Federated Learning:
Collaborative ML without centralized training data

Sara Khalifa and Abdelwahed Khamis

May 03, 2023

Outline

- Recap on Machine Learning /Deep Learning
- Centralised Machine Learning
- On device inference /Distributed Learning
- Federated Learning
- Implementing Federated Learning

Recap on ML/DL

- The goal of machine learning/deep learning is to find a model, which produces a desired output given a particular input.

Example task	Given input	Desired output
Image classification		8
Next-word-prediction	*Looking forward to your* ***?***	*reply*

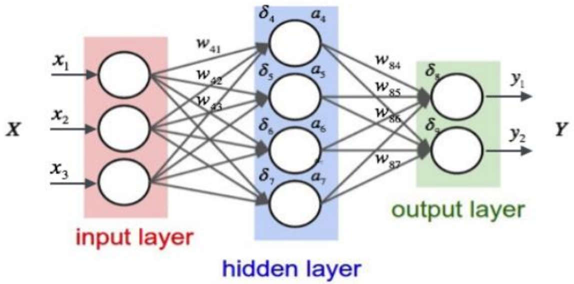
- Deep Learning shows great performance on complex tasks:
  - Computer Vision
  - Natural Language Processing
  - Robotics and Internet of Things

Recap on ML/DL

- It learns and represents complex patterns in the input data.

Stochastic Gradient Descent

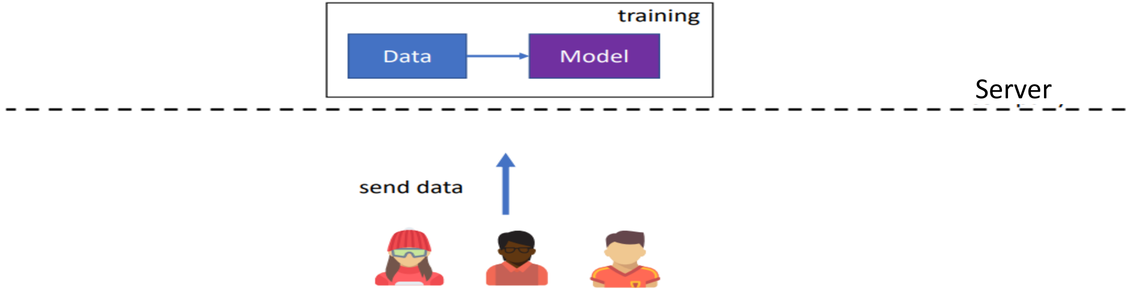
```
model = ML()
optimizer = SGD(model)
for e in epochs:
  for pos in range(batch_number):
    input, target = get_batch(dataset, pos)
    pred = model(input)
    loss = Loss(pred, target) # compare results
    loss.backward() # calculate updates to model
    optimizer.step() # apply model
```



