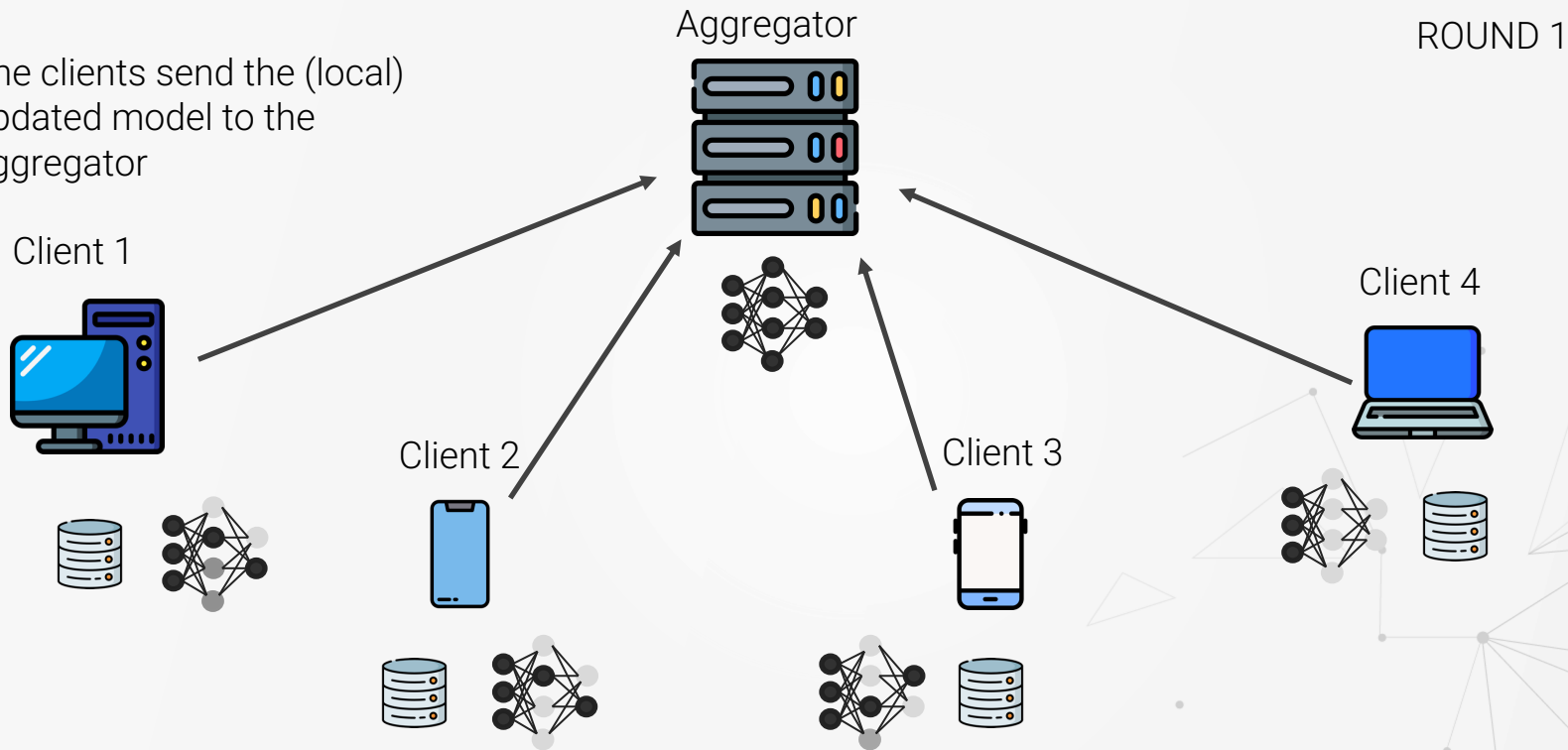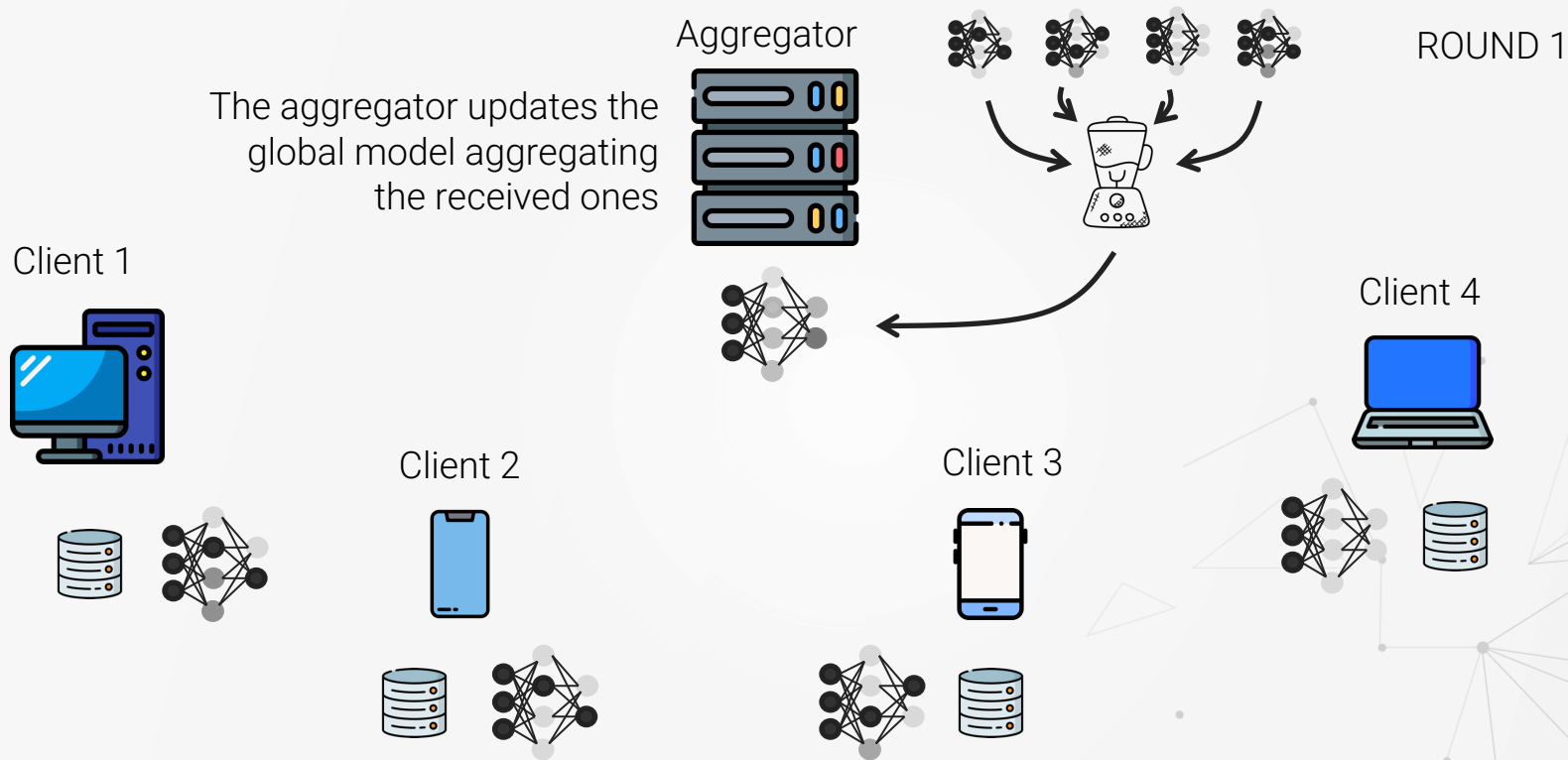