JC4001: Distributed Systems

Federated Learning

Xiaonan Liu
xiaonan.liu@abdn.ac.uk

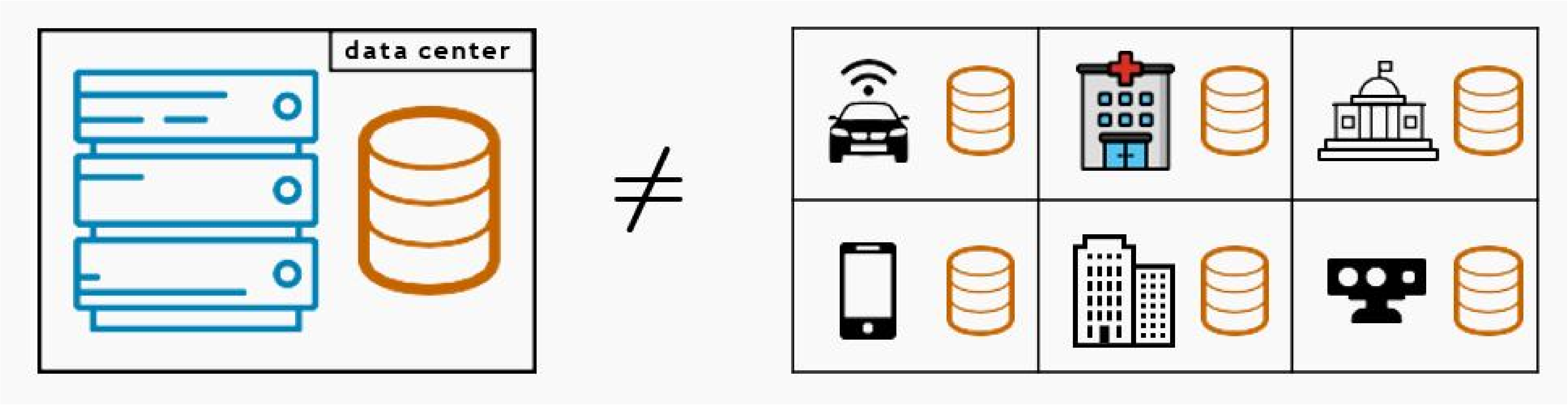
1495 UNIVERSITY OF ABERDEEN

Outline

- What is Federated Learning
- A Baseline Algorithm: Federated Averaging
- Some Challenges in Federated Learning

What is Federated Learning

- The standard setting in Machine Learning (ML) considers a centralized dataset processed in a tightly integrated system
- But in the real world data is often decentralized across many parties

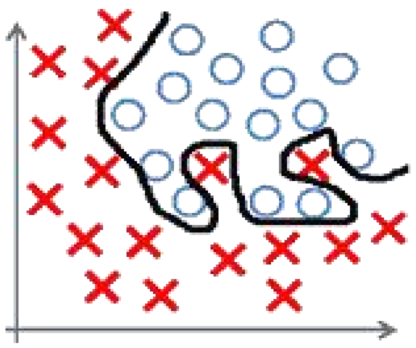


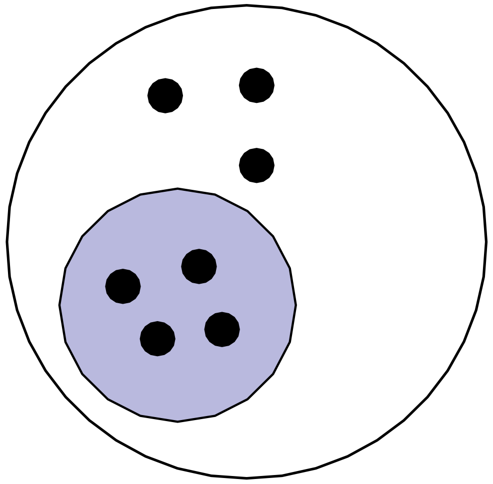
What is Federated Learning

- Sending the data may be too costly
  - Self-driving cars are expected to generate several TBs of data a day
  - Some wireless devices have limited bandwidth/power
- Data may be considered too sensitive
  - We see a growing public awareness and regulations on data privacy
  - Keeping control of data can give a competitive advantage in business and research