Common ML/DL pipeline

Centralized Machine Learning

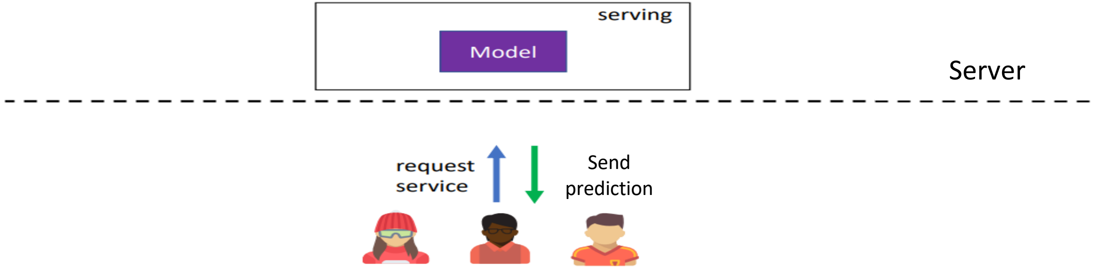
Training



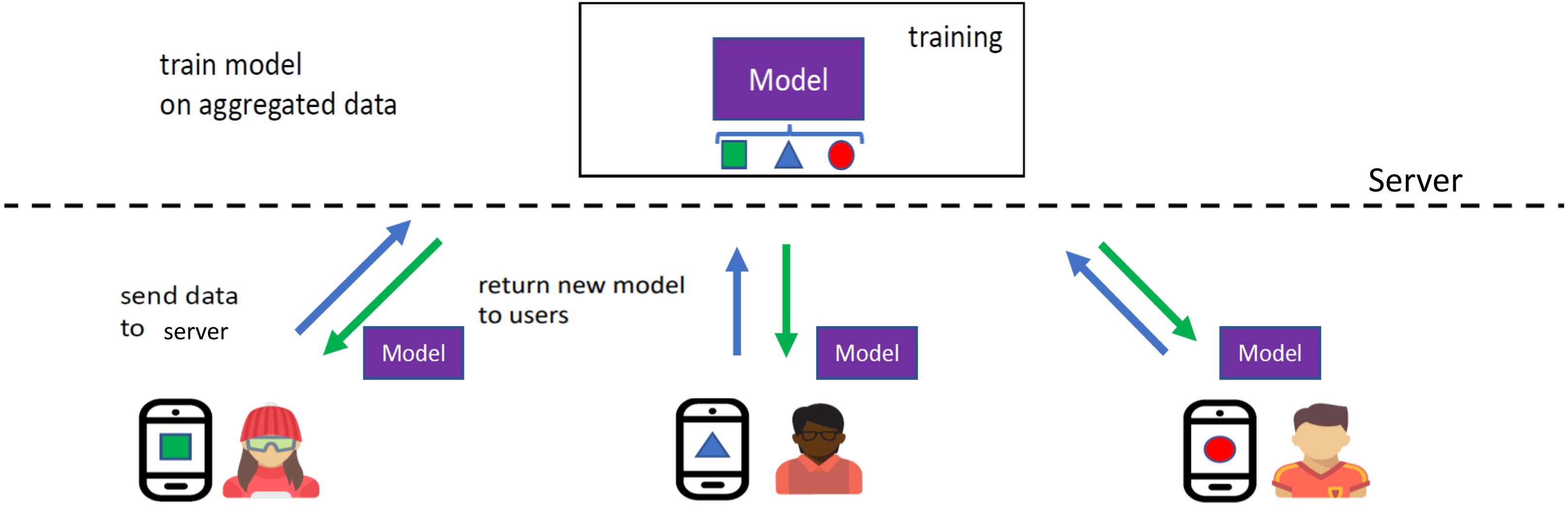
Issues:

- Slow training
- Network Latency
- Limited scalability
- Privacy and security

Testing/Evaluation



On-device inference / Distributed ML

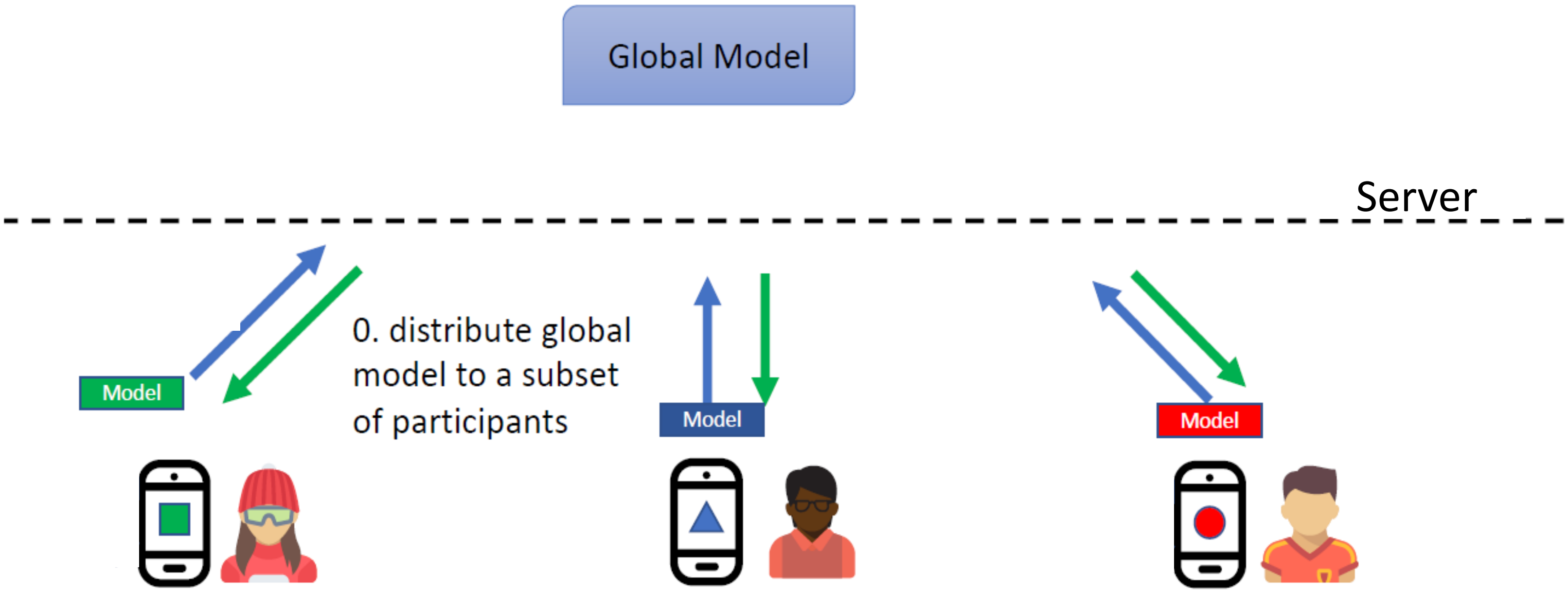


Federated Learning (FL)

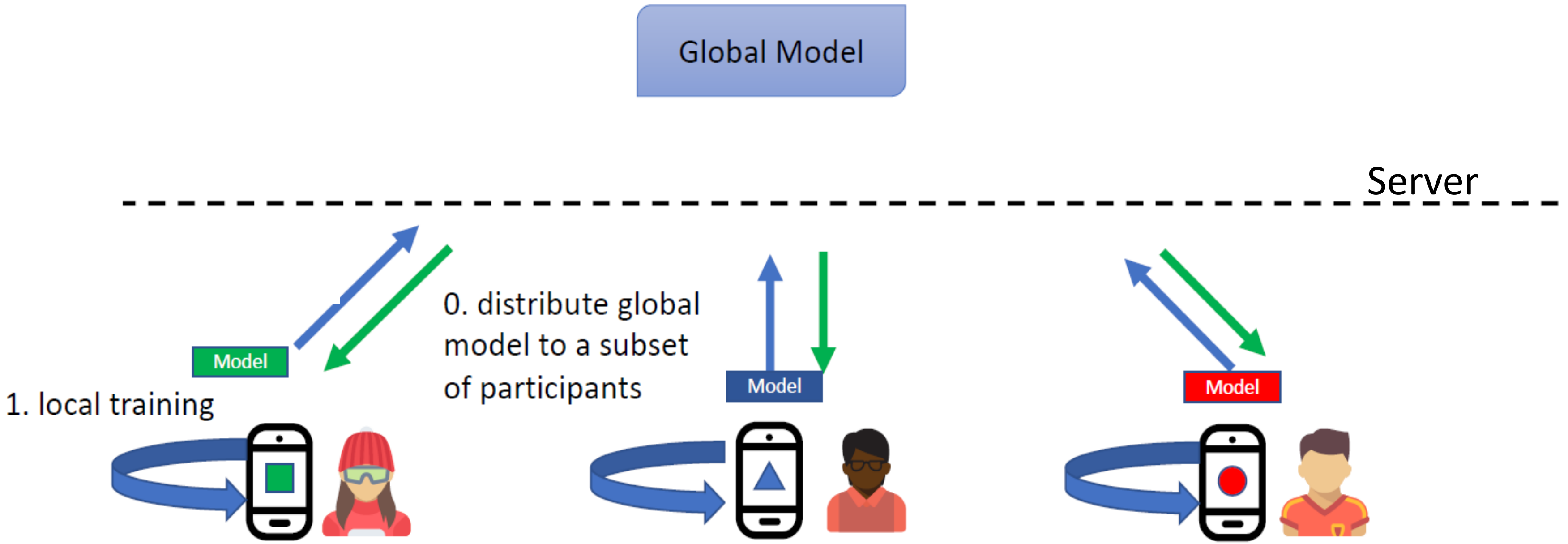
- In 2016, Google introduced the concept of FL, enabling collaborative ML without centralized training data.
- FL does not share local data but ML models, offering applications in diverse domains such as healthcare, finance, mobile and IoT applications.