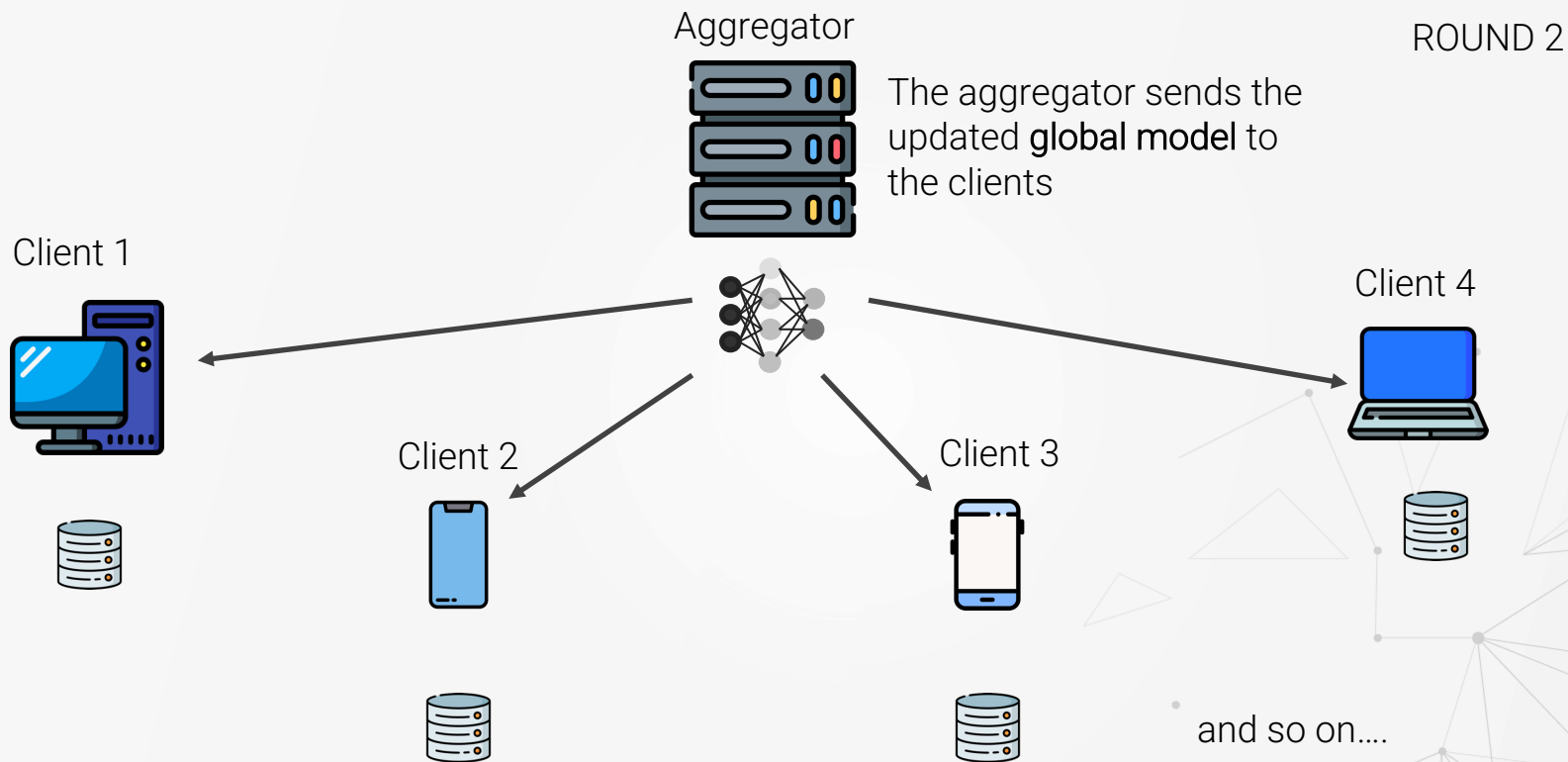
Cross-device FL: a