Security & Privacy in 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
15.06.2022

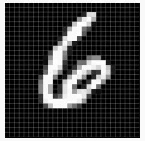
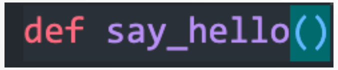
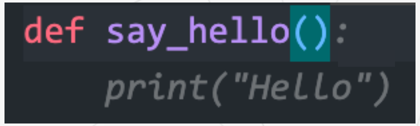
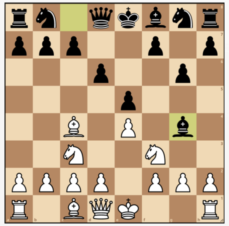
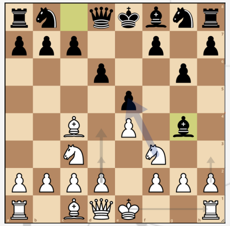
TABLE OF CONTENTS

01

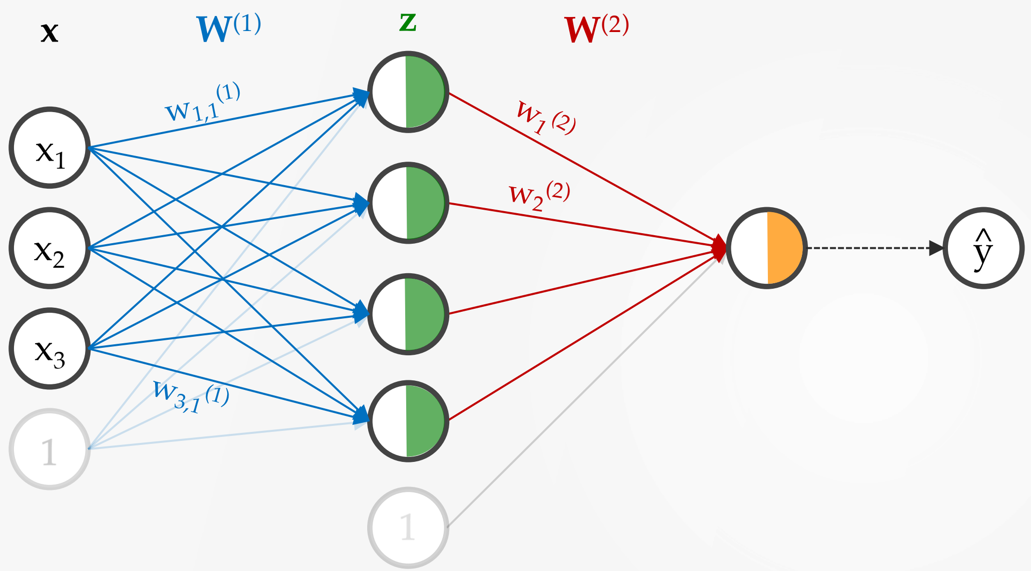
Neural Network Refresher

Goal of Machine Learning

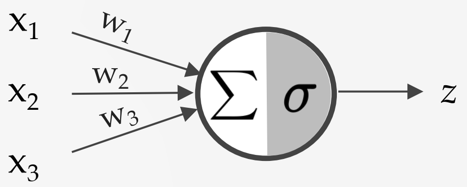
- Find a **function** that given an input produces a desired output

TASK	INPUT	OUTPUT
Image Classification		6
Next word(s) prediction	 `def say_hello()`	 `def say_hello():` `print("Hello")`
Playing chess		

(Deep) Neural Networks



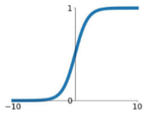
Neuron (Perceptron)



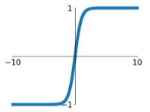
$$z = \sigma\left(\sum_{i=1}^{3} w_i x_i\right) = \sigma(\mathbf{w}^\top \mathbf{x})$$

Activation functions

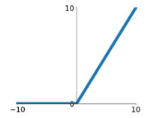
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



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 $\mathbf{X}, \mathbf{y}$ containing $\mathbf{n}$ input-output pairs $(\mathbf{x}_i, y_i)$ the goal of deep learning model training is to find a set of parameters $\boldsymbol{\theta}$, such that to maximize (on average) $p(\tilde{y}=y_i \mid \mathbf{x}_i)$, where $\tilde{y} = f(\mathbf{x}_i; \boldsymbol{\theta})$

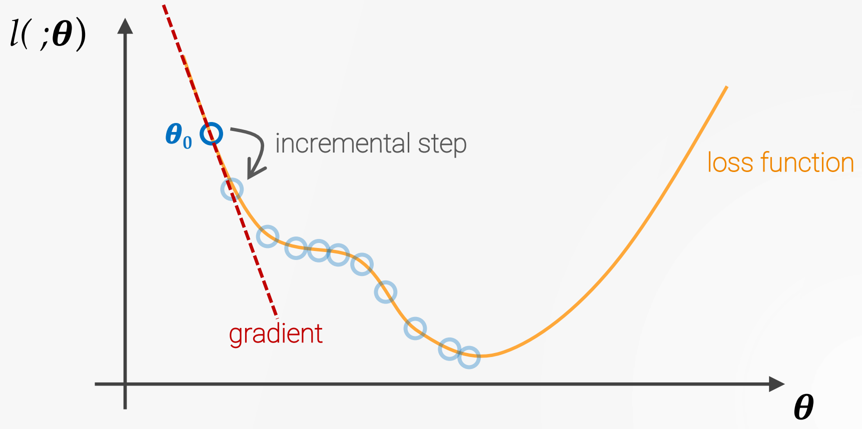
- The **loss function** defines what we want to optimize and it is a function of the model parameters and the training examples

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

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)

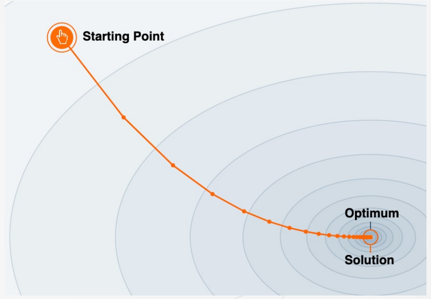


```
1.  Randomly initialize $\boldsymbol{\theta}_0$
2.  For t = 1, 2, … do
3.      Pick a random training instance $(\mathbf{x}_i, y_i)$
4.      $\theta_t \leftarrow \theta_{t-1} - \gamma \nabla l(\mathbf{x}_i, y_i; \theta_{t-1})$
```

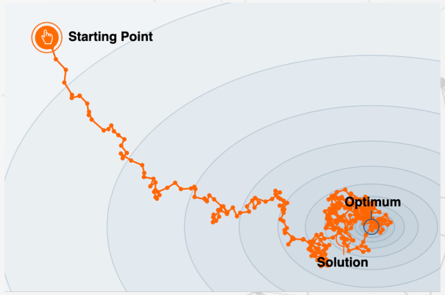
Learning rate

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

Full Gradient



Stochastic Gradient



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

H.McMahan, Eider Moore, D.Ramage, S.Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In AISTATS 2017