- FL Advantage:
  - Smarter models
  - Reduced latency
  - Less communication cost
  - Privacy preserving

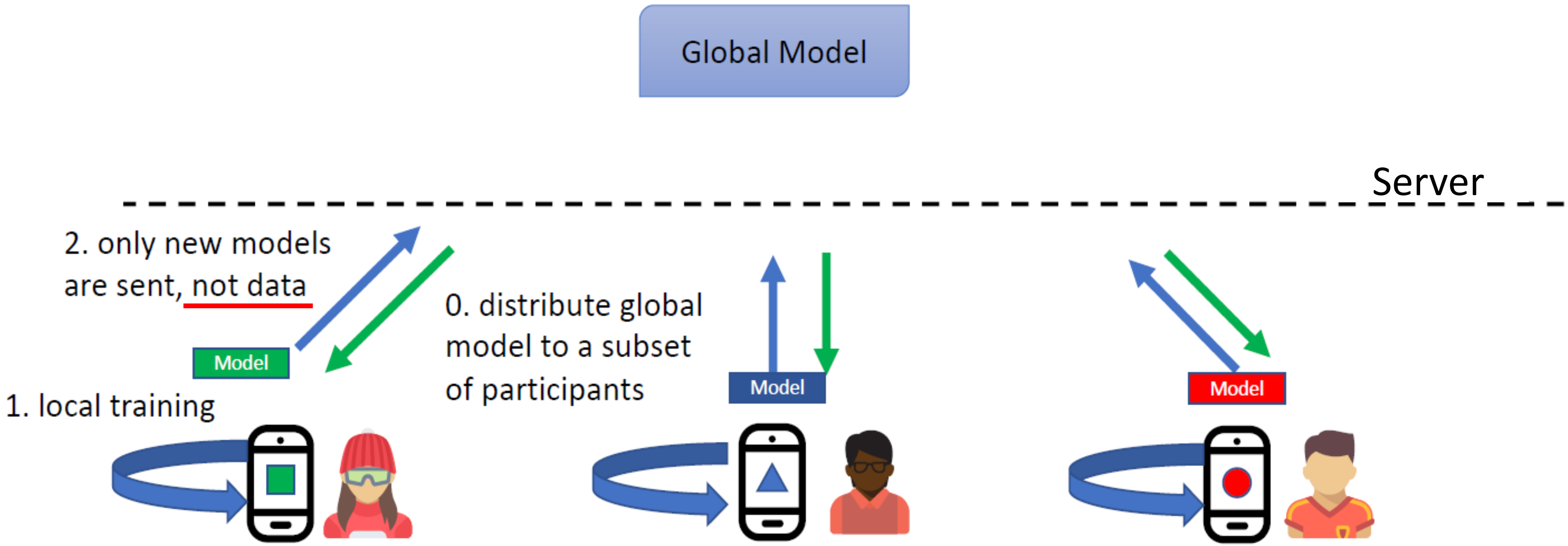
FL Workflow


Global Model
Server
0. distribute global model to a subset of participants
Model
Model
Model