toy example
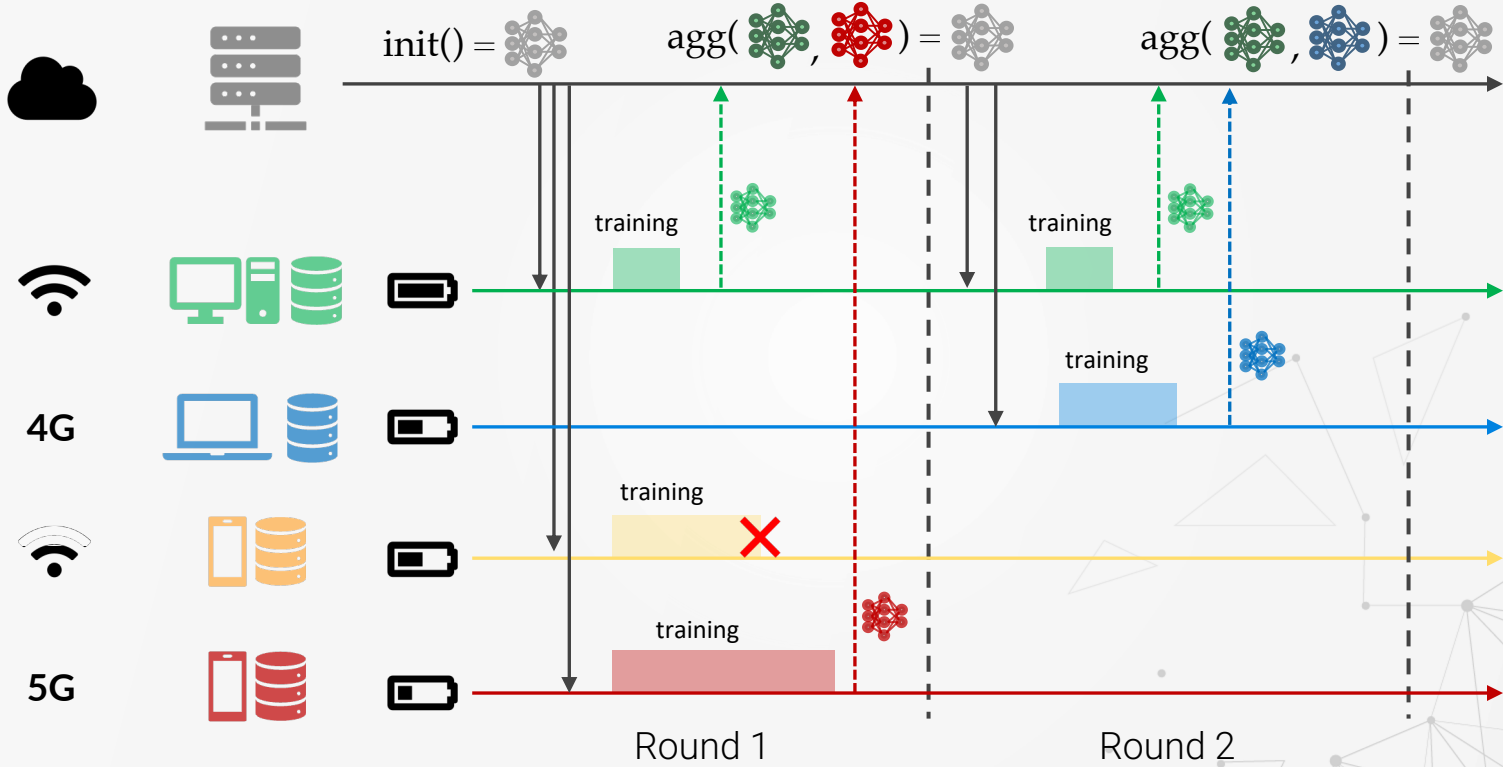
A bit of notation