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

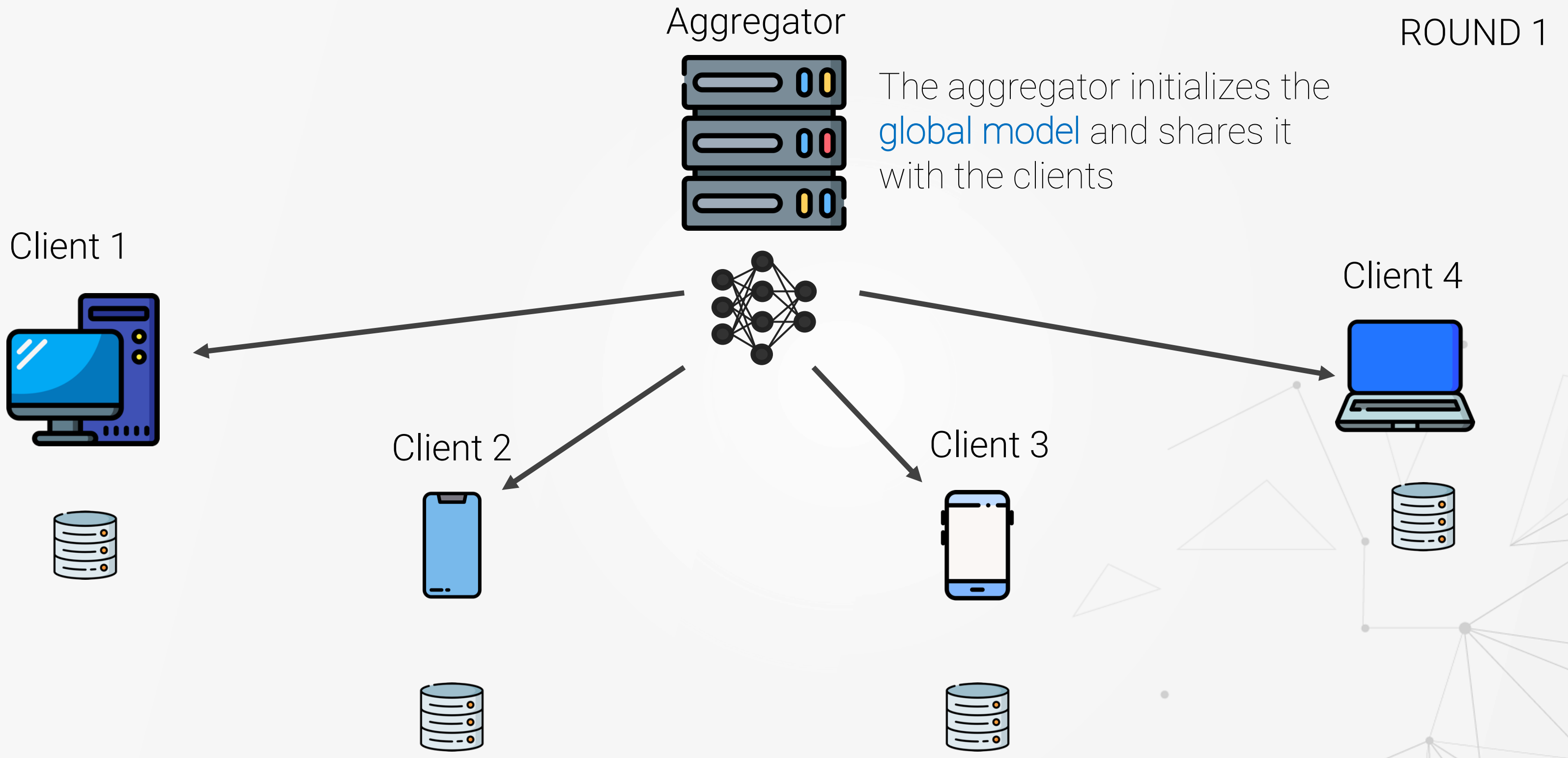
DESIDERATA

- 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 $\gg$ avg # training samples on each device
- **Limited communication**: unstable mobile network connections

FL: the general protocol

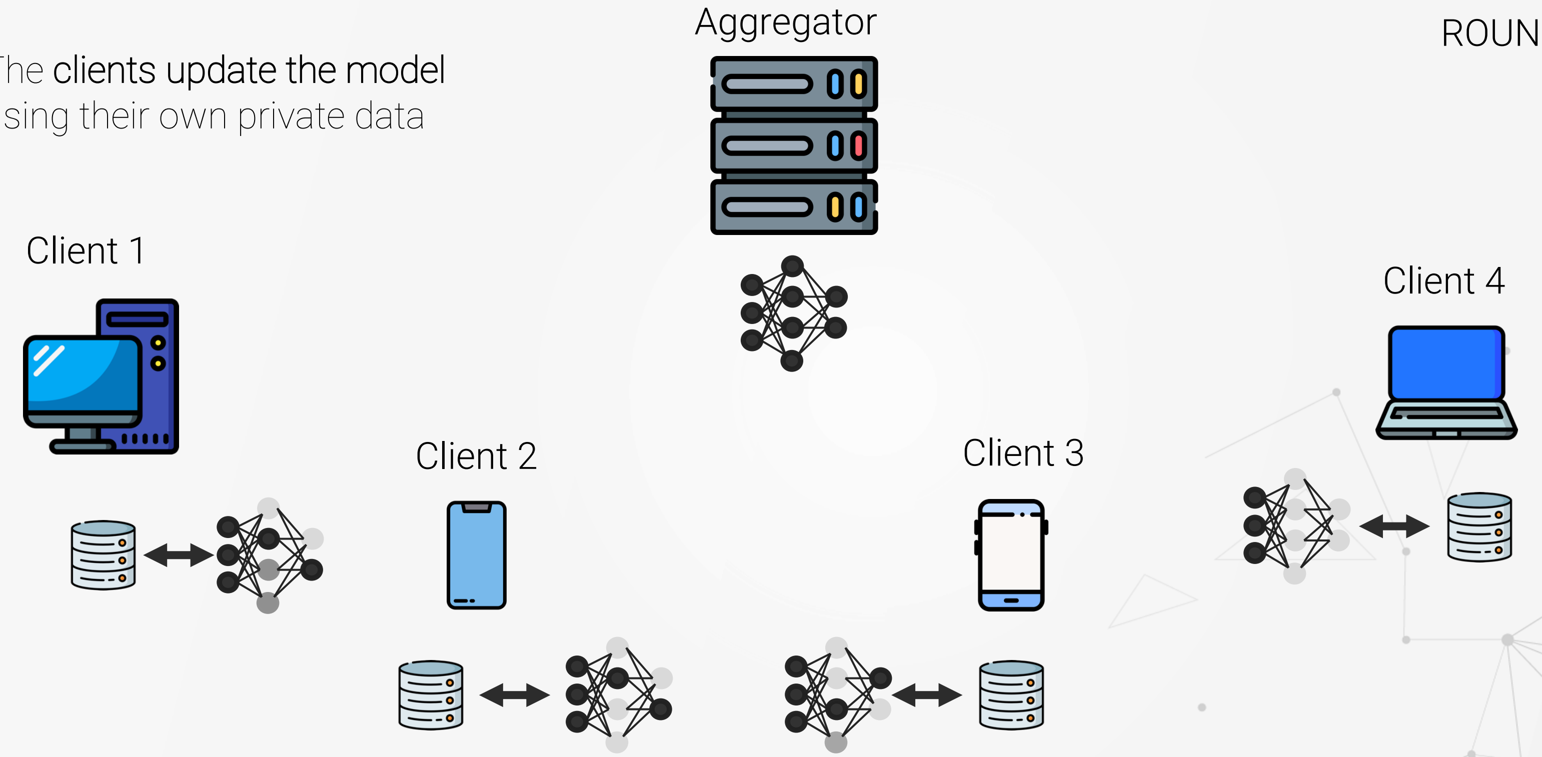


The aggregator initializes the global model and shares it with the clients

FL: the general protocol

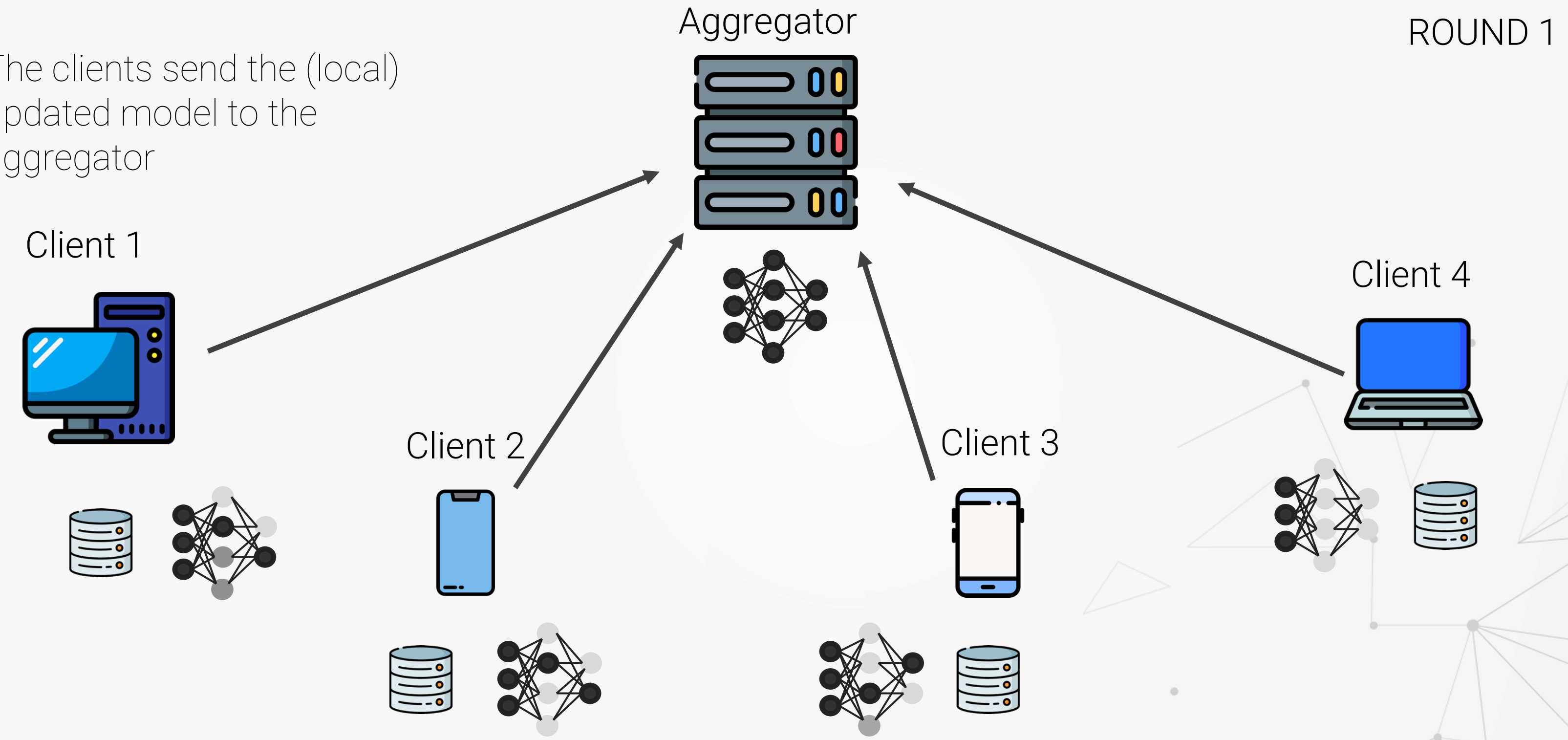
The **clients update the model** using their own private data