What is Federated Learning

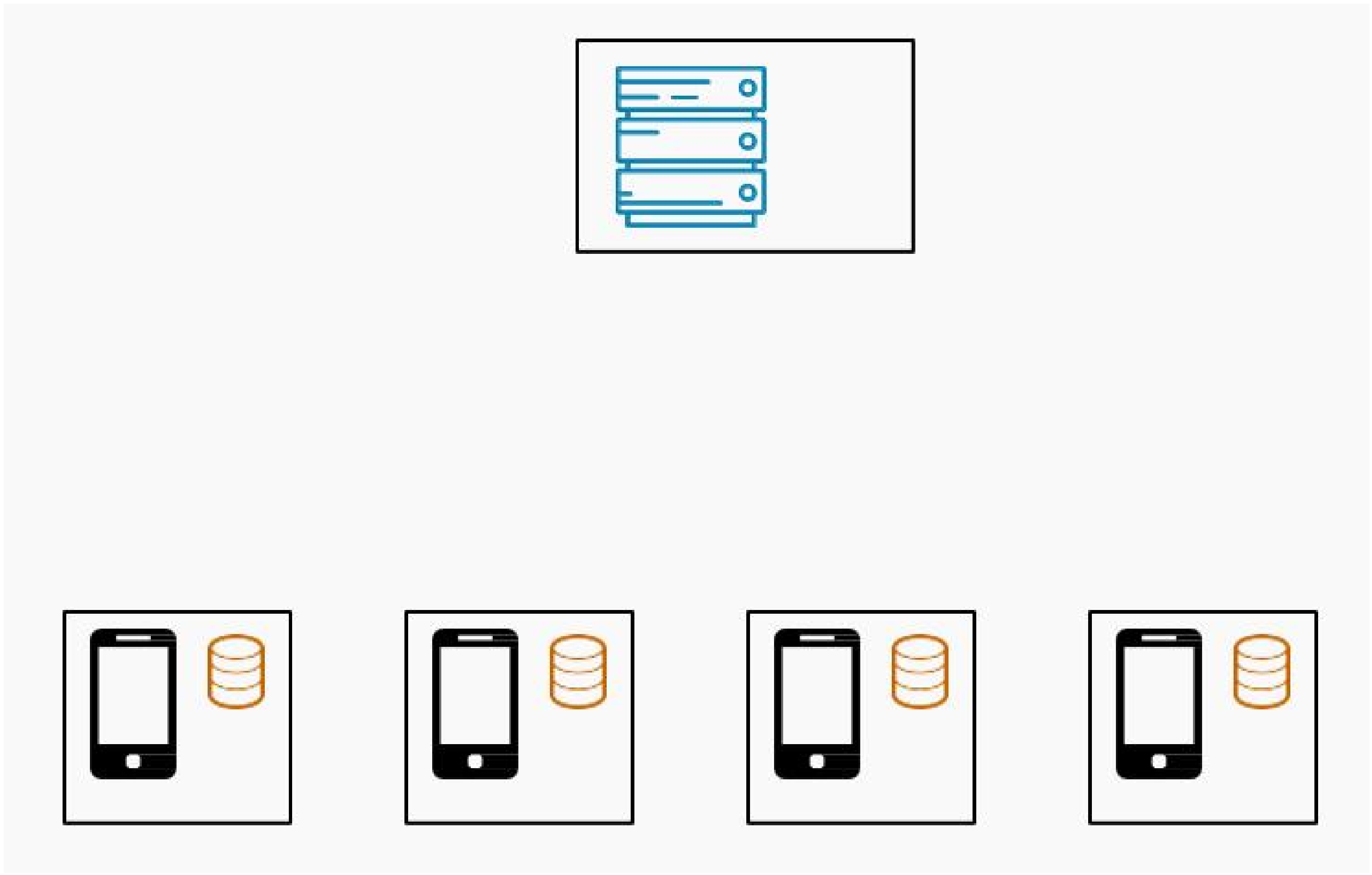
- The local dataset may be too small
  - Sub-par predictive performance (e.g., due to overfitting)
  - Non-statistically significant results (e.g., medical studies)
- The local dataset may be biased
  - Not representative of the target distribution





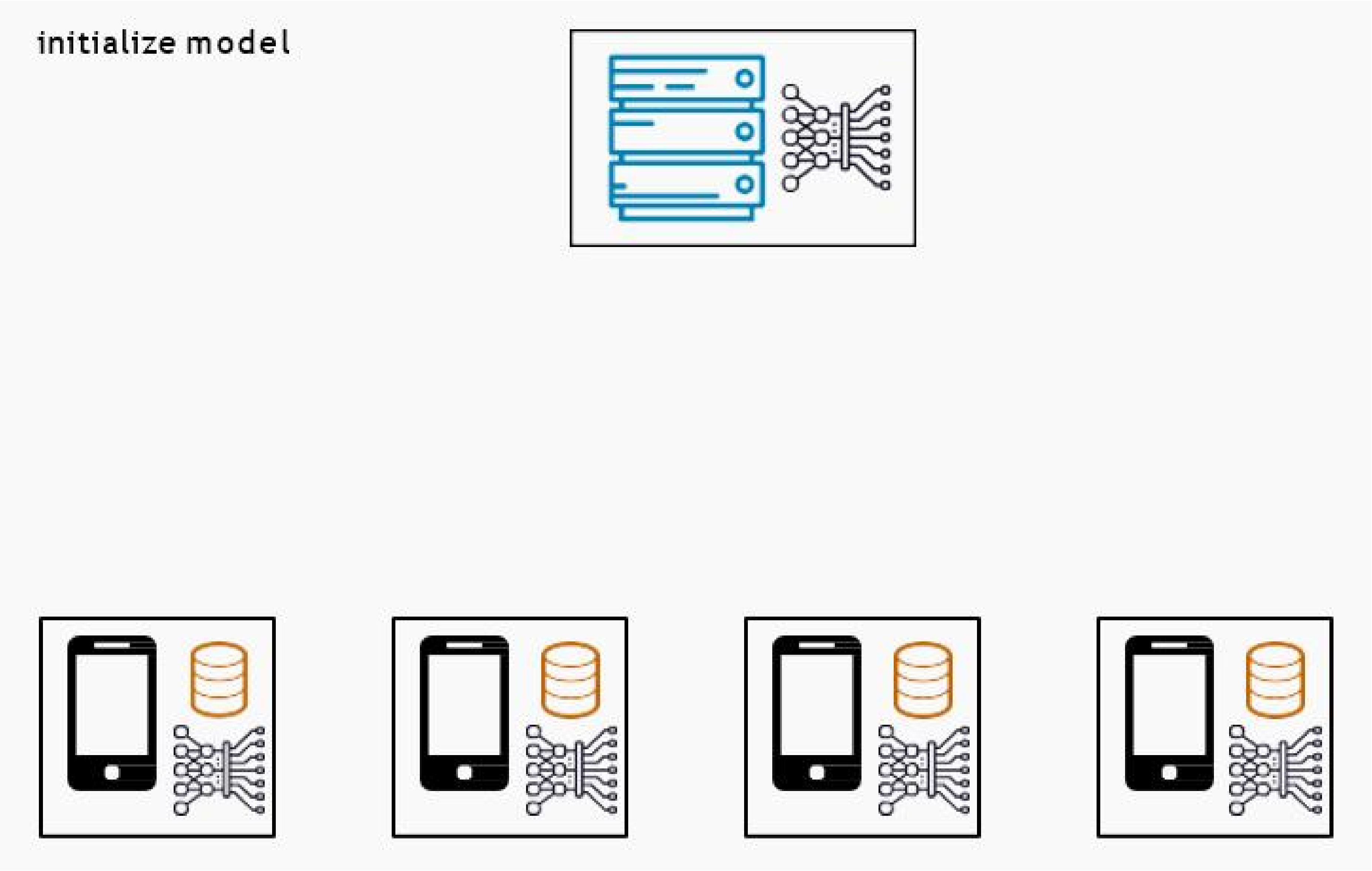
What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized



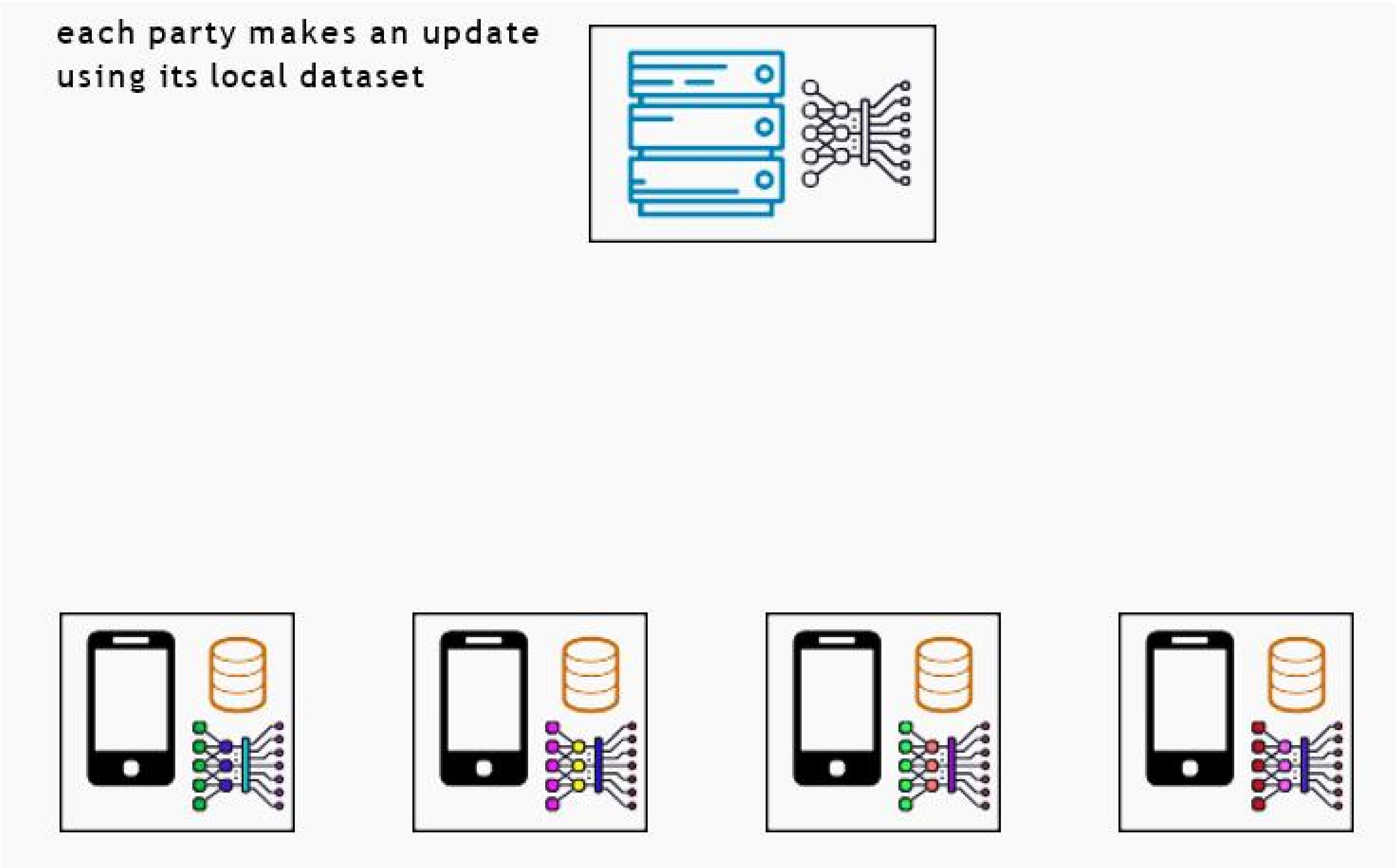
What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized



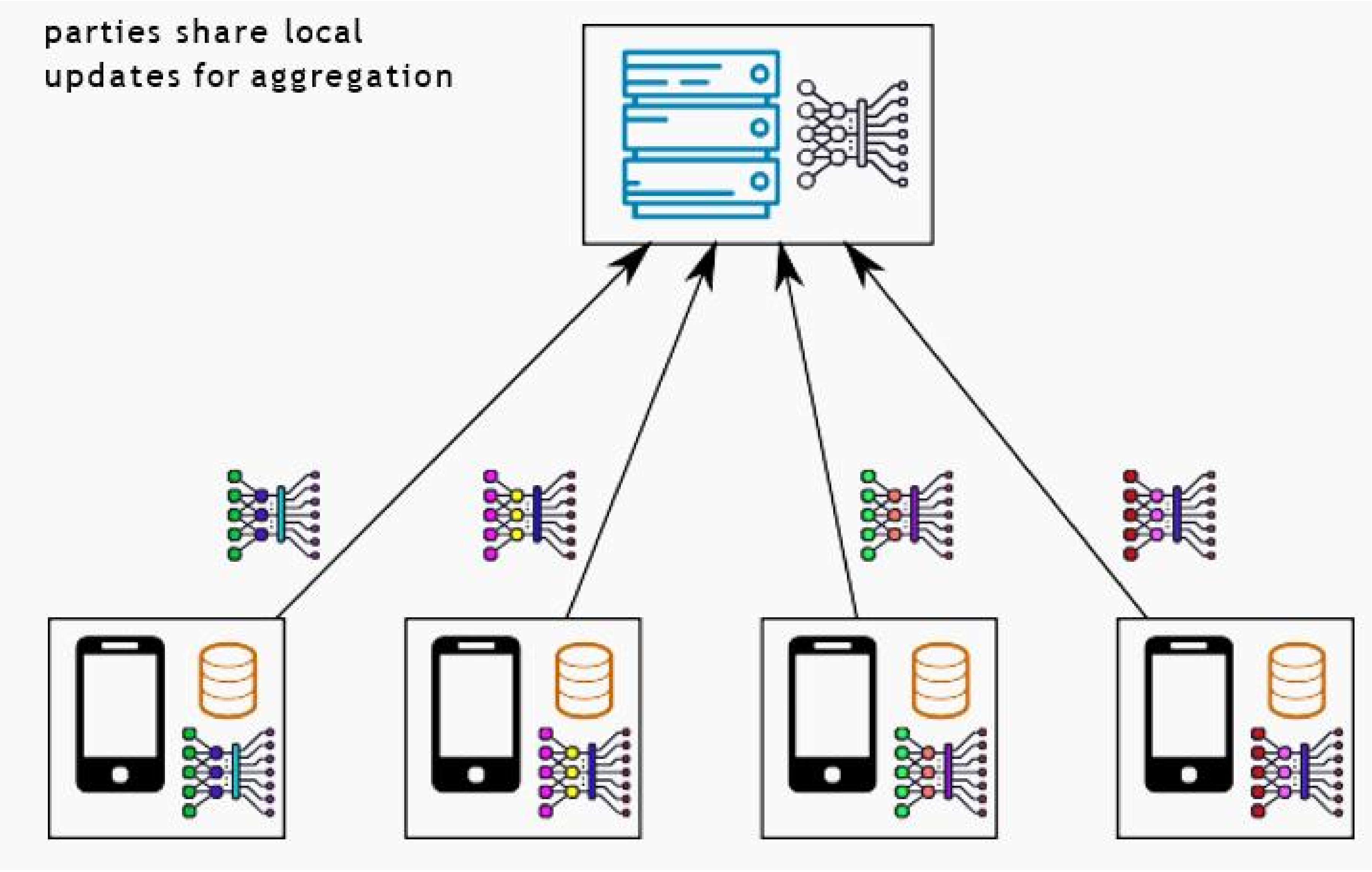
What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized



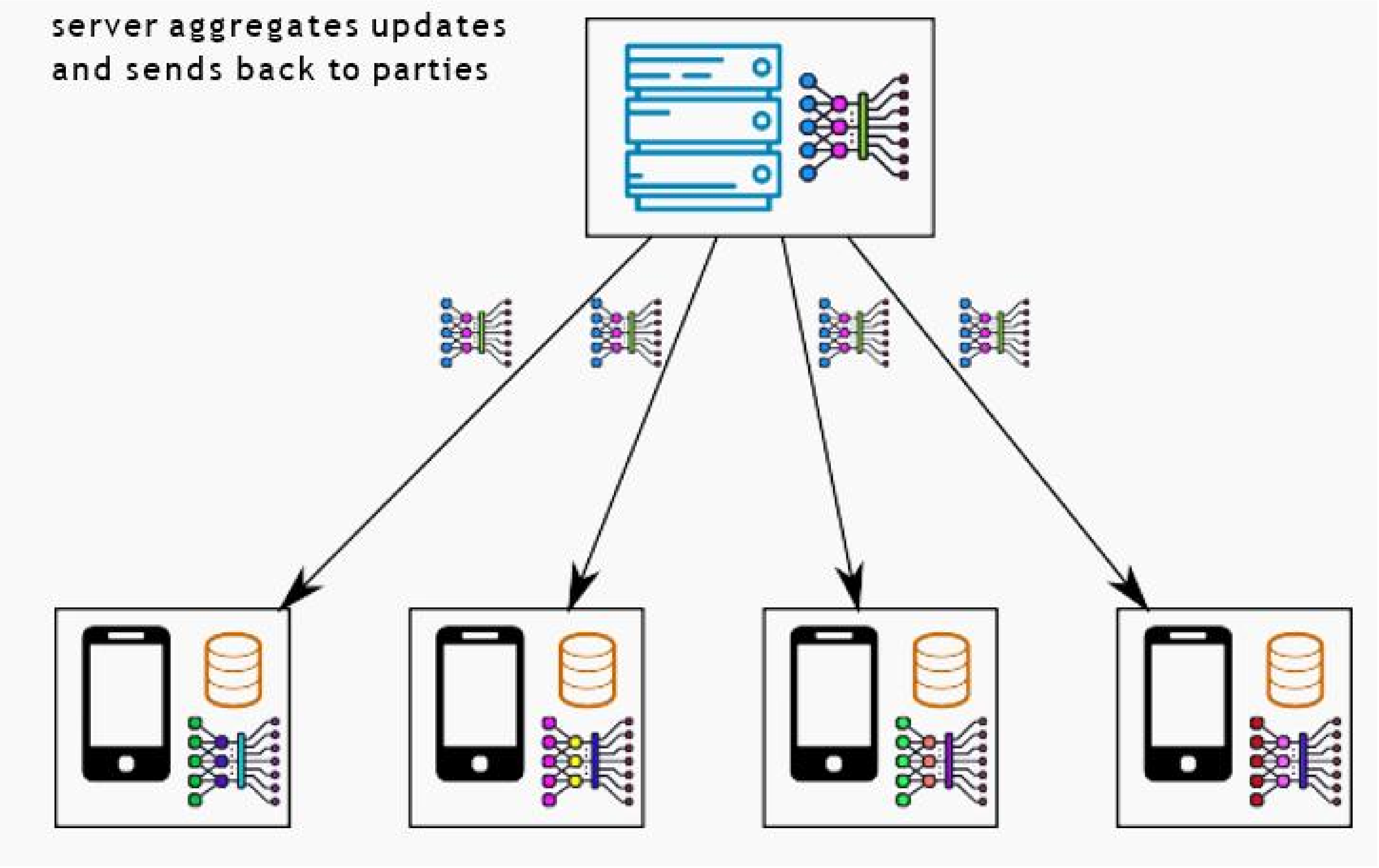
What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized



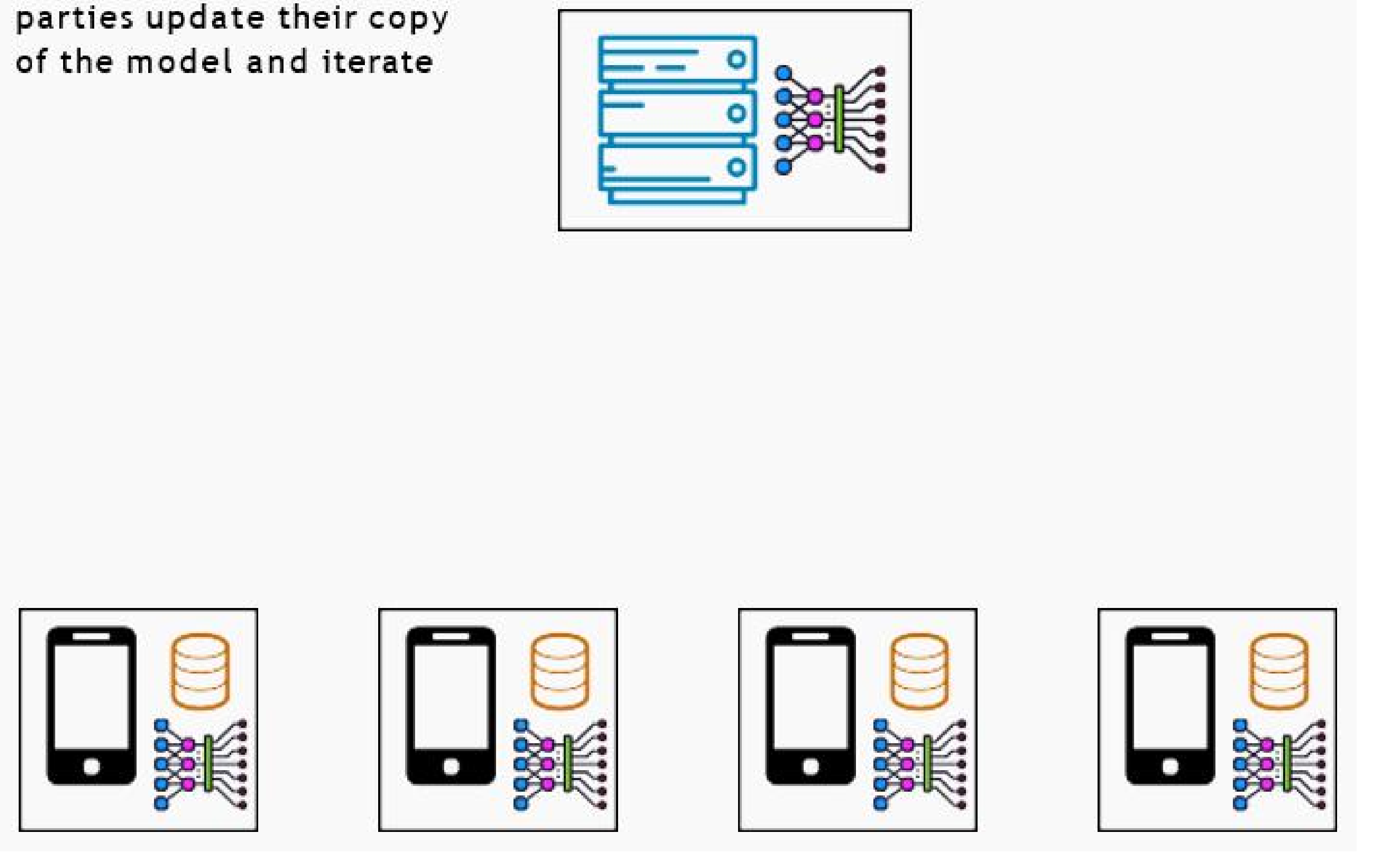
What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized



What is Federated Learning

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized

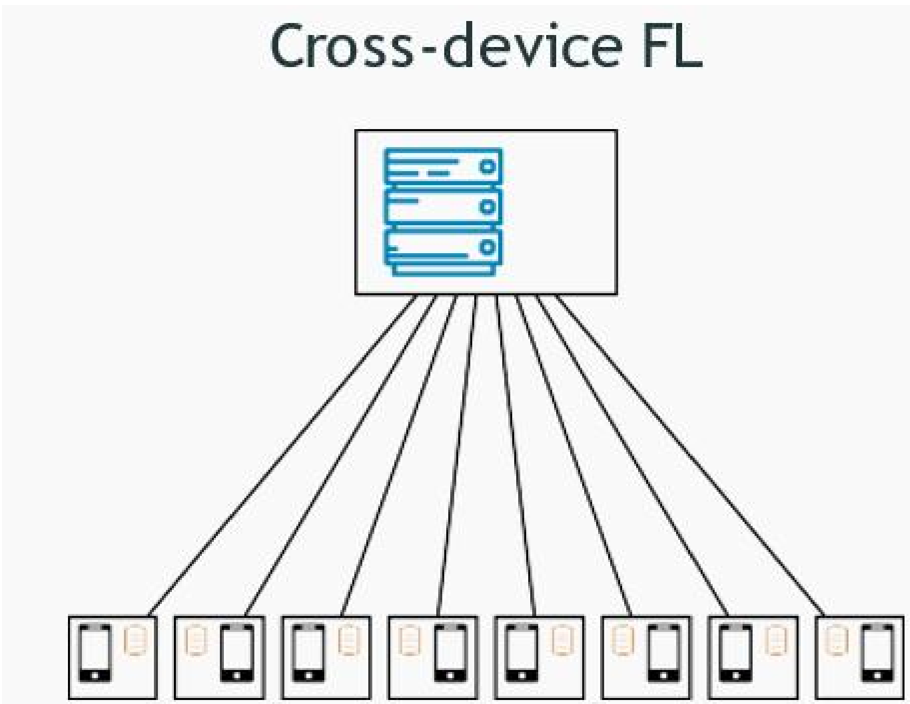


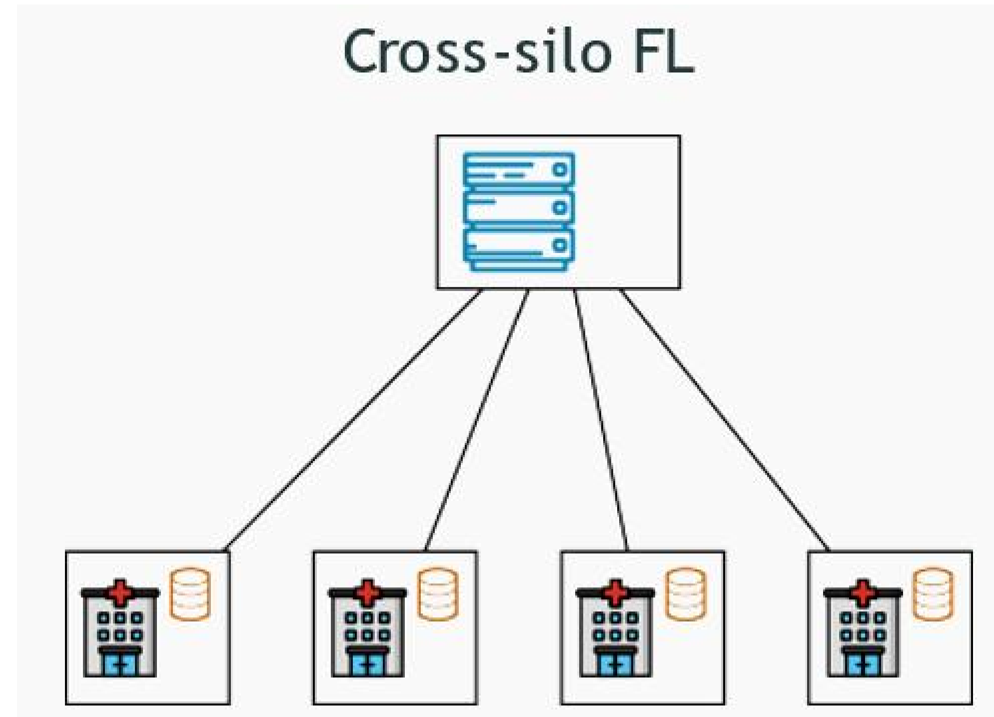
- We would like the final model to be as good as the centralized solution (ideally), or at least better than what each party can learn on its own

Differences with Distributed Learning

- In distributed learning, data is centrally stored (e.g., in a data center)
  - The main goal is just to train faster
  - We control how data is distributed across workers: usually, it is distributed uniformly at random across workers
- In FL, data is naturally distributed and generated locally
  - Data is not independent and identically distributed (non-i.i.d.), and it is imbalanced
- Additional challenges that arise in FL
  - Enforcing privacy constraints
  - Dealing with the possibly limited reliability/availability of participants
  - Achieving robustness against malicious parties