- n : total number of samples

- K: number of clients

- n_k : number of samples on client k

- η: learning rate

- T: total number of rounds

- t: "current" round

- w: from now on, this indicates the parameters of the model (thus the model itself)

- f : from now on, the loss function

Federated SGD: FedSGD

- <u>Observation</u>: a randomly selected client that has $n_k < n$ training data samples in federated learning ≈ A randomly selected sample/batch in traditional deep learning

- Federated SGD (<u>FedSGD</u>): **a single step of gradient descent** is done per round

- In federated learning only a **C-fraction of clients** are selected at each round.
  - There are many possible selection criteria: on charge, idle...
  - C=1: full-batch (non-stochastic) gradient descent − *Unrealistic*
  - C<1: stochastic gradient descent (SGD)

FedSGD

- We assume the aggregator initialized the global model w

- In a round $t < T$:

  - The aggregator broadcasts the current *global* model w to each client;

  - Each client k computes gradient on its local data (single batch)

  - *Alternative 1:*

    - Each client k submits g_k;

    - The aggregator aggregates the gradients to generate a new global model:

    $$w_{t+1} \leftarrow w_t - \eta \nabla f(w_t) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

  - *Alternative 2:*

    - Each client k computes: $w_{t+1} \leftarrow w_t - \eta g_k$

    - The central server performs aggregation: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

Federated Averaging: FedAvg

- FedSGD communication is highly inefficient

  - A client (participating in a round) sends and receives one model at every (mini-batch) update

- Improve computation efficiency:

  - Selects more client in each round: more reliable gradient estimate

  - Increase the computation on each client 💡

FedAvg

- We assume the aggregator initialized the global model w

- In a round $t < T$:
  - The aggregator broadcasts the current *global* model w to each client;
  - Each client k computes gradient on its local data
  - *Like alternative 2 of FedSGD:*
    - Each client k computes for E epochs: $w_{t+1} \leftarrow w_t - \eta g_k$
      In this case, clients perform local mini-batch SGD

    - The central server performs aggregation: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

If $E=1$ and batch size $= n_k$, then FedSGD=FedAvg
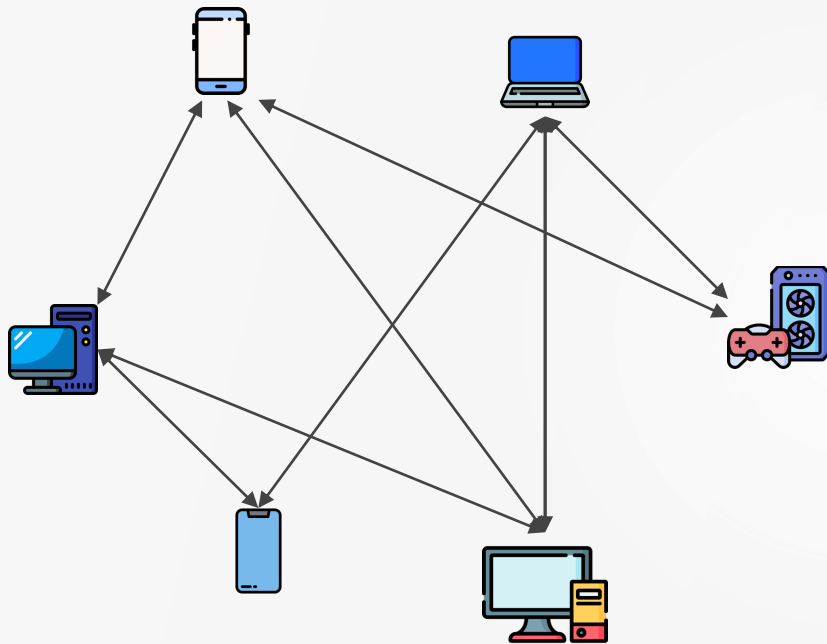
IMPORTANT