ROUND 1



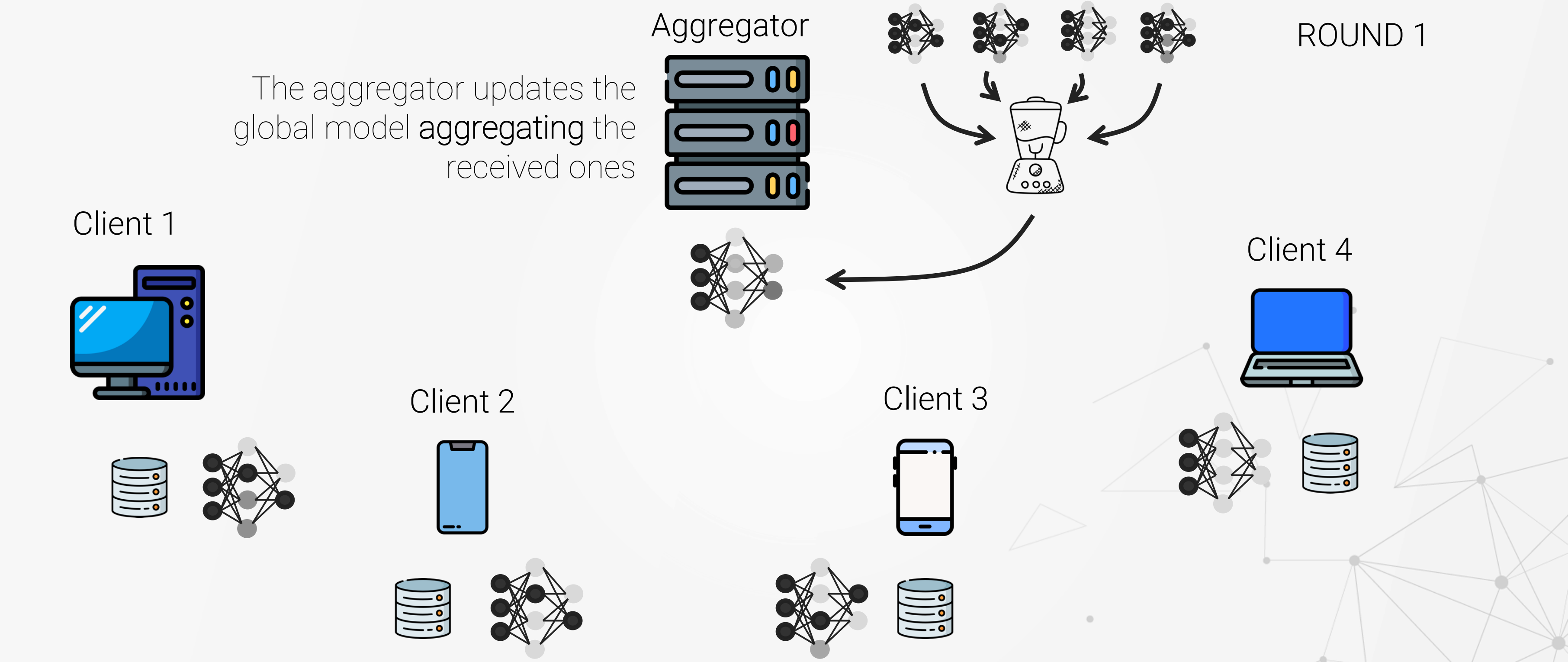
FL: the general protocol

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



FL: the general protocol

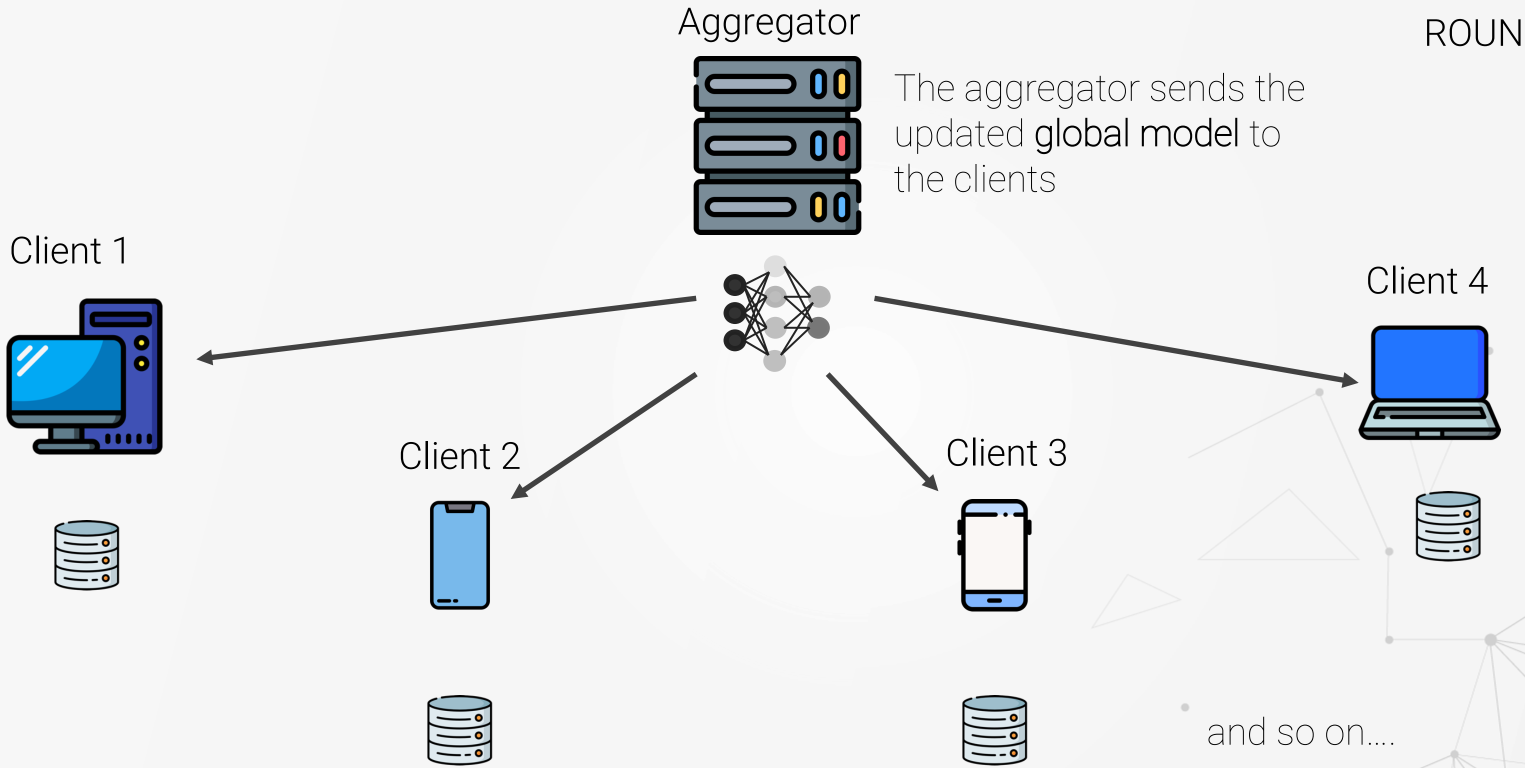
The aggregator updates the global model **aggregating** the received ones