FL Workflow

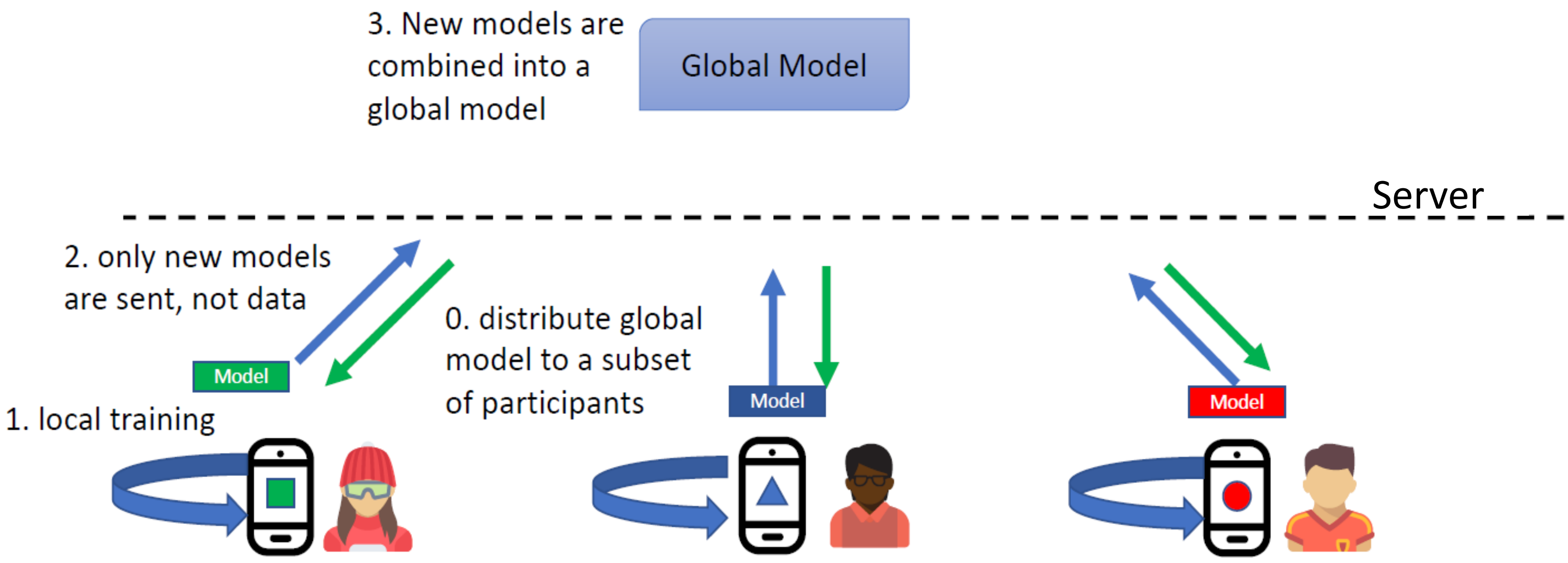

Global Model
Server
0. distribute global model to a subset of participants
Model
Model
Model
1. local training