FedAvg, good but…

- FedAvg works decently in practice

- However, <u>FedAvg does not guarantees linear convergence</u> for smooth, strongly convex losses

- Intuition
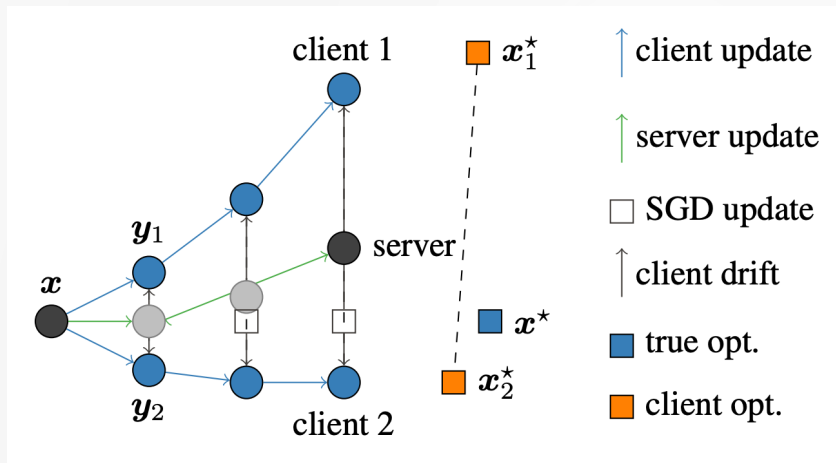
In the figure the model parameters are called x and y

Decentralized Federated Learning

Algorithm   Gossip Learning Framework

1: $(t_k, w_k) \leftarrow \text{init}()$
2: **loop**
3:      $\text{wait}(\Delta_g)$
4:      $p \leftarrow \text{select}()$
5:      send $(t_k, \text{compress}(w_k))$ to p
6: **end loop**

7: **procedure** ONRECEIVEMODEL(t_r, w_r)
8:      $(t_k, w_k) \leftarrow \text{merge}((t_k, w_k), (t_r, w_r))$
9:      $(t_k, w_k) \leftarrow \text{update}((t_k, w_k), D_k)$
10: **end procedure**

03

Privacy in Federated Learning

Is FL really privacy-preserving?

- Privacy in Federated learning is based on the fact that private data does not leave the device but only the model that is being trained

- But, may the model leak information about the training data?

  - Unfortunately, yes!

Theorem: Consider a neural network containing a biased fully-connected layer preceded solely by (possibly unbiased) fully-connected layers. Furthermore assume for any of those fully connected layers the derivative of the loss w.r.t. to the layer's output contains at least one non-zero entry. Then the input to the network can be reconstructed uniquely from the network's gradients.

IMPORTANT

Type of attackers in FL

- We can identify two types of attackers:

  - **Semi-Honest:** adversaries are considered passive/honest-but-curious. They try to learn the private states of other participants without deviating from the FL protocol. The adversaries can only observe the received information, i.e., parameters of the global model.

  - **Malicious:** adversaries who try to learn the private states of honest participants, by arbitrarily deviating from the FL protocol by modifying, re-playing, or removing messages.
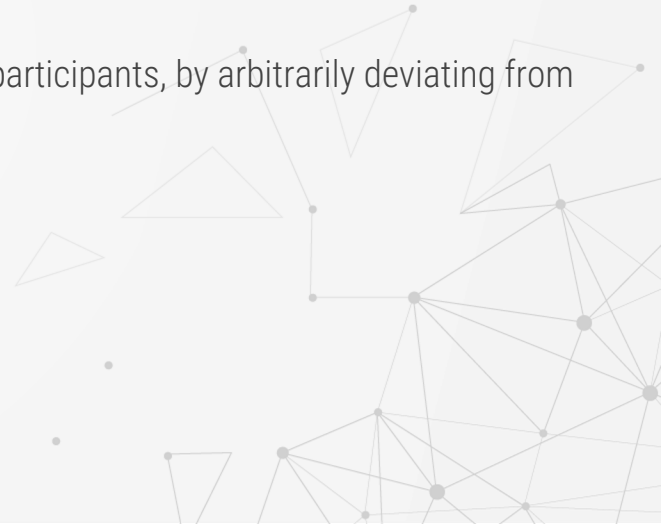
Privacy threats in FL

- The main attacks to users' privacy are:

  - **Membership inference:** the goal is to infer whether some (given) data belongs to the training dataset. Under FL, it is even possible to suggest which user owns the dataset. The attacker aim is to infer if some data piece $\{(\mathbf{x}, \mathbf{y})\}$ belongs to a local dataset

  - **Model inversion:** a machine learning model (e.g., GAN) is trained to infer the e class conditional distribution $p(\mathbf{x}|\mathbf{y})$. In other words, the attacker tries to infer the training data.