FL: the general protocol

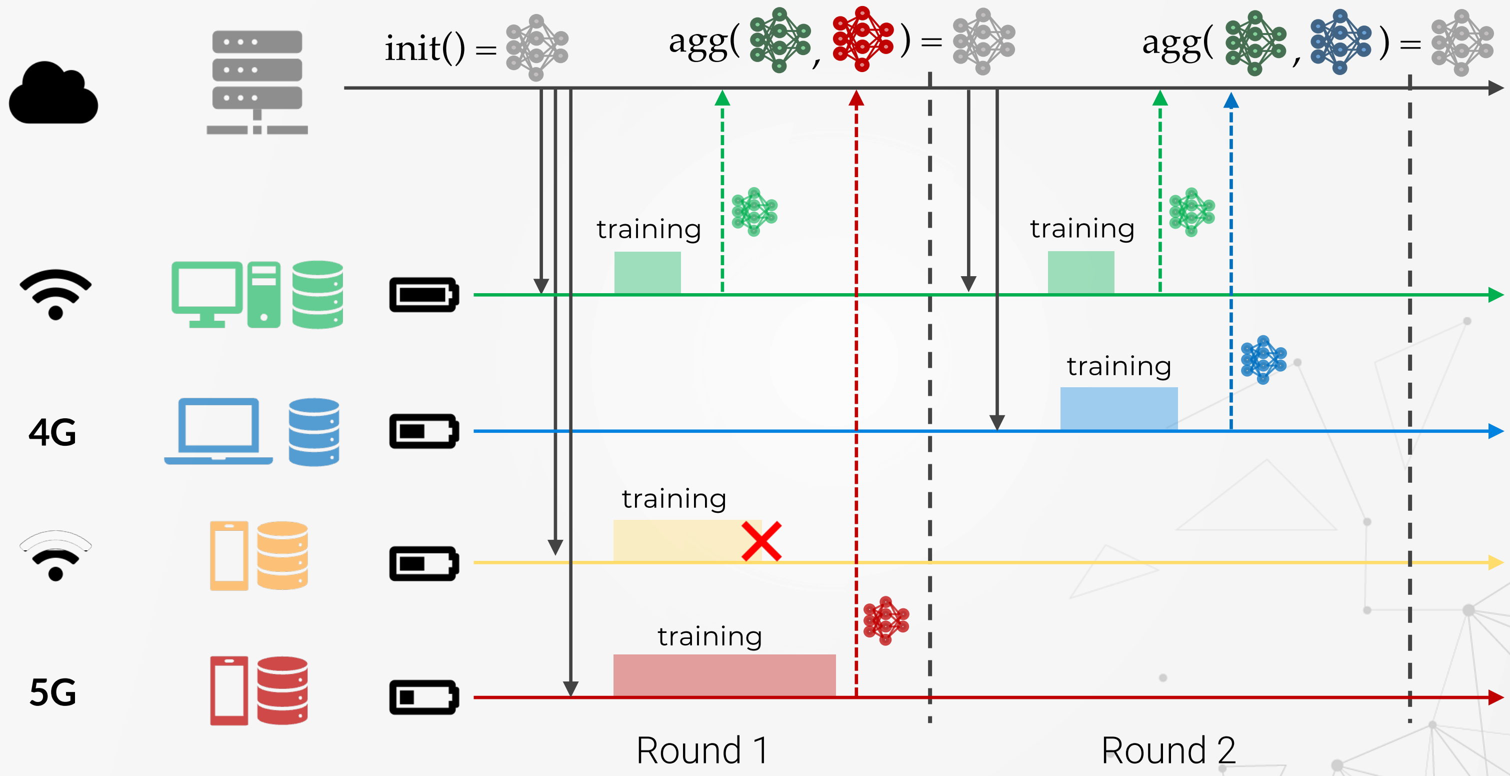
ROUND 2

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



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
- $\boldsymbol{w}$: from now on, this indicates the parameters of the model (thus the model itself)
- f : from now on, the loss function

Federated SGD: FedSGD

- Observation: 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 (FedSGD): **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 $\boldsymbol{w}$
- In a round $\mathrm{t} < T$:
    - The aggregator broadcasts the current ***global*** model $\boldsymbol{w}$ to each (eligible) 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

- Improving computation efficiency:
  - Selects more client in each round: more reliable gradient estimate
  - Increase the computation on each client

FedAvg

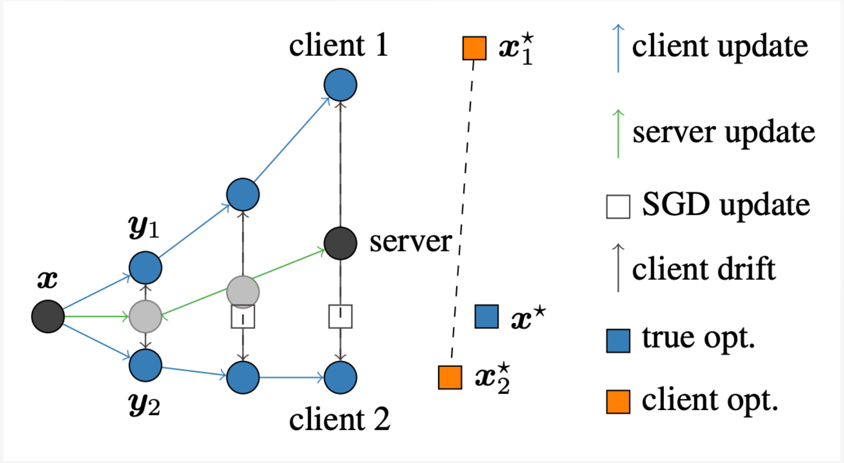
- We assume the aggregator initialized the global model $\boldsymbol{w}$
- In a round $\mathrm{t} < T$:
    - The aggregator broadcasts the current ***global*** model $\boldsymbol{w}$ to each (eligible) client;
    - Each client k computes gradient on its local data
    - *Like alternative 2 of FedSGD:*
        - 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, FedAvg does not guarantees linear convergence for smooth, strongly convex losses
- Intuition

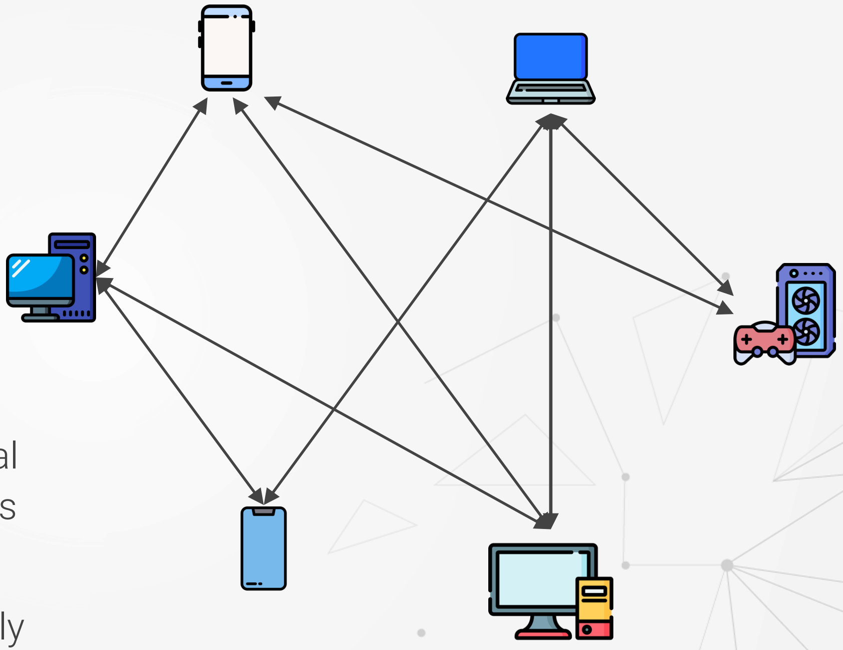


In the figure the model parameters are called $\boldsymbol{x}$ and $\boldsymbol{y}$