FL Workflow

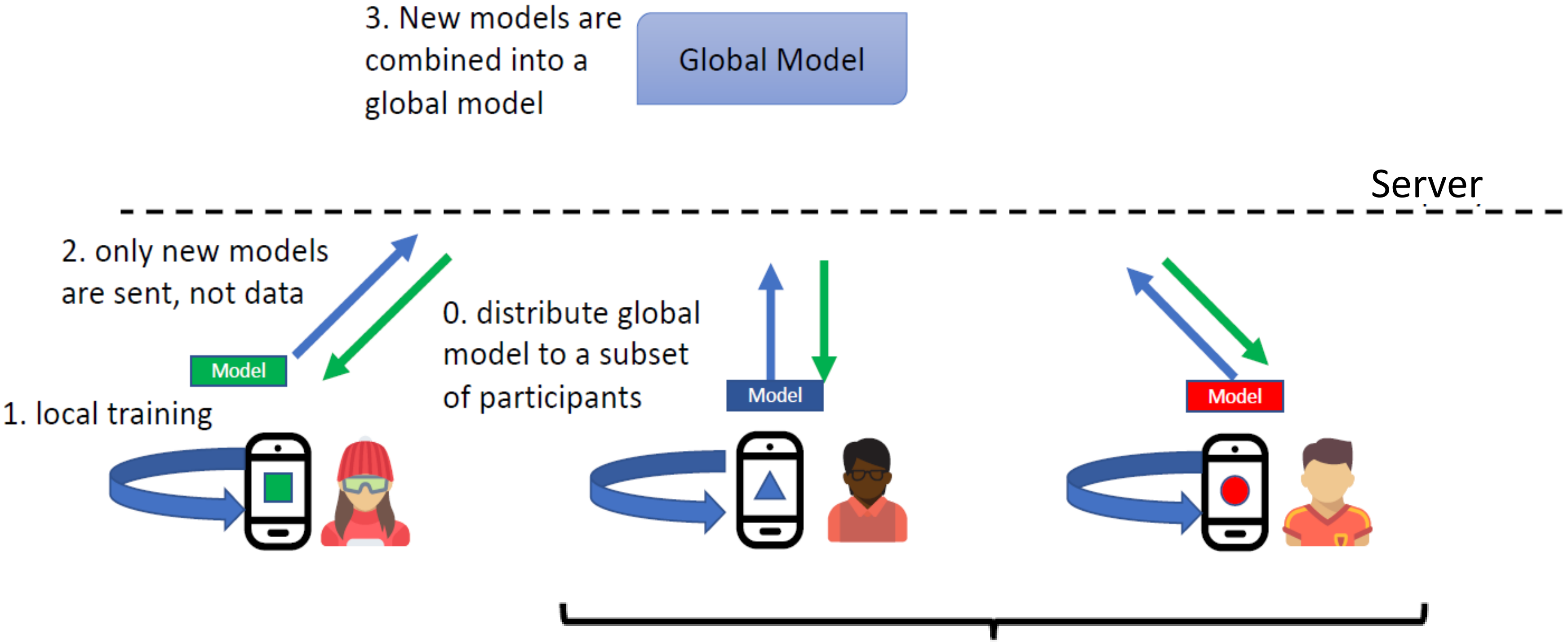


Global Model
Server
2. only new models are sent, not data
0. distribute global model to a subset of participants
1. local training
Model
Model
Model

FL Workflow


3. New models are
combined into a
global model
Global Model
Server
2. only new models
are sent, not data
0. distribute global
model to a subset
of participants
Model
Model
Model
1. local training

FL Workflow



Averaging subset of users allows to train model faster
Algorithm can tolerate drop of participants

User’s data distribution and capabilities

- **Non-IID** – every user has different data;
- **Unbalanced** – some users have more data than others;
- **Massively distributed** – millions of smartphones;
- **Limited communication** – mobile devices are frequently offline or on slow or expensive connections.

Model Aggregation

How to aggregate the models after they trained locally?

1. FedSGD (Federated Stochastic Gradient Descent): FedSGD is a simple federated learning algorithm that aggregates the local model updates using plain averaging. It does not perform any regularization and assumes that the data distribution across the devices is identically and independently distributed (IID).

2. FedAvg (Federated Averaging): FedAvg is a popular federated learning algorithm that uses a weighted averaging scheme to aggregate local model updates based on the number of examples on each device. FedAvg can handle non-IID data distributions and is more robust to noisy updates than FedSGD.