Standard defenses against privacy attacks
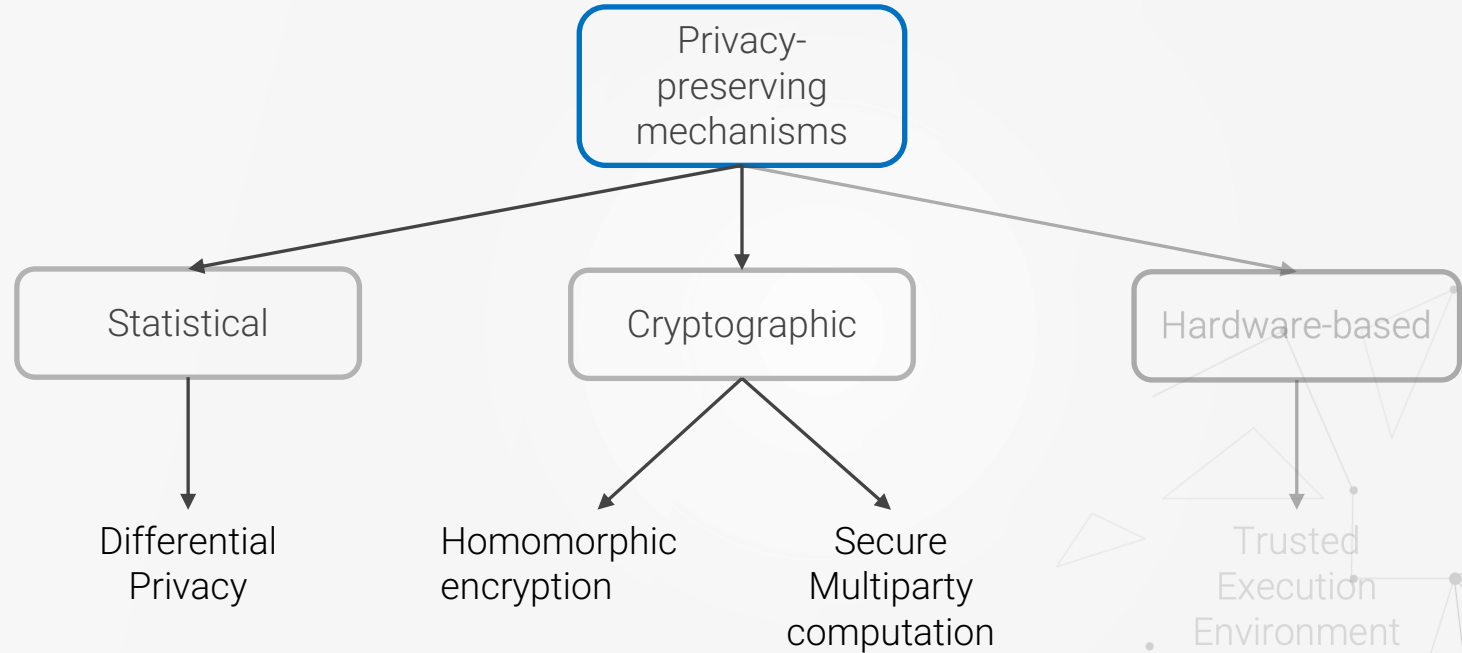
Homomorphic Encryption
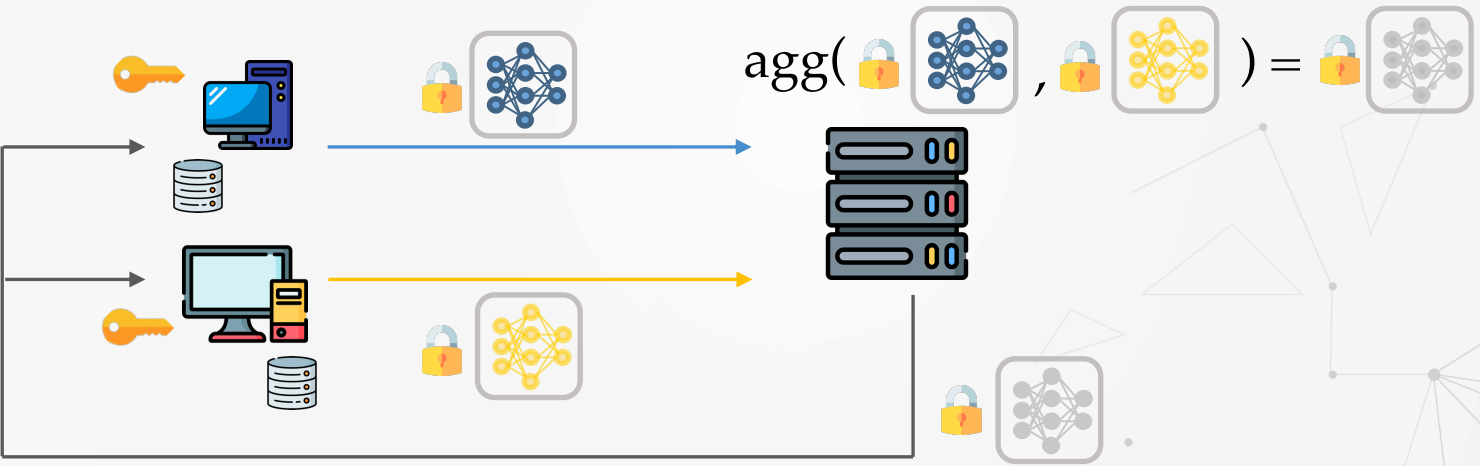
Differential Privacy