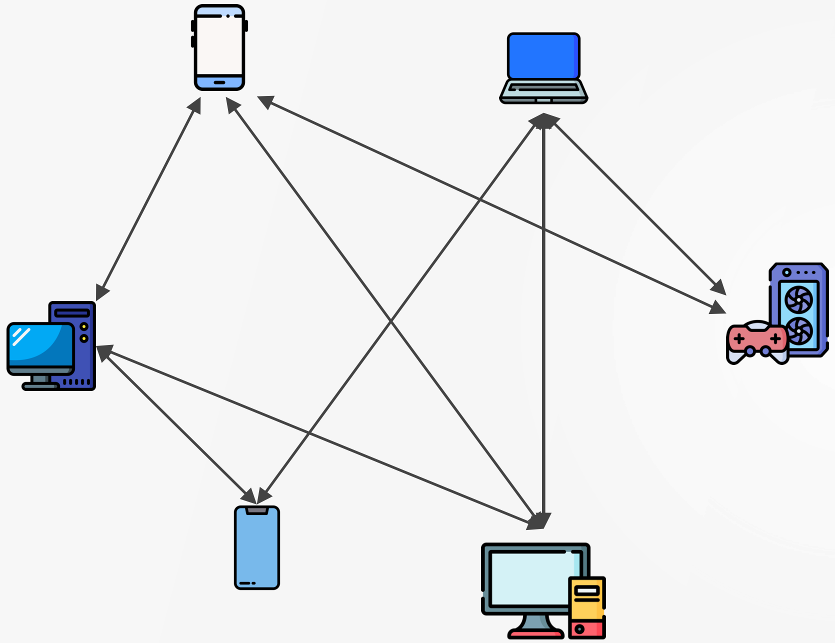
Decentralized Federated Learning (DFL)

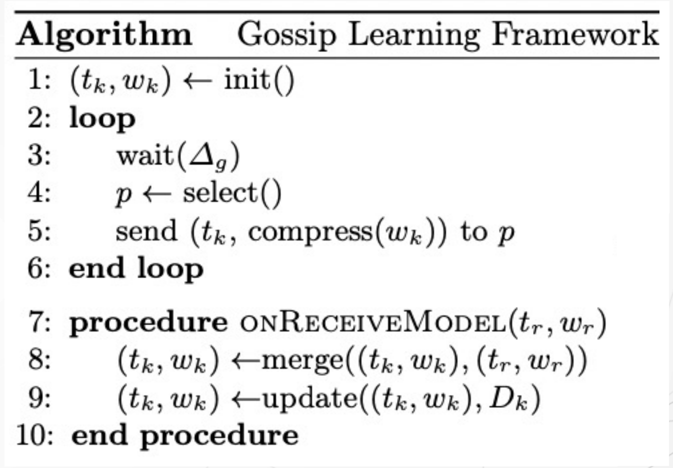
- No central server/aggregator
  - No single point of failure
  - No trust in an orchestrator
  - The learning must happen in a peer-to-peer fashion

- Two main approaches:
  - Broadcast: each client broadcasts its local model/gradients to every reachable nodes
  - Gossip: each client sends its local model/gradients to a small subset (usually one) node in the network



Gossip Learning





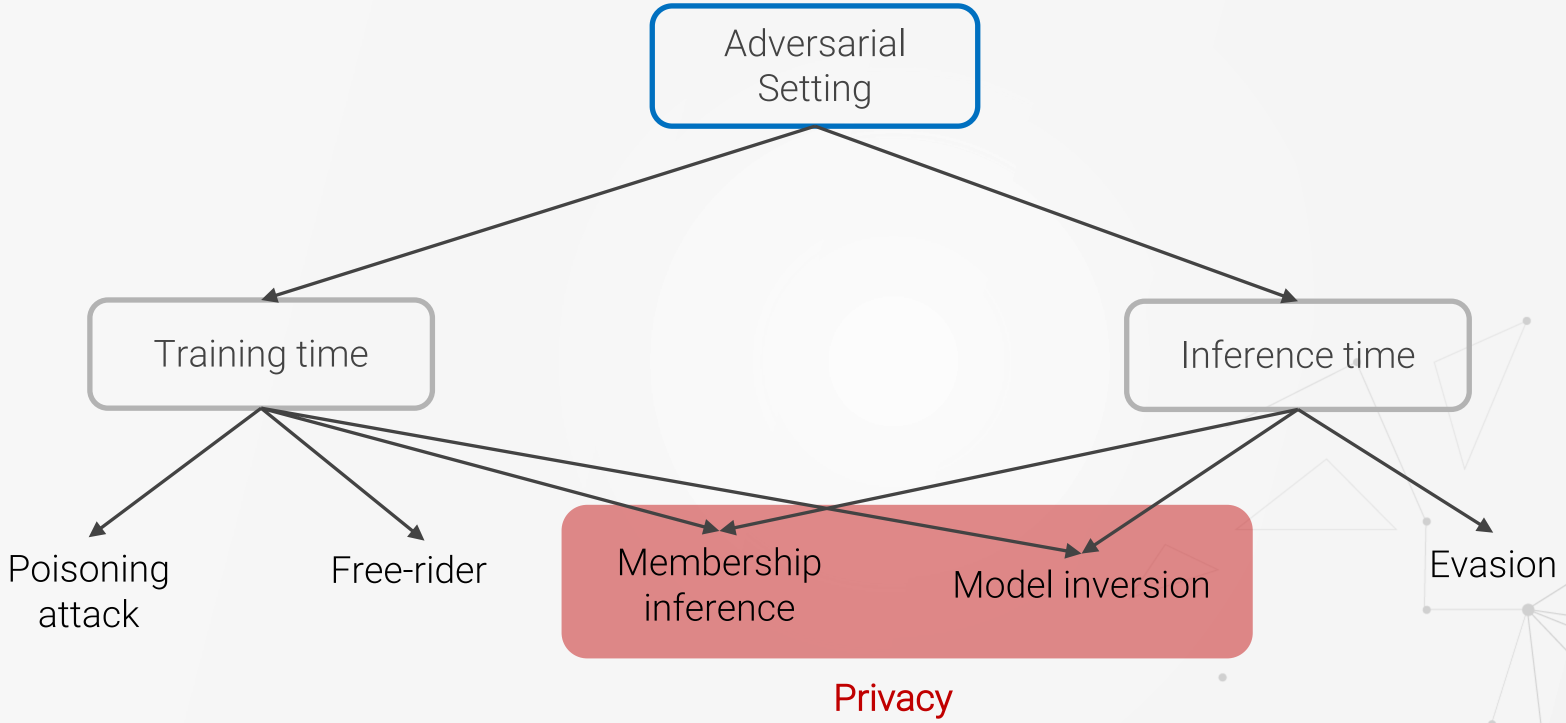
```
Algorithm  Gossip Learning Framework
 1: $(t_k, w_k) \leftarrow \text{init}()$
 2: loop
 3:     wait$(\Delta_g)$
 4:     $p \leftarrow \text{select}()$
 5:     send $(t_k$, compress$(w_k))$ to $p$
 6: end loop
 7: procedure ONRECEIVEMODEL$(t_r, w_r)$
 8:     $(t_k, w_k) \leftarrow \text{merge}((t_k, w_k), (t_r, w_r))$
 9:     $(t_k, w_k) \leftarrow \text{update}((t_k, w_k), D_k)$
10: end procedure
```

03

Security & Privacy in Federated Learning

FL attacks' taxonomy

Gorka, Picek, Ramírez-Durán and Urbieta. On the Security & Privacy in Federated Learning. https://arxiv.org/pdf/2112.05423.pdf



Is FL really privacy-preserving?

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients - how easy is it to break privacy in federated learning? In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20).

- 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!