3. FedAvgM (Federated Averaging with Momentum): FedAvgM is an extension of FedAvg that adds momentum to the local and global updates to improve convergence. FedAvgM can improve the convergence speed and performance compared to FedAvg, especially for non-IID data distributions.

4. FedProx (Federated Proximal): FedProx is a federated learning algorithm that adds a proximal term to the local optimization objective to regularize the model and improve its robustness to noisy updates.

FedAvg Algorithm

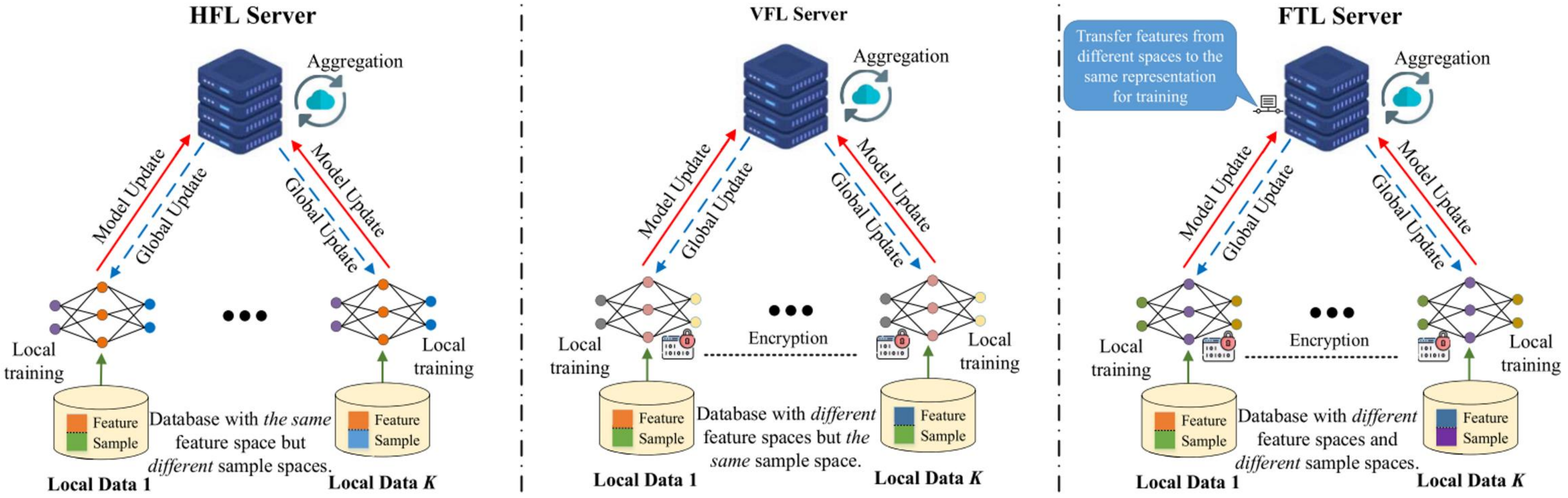
Server:
Initialize global model θ_0
for each communication round $t = 1, 2, \ldots T$ **do**
  Select $m = C \times K$ clients, where $C \in (0, 1)$
  for each **Client** $k = 1, 2, \ldots m$ in parallel **do**
    Download θ_t to **Client** k
    Do **Client** k update and receive θ^k
  end for
  Update global model $\theta_t \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} \theta^k$
end for

Client k update:
Replace local model $\theta^k \leftarrow \theta_t$
for local epoch from 1 to E **do**
  for batch $b \in (1, B)$ **do**
    $\theta^k \leftarrow \theta^k - \eta \nabla L_k(\theta^k, b)$
  end for
end for
Return θ^k

T is the total number of communication rounds,
C is the participation ratio assuming that not all local clients participate in each round of model updates
K is the total numbers of clients;
M =C*k is the number of participating clients
nk is the local data size, n is the total number of sample pairs
B is the size of mini-batches,
E is the total local training epochs,
n is the learning rate.
L is the loss function
k is the client index