Cross-Device and Cross-Silo FL

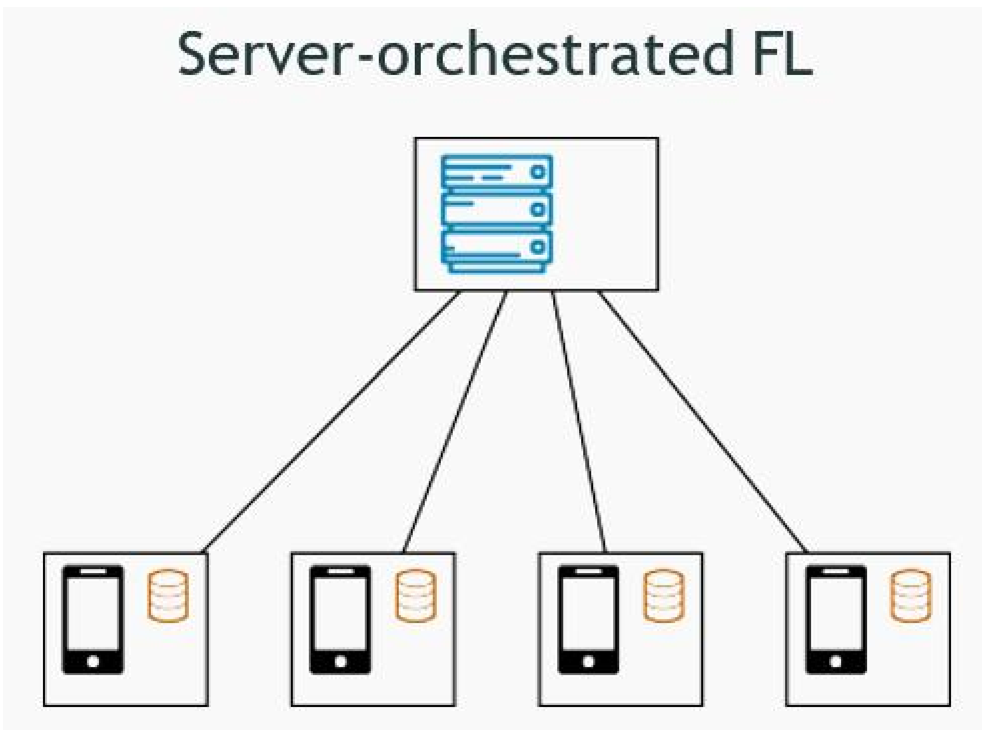


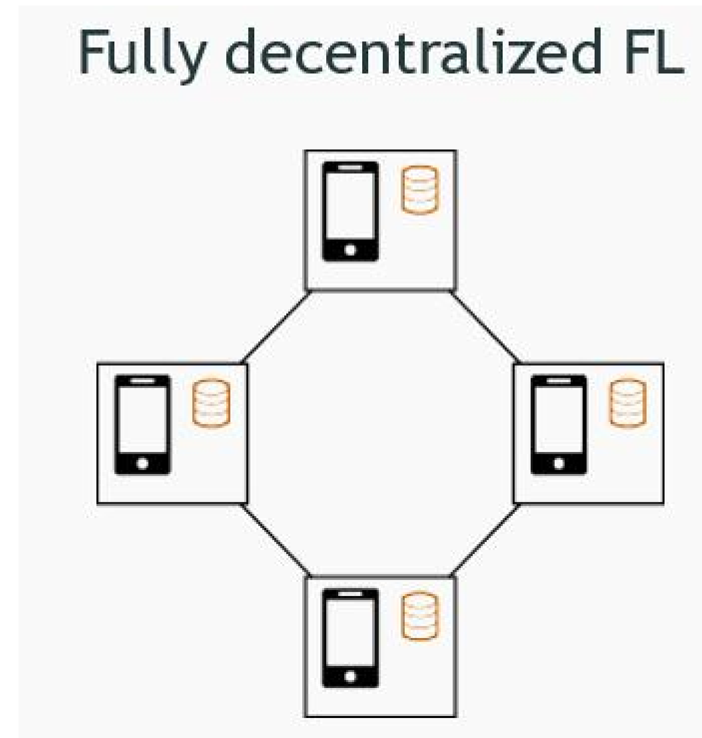


- Massive number of parties (up to 10^10)
- Small dataset per party (could be size 1)
- Limited availability and reliability
- Some parties may be malicious

- 2-100 parties
- Medium to large dataset per party
- Reliable parties, almost always available
- Parties are typically honest

Server Orchestrated and Fully Distributed FL





- Server-client communication
- Global coordination, global aggregation
- Server is a single point of failure and may become a bottleneck

- Device-to-device communication
- No global coordination, local aggregation
- Naturally scales to a large number of devices

FL is a Booming Topic

- 2016: the term FL is first coined by Google researchers; 2020: more than 1,000 papers on FL in the first half of the year
- We have already seen some real-world deployments by companies and researchers
- Several open-source libraries are under development: PySyft, TensorFlow Federated, FATE, Flower, Substra...
- FL is highly multidisciplinary: it involves machine learning, numerical optimization, privacy & security, networks, systems, hardware...

A Baseline Algorithm: Federated Averaging

- We consider a set of K parties (clients)
- Each party k holds a dataset D_k of n_k points
- Let D = D1 ∪ · · · ∪ DK be the joint dataset and $n = \sum_k n_k$ the total number of points
- We want to solve problems of the form $\min_{\theta \in R^p} F(\theta; D)$ where:

$$F(\theta; \mathcal{D}) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta; \mathcal{D}_k) \qquad F_k(\theta; \mathcal{D}_k) = \sum_{d \in \mathcal{D}_k} f(\theta; d)$$

- $\theta \in R^p$ are model parameters (e.g., weights of a logistic regression or neural network)
- This covers a broad class of ML problems formulated as empirical risk minimization

FedAvg

Algorithm FedAvg (server-side)

Parameters: client sampling rate ρ

initialize θ
for each round $t = 0, 1, \ldots$ **do**
  $\mathcal{S}_t \leftarrow$ random set of $m = \lceil \rho K \rceil$ clients
  for each client $k \in \mathcal{S}_t$ in parallel **do**
    $\theta_k \leftarrow$ ClientUpdate(k, θ)
  $\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

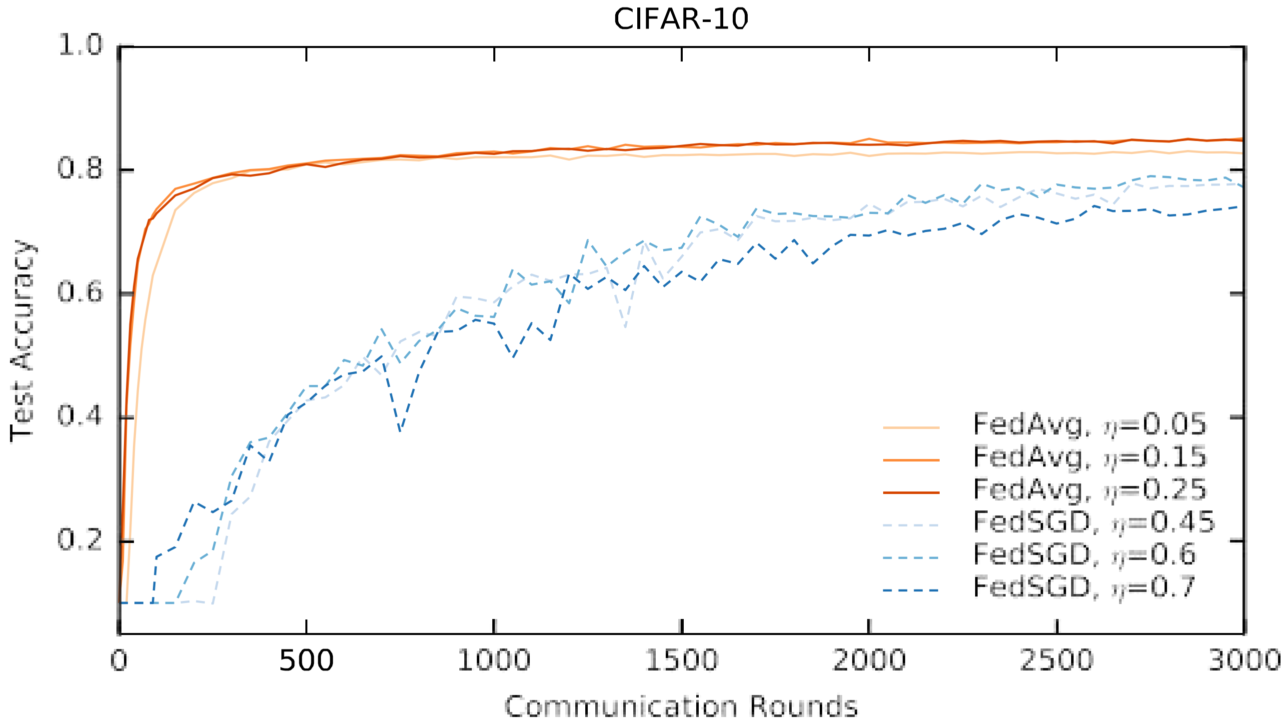
Algorithm ClientUpdate(k, θ)