IMPORTANT

***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.*

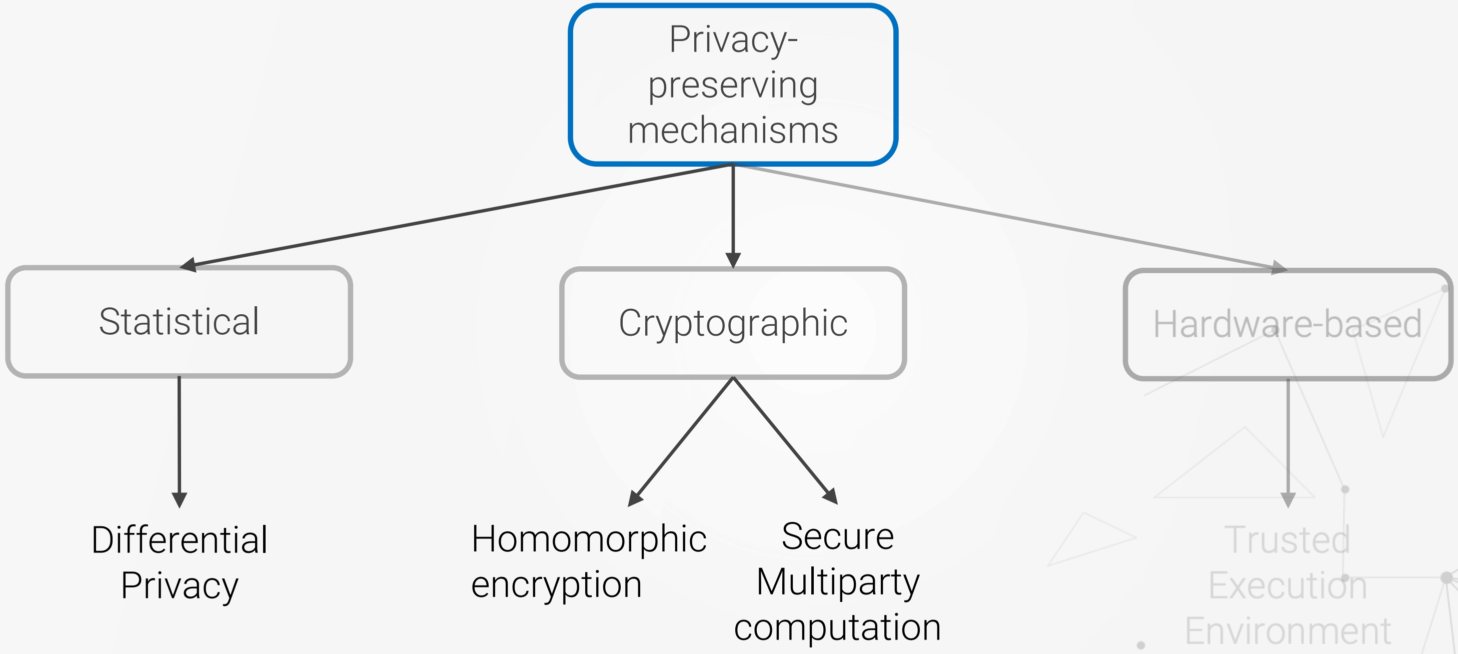
Type of attackers in FL

- 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.

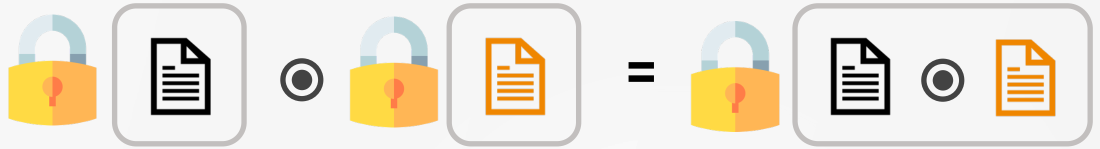
Privacy threats in FL

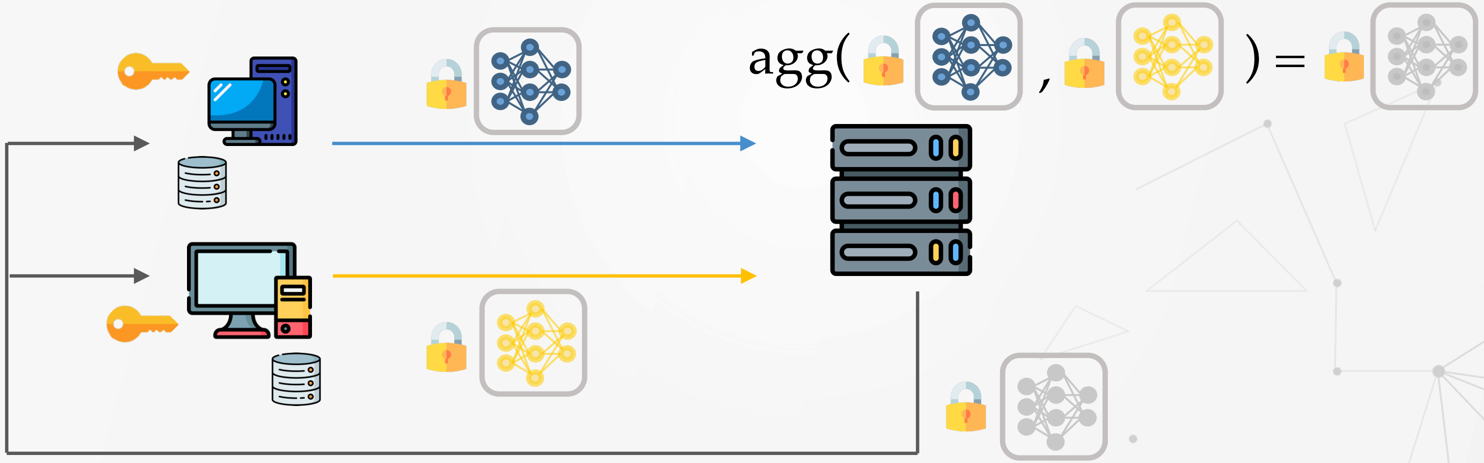
- The main attacks to users' privacy are:
  - **Membership inference:** the goal is to infer whether some (given) data belongs to the training dataset. Under FL, it is even possible to suggest which user owns the dataset. The attacker aim is to infer if some data piece $\{(\mathbf{x}, \mathbf{y})\}$ belongs to a local dataset
  - **Model inversion:** a machine learning model (e.g., GAN) is trained to infer the class conditional distribution $p(\mathbf{x} | \mathbf{y})$. In other words, the attacker tries to infer the training data.

Standard defenses against privacy attacks


Privacy-preserving mechanisms
Statistical
Cryptographic
Hardware-based
Differential Privacy
Homomorphic encryption
Secure Multiparty computation
Trusted Execution Environment

Homomorphic Encryption





Differential Privacy

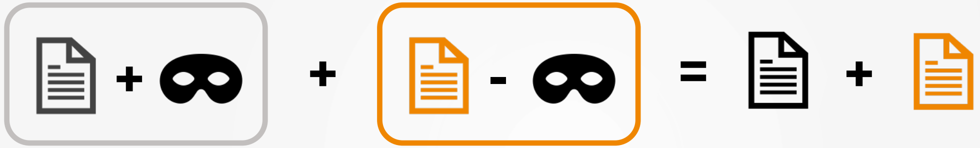
Differential Privacy (DP) is a privacy-preserving mechanism that adds noise to the model for limiting a wide range of attacks. For example, an Inference attack, such as Model Inversion, will contain noise, degrading attack efficiency.

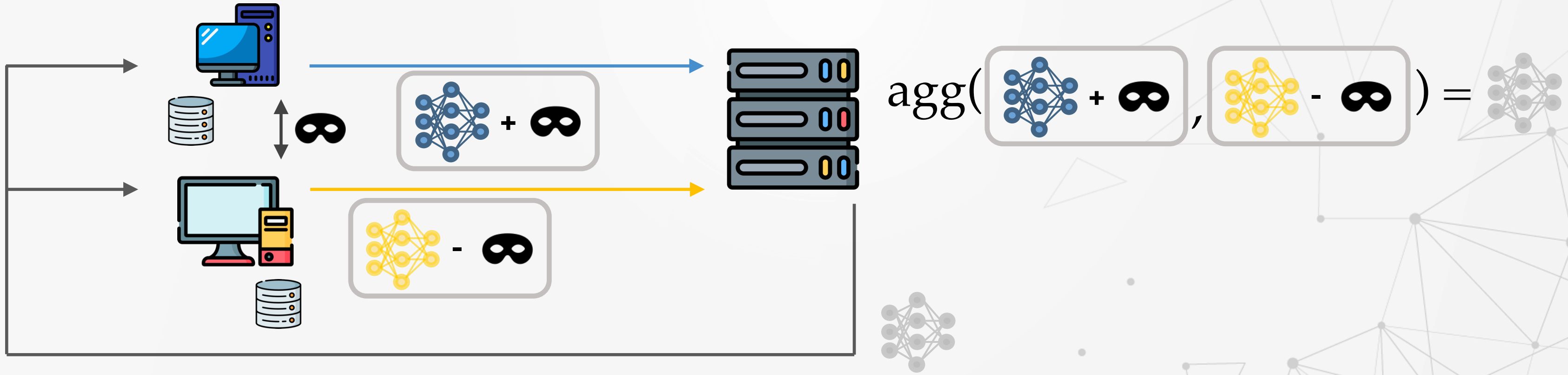
- **LOCAL Differential Privacy**: the noise to the model is added by the clients before sending the model to the server
    - This mechanism protects the privacy even in case of a malicious server, i.e., it sees only noisy models
    - All the model updates are noisy affecting the overall training process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