Differential Privacy (DP) is a privacy-preserving mechanism that adds noise to the model for limiting a wide range of attacks. For example, an Inference attack, such as Model Inversion, will contain noise, degrading attack efficiency.

- LOCAL Differential Privacy: the noise to the model is added by the clients before sending the model to the server
    - This mechanism protects the privacy even in case of a malicious server, i.e., it sees only noisy models
    - All the model updates are noisy affecting the overall training process degrading the final performance

- GLOBAL Differential Privacy: the noise is added only server-side
    - Generally, the training process is less affected w.r.t. local DP. Still not ideal
    - The server can see the *plain* model updates sent by the clients

Secure Aggregation (& SMC)

- **Secure Aggregation** is a class of Secure Multi-Party Computation (SMC) algorithms wherein a group of mutually distrustful parties collaborate to compute an aggregate value without revealing to one another any information about their private value except what is learnable from the aggregate
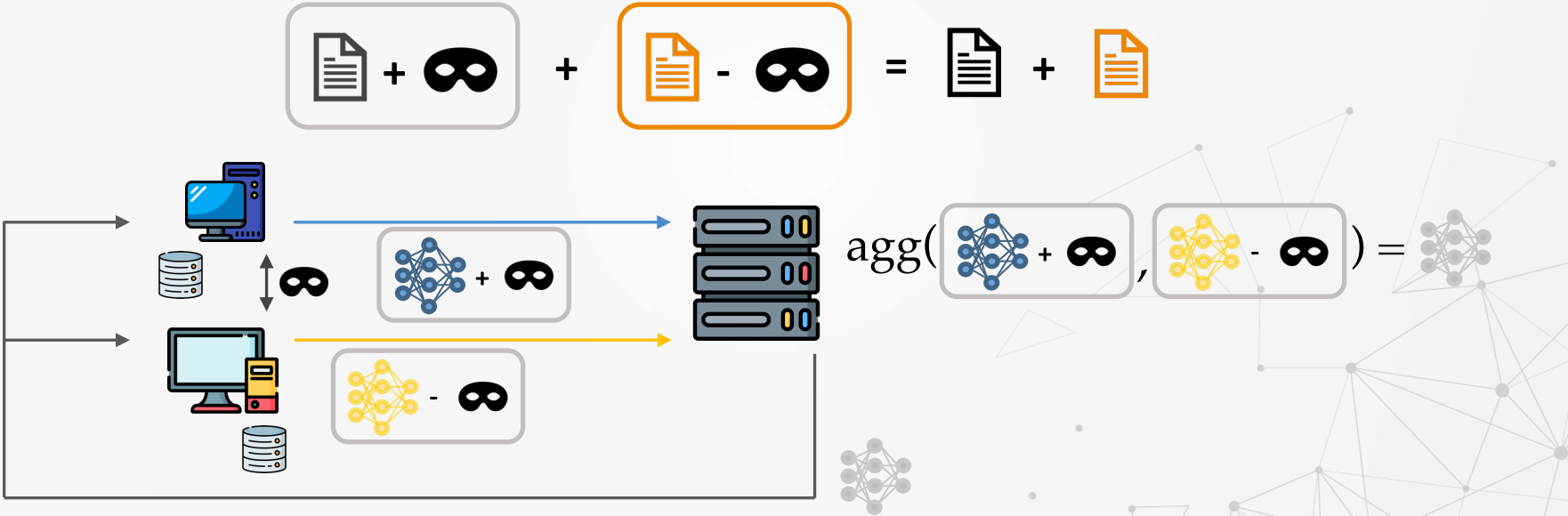
Secure Aggregation: an example

ASSUMPTION: all parties complete the protocol and possess pair-wise secure communication channels with ample bandwidth

1. Each pair of users u,v first agree on a matched pair of input perturbations.
   That is, user u samples a vector $s_{u,v}$ uniformly from $[0, R)^k$

2. For each other user v.

   ○ Users u and v exchange $s_{u,v}$ and $s_{v,u}$ over their secure channel and compute perturbations
   $p_{u,v} = s_{u,v} - s_{v,u} \pmod{R}$, noting that $p_{u,v} = -p_{v,u} \pmod{R}$ and taking $p_{u,v} = 0$ when $u = v$.