Parameters: batch size B, number of local steps L, learning rate η

for each local step $1, \ldots, L$ **do**
  $\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$
  $\theta \leftarrow \theta - \frac{n_k}{B} \eta \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$
send θ to server

- For L = 1 and ρ = 1, it is equivalent to classic parallel SGD: updates are aggregated and the model synchronized at each step
- For L > 1: each client performs multiple local SGD steps before communicating

FedAvg



- FedAvg with L > 1 allows to reduce the number of communication rounds, which is often the bottleneck in FL (especially in the cross-device setting)
- It empirically achieves better generalization than parallel SGD with large mini-batch
- Convergence to the optimal model can be guaranteed for i.i.d. data [Stich, 2019] [Woodworth et al., 2020] but issues arise in strongly non-i.i.d. case (more on this later)

Fully Decentralized Learning

- We can derive algorithms similar to FedAvg for the fully decentralized setting, where parties do not rely on a server for aggregating updates
- Let $G = (\{1, . . . , K\}, E)$ be a connected undirected graph where nodes are parties and an edge $\{k, l\} \in E$ indicates that k and l can exchange messages
- Let $W \in [0, 1]^{K \times K}$ be a symmetric, doubly stochastic matrix such that $W_{k,l} = 0$ if and only if $\{k, l\} \notin E$
- Given models $\Theta = [\theta_1, . . . , \theta_K]$ for each party, $W\Theta$ corresponds to a weighted aggregation among neighboring nodes in G:

$$[W\Theta]_k = \sum_{l \in \mathcal{N}_k} W_{k,l}\theta_l, \quad \text{where } \mathcal{N}_k = \{l : \{k, l\} \in E\}$$

Fully Decentralized SGD

Algorithm Fully decentralized SGD (run by party k)

Parameters: batch size B, learning rate η, sequence of matrices $W^{(t)}$

initialize $\theta_k^{(0)}$

for each round $t = 0, 1, \ldots$ **do**

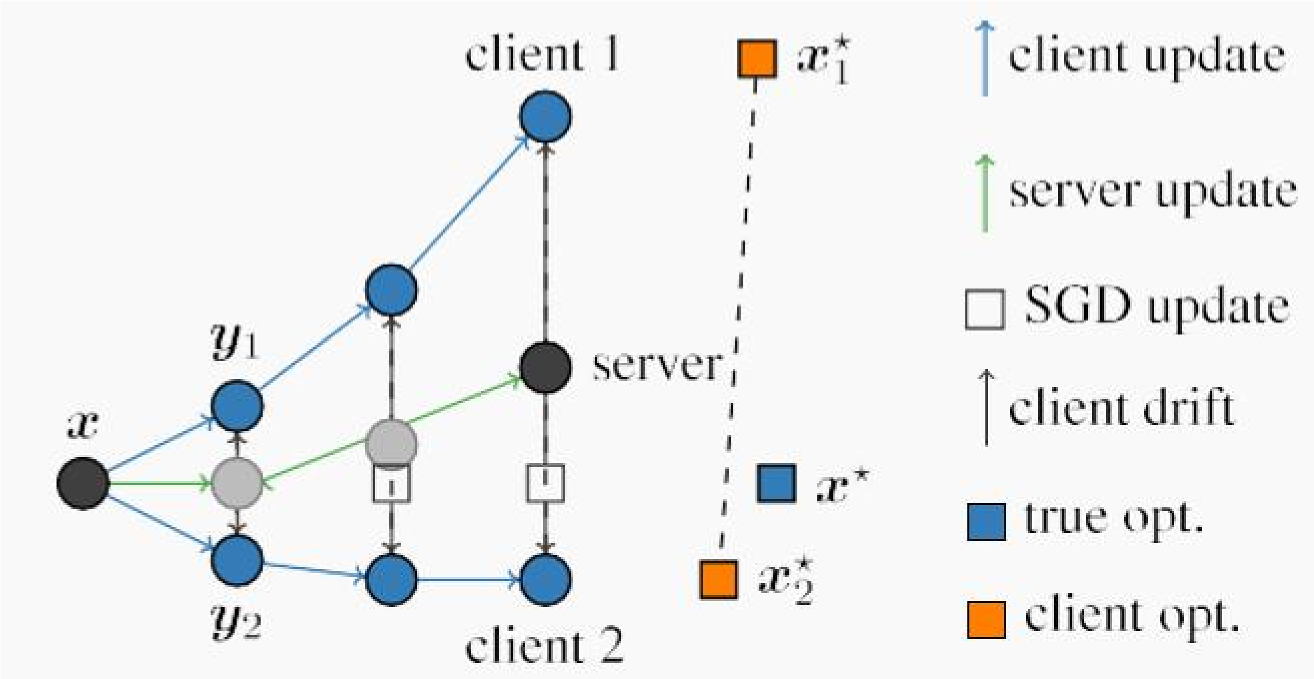
$\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$

$\theta_k^{(t+\frac{1}{2})} \leftarrow \theta_k^{(t)} - \frac{n_k}{B}\eta \sum_{d\in\mathcal{B}} \nabla f(\theta_k^{(t)}; d)$

$\theta_k^{(t+1)} \leftarrow \sum_{l\in\mathcal{N}_k^{(t)}} W_{k,l}^{(t)} \theta_l^{(t+\frac{1}{2})}$

- Decentralized SGD alternates between local updates and local aggregation
- Doing multiple local steps is equivalent to choosing W(t) = In in some of the rounds
- The convergence rate depends on the topology (the more connected, the faster)

Some Challenges in FL



- When local datasets are non-i.i.d., FedAvg suffers from client drift
- To avoid this drift, one must use fewer local updates and/or smaller learning rates, which hurts convergence

FL of Personalized Models

- Learning from non-i.i.d. data is difficult/slow because each party wants the model to go in a particular direction
- If data distributions are very different, learning a single model which performs well for all parties may require a very large number of parameters
- Another direction to deal with non-i.i.d. data is thus to lift the requirement that the learned model should be the same for all parties ("one size fits all")
- Instead, we can allow each party k to learn a (potentially simpler) personalized model θ_k but design the objective so as to enforce some kind of collaboration

Thank You