K. Bonawitz, et al. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). https://doi.org/10.1145/3133956.3133982

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 $\boldsymbol{s}_{u,v}$ uniformly from $[0, R)^k$
2. For each other user v.
   - Users u and v exchange $\boldsymbol{s}_{u,v}$ and $\boldsymbol{s}_{v,u}$ 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: $y_u = x_u + \sum_{v \in \mathcal{U}} p_{u,v} (\bmod\ R)$
4. The server simply sums the perturbed values: $\bar{x} = \sum_{u \in \mathcal{U}} y_u (\bmod\ R)$

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

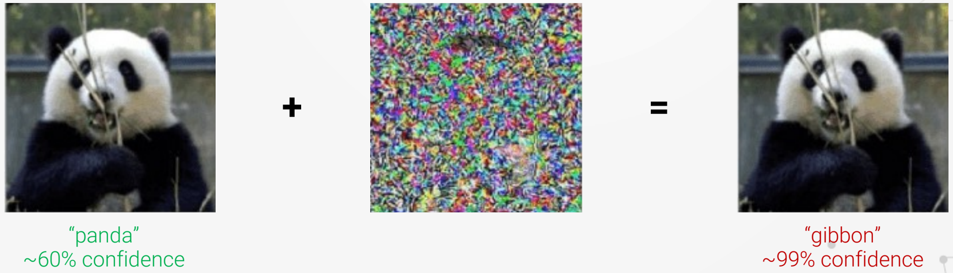
$$\bar{x} = \sum_{u \in \mathcal{U}} x_u + \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} p_{u,v} = \sum_{u \in \mathcal{U}} x_u + \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} s_{u,v} - \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{U}} s_{v,u} = \sum_{u \in \mathcal{U}} x_u \pmod R$$

Security threats in Federated Learning

- **Poisoning attack**: the attacker modifies the client(s) data or the model update(s) for malicious purposes.
    - **Untargeted**: the attacker aims to reduce the overall performance of the global model
    - **Targeted**: the attacker aims to modify the predictions of the model in a specific way

- **Free-rider**: it is a client that does not contribute to the learning but simply take advantage of the learned model and may send dummy updates. Especially critical in cross-silo setting

- **Evasion attack**: it is an attack at inference time and the goal is to trick the model by manipulating the input data. The crafted input is usually manipulated by specific types of noise usually hard to be detected



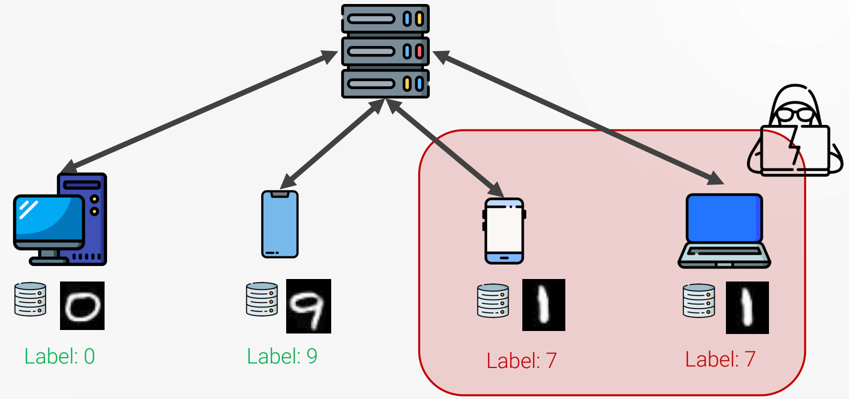
Data Poisoning

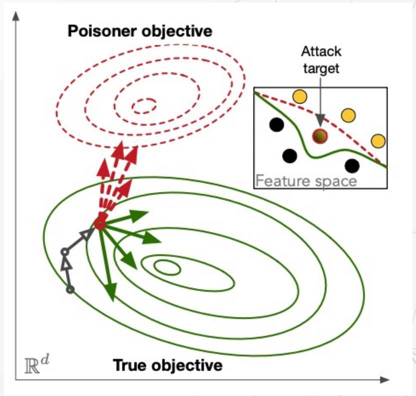
- Most common in ML but **not very effective in FL** because updates from benign clients tend to reduce the negative effect of the malicious dataset during the aggregation process

- Data Poisoning attacks rely on dataset modification during local training to degrade the overall model performance achieved by **label-flipping**
    - **Clean-label** attacks do not require control over the labeling function. The poisoned dataset seems correct to the eyes of an expert label-verifier identity
    - **Dirty-label** consists of modifying the dataset by changing labels to the desired ones (targeted) or to a random ones (untargeted), e.g., changing all cats labels into birds

Model Poisoning

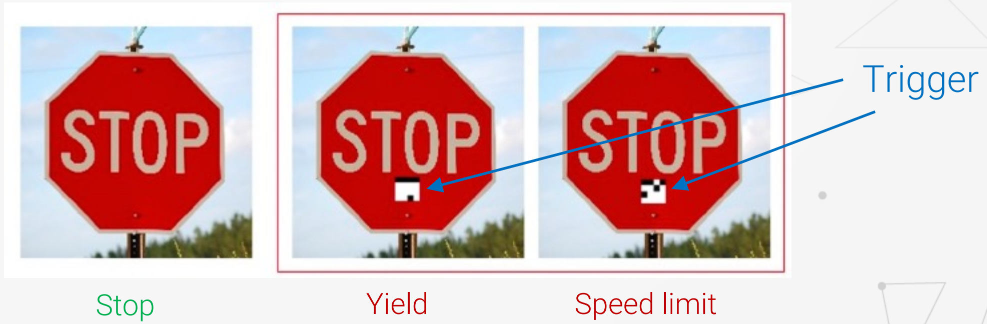
- Malicious users deliberately **change the model parameters** to achieve a malicious goal (**white-box attack**)
- **Boosting** is a technique that enlarges clients' contributions by multiplying the vector weights (the update) by a scaling factor. **Inverse gradient** and **scaling gradient** attack are two examples of boosting.
- **Sybil attacks** are coordinated attacks among various adversarial clients. A single attacker could control more than a single client and usually the intent is to disrupt the learning process