3. Each user sends to the aggregator: $\qquad y_u = x_u + \sum_{v \in \mathcal{U}} p_{u,v} \pmod{R}$

4. The server simply sums the perturbed values: $\qquad \bar{x} = \sum_{u \in \mathcal{U}} y_u \pmod{R}$

Correctness is guaranteed because the paired perturbations cancel each other out:

$$\bar{x} = \sum_{u \in \mathcal{U}} x_u + \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} p_{u,v} = \sum_{u \in \mathcal{U}} x_u + \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} s_{u,v} - \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} s_{v,u} = \sum_{u \in \mathcal{U}} x_u \pmod{R}$$
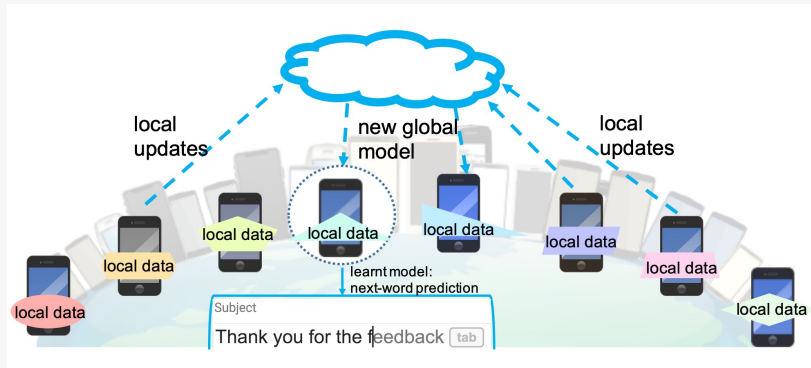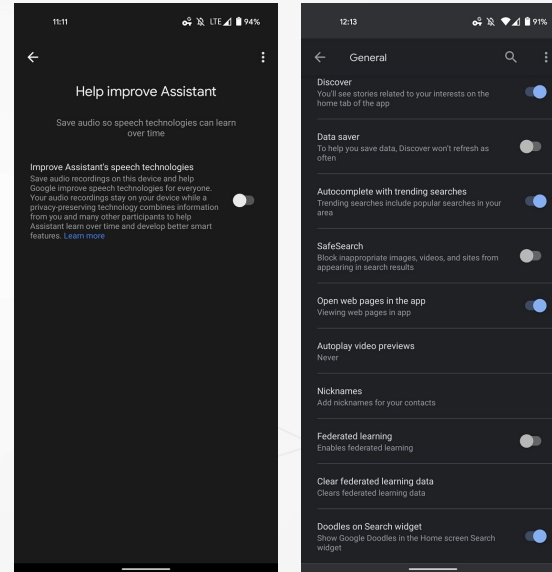
04

Wrapping up!

Is Federated Learning used, yet?

Google's GBoard

Google's "Hey Google!" recognition

Take home message!

- Federated Learning represents a step towards privacy-preserving collaborative learning

- FL is still in its infancies and many problems regarding its application are still far from being solved

- FL alone do not guarantee perfect privacy

- The state-of-the-art privacy-preserving mechanism have serious drawbacks,
e.g., computational and/or perfomance-wise
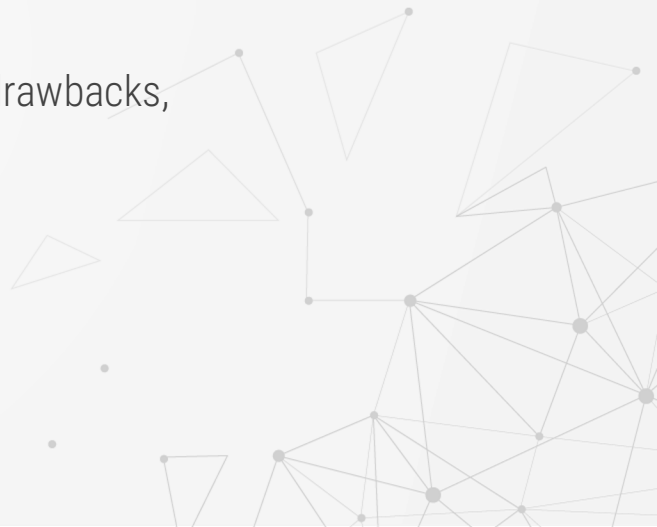
We are "hiring" ☺

Possible thesis topics:

- Development of novel FL approaches for *classic* Machine Learning approaches (not based on SGD)

- Study of the privacy/security in decentralized FL setting, e.g., gossip learning

- Study of the effect of DP techniques on different model architectures & tasks