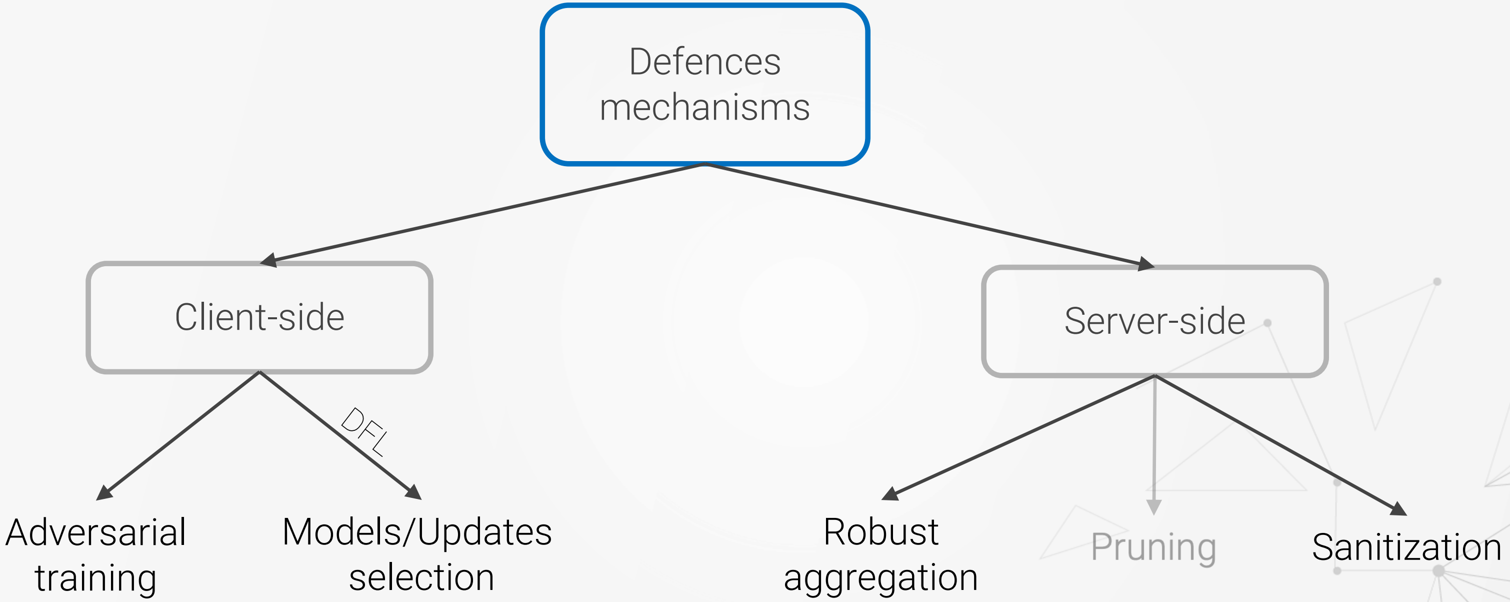
Backdoor attack

- **Backdoor attacks** (aka Trojan) are a particular type of data poisoning where the poisoned data (usually label-flipped) belongs to a specific category or have specific features
  - **Edge backdoor** poisons rare data pieces (i.e., edge cases). Gradient-based defense techniques are unlikely to detect this attack
  - **Semantic backdoor** modifies the label of inputs with specific features, e.g., all pink cars are poisoned. Boosting is usually used along with this type of attack
  - **Pixel-pattern backdoor**, primarily used on image recognition, it **injects a pattern** in a particular area of the input space (which act as a trigger for the classifier) and flips the label



Defences for security vulnerabilities in FL

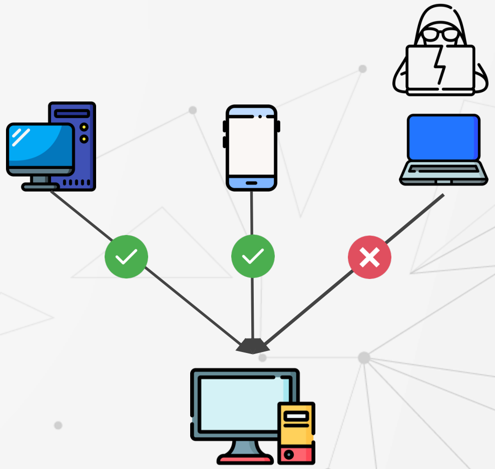
Mothukuri, Parizi, Pouriyeh, Huang, Dehghantanha, Srivastava, A survey on security and privacy of federated learning. Future Generation Computer Systems, Volume 115, 2021, Pages 619-640.



Client-side defences

- **Adversarial Training**: the client augment its own local dataset with adversarial examples in order to make the learned model robust to data poisoning attacks and to evasion attacks

- **Models/updates selection**: in case of distributed federated learning, e.g., gossip learning, clients can cache the models/updates from the neighbour nodes and then it can validate them on its local datasets selecting only the best performing ones



Server-side defence: sanitization

In FL, the server/aggregator cannot perform data-filtering (clearning) however it can **filter out malicious updates**. This is called **sanitization** or **server cleaning**.

- **Accuracy-based**: models/updates are evaluated on an held-out test set (on the server) and discarded or down-weighted according to the achieved accuracy. It is not always possible to have an held-out test set on the server.
- **Similarity-based**: the local models/updates are compared in terms of similarity (e.g., clustering, dimensionality reduction) and the most dissimilar ones are discarded before the aggregation. May not be the best solution in case of free-riders.
- **Blockchain-based**: ensure data integrity and traceability via Proof-of-Work. Computationally demanding.

Server-side defence: robust aggregation

The server/aggregator uses a **modified aggregation function** to be **robust** against adversarial updates/models.

- **Krum**: for each client's update, the aggregator calculates the sum of the Euclidean distance towards the other received models. Then, the aggregator selects the model with a lower distance. **Effective in networks with less than 50% of malicious users**. Unfortunately, the Krum algorithm seems to **fail on non-IID settings**.

- **FoolsGold**: desigend for Sybil attacks, they are based on the observation that **sybils** work together towards a common adversarial goal thus their **updates should be cosine-similar**. Potential sybils models are down-weighted during aggregation.

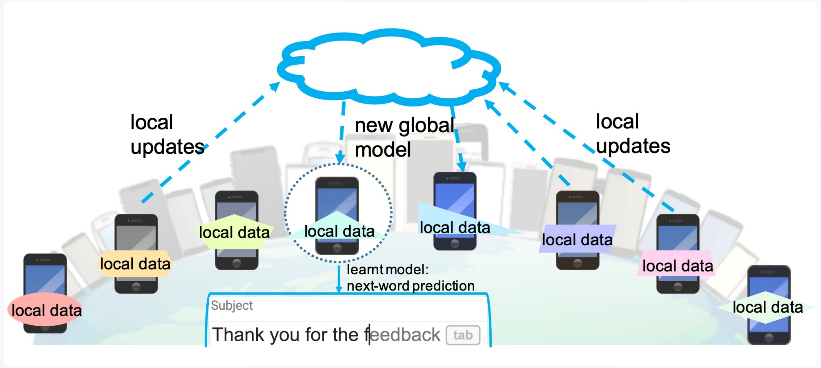
- **Trimmed mean**: for every dimension of the parameter space the median among the client updates is computed and only values close to the median are used in subsequent aggregation.

04

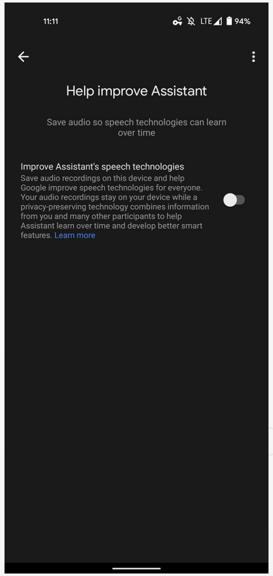
Wrapping up!

Is Federated Learning used, yet?

Google's GBoard



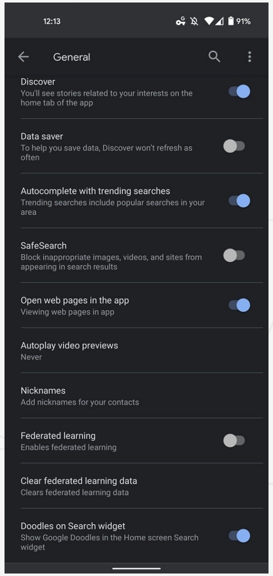
Google's "Hey Google!" recognition



Help improve Assistant

Save audio so speech technologies can learn over time

Improve Assistant's speech technologies
Save audio recordings on this device and help Google improve speech technologies for everyone. Your audio recordings stay on your device while a privacy-preserving technology combines information from you and many other participants to help Assistant learn over time and develop better smart features. Learn more



Discover
You'll see stories related to your interests on the home tab of the app

Data saver
To help you save data, Discover won't refresh as often

Autocomplete with trending searches
Trending searches include popular searches in your area

SafeSearch
Block inappropriate images, videos, and sites from appearing in search results

Open web pages in the app
Viewing web pages in app

Autoplay video previews
Never

Nicknames
Add nicknames for your contacts

Federated learning
Enables federated learning

Clear federated learning data
Clears federated learning data

Doodles on Search widget
Show Google Doodles in the Home screen Search widget

Current research interests

- Development of federated learning algorithms/approaches for gradient-free machine learning models, e.g., SVM, boosting etc…
- Development of a python framework to simulate and experiments with gossip learning https://github.com/makgyver/gossipy
- Study of vulnerabilities of decentralized federated learning algorithms
- Efficient algorithms for decentralized federated learning
- Federated recommender systems

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
- **FL inherits all the vulnerabilities of standard machine learning models**

We are “hiring” ☺

Possible thesis topics:

- Study of the privacy and security aspects in decentralized FL setting such as gossip learning
- Study of the effect of Differential Privacy techniques on different model architectures & tasks
- Study of novel defenses against attacks to Federated Learning (both centralized and decentralized)