Types of FL

- ***FL Models based on data partitioning***



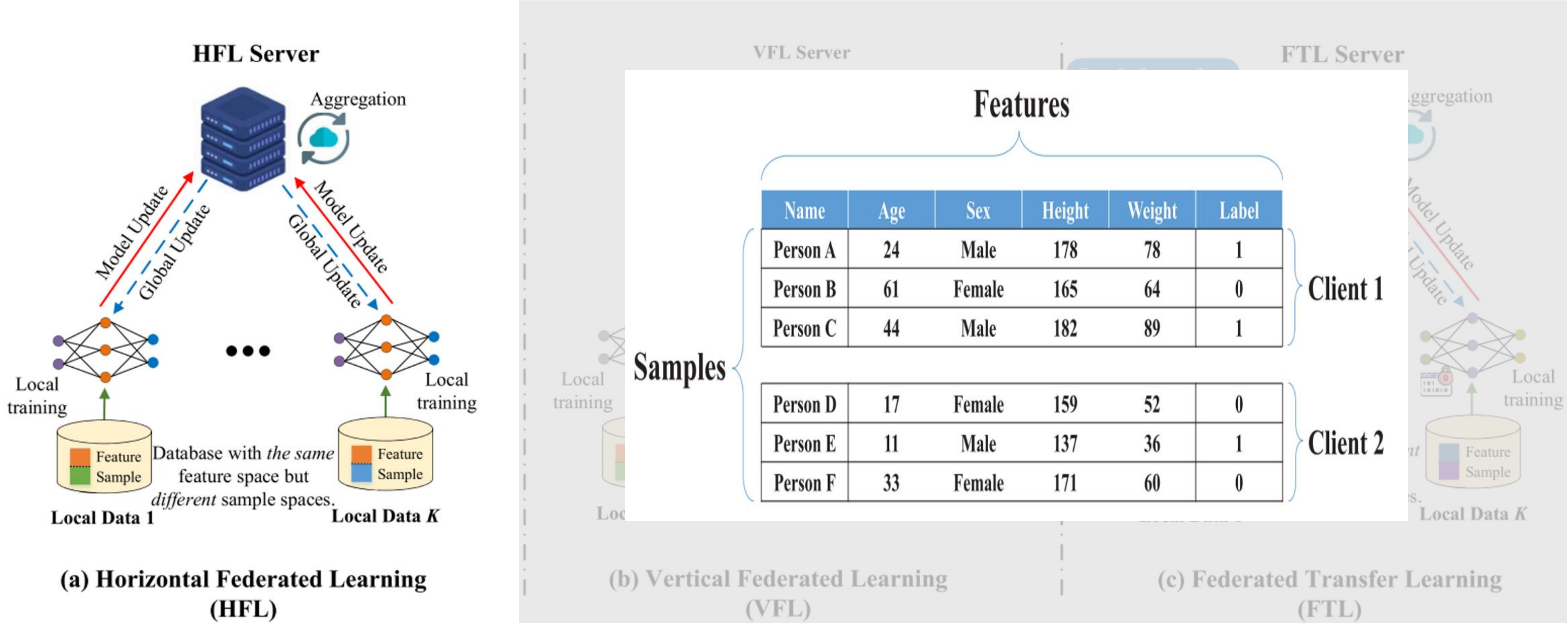
(a) Horizontal Federated Learning (HFL)

(b) Vertical Federated Learning (VFL)

(c) Federated Transfer Learning (FTL)

Types of FL

- ***FL Models based on data partitioning***



Name	Age	Sex	Height	Weight	Label
Person A	24	Male	178	78	1
Person B	61	Female	165	64	0
Person C	44	Male	182	89	1
Person D	17	Female	159	52	0
Person E	11	Male	137	36	1
Person F	33	Female	171	60	0

Types of FL

- ***FL Models based on data partitioning***

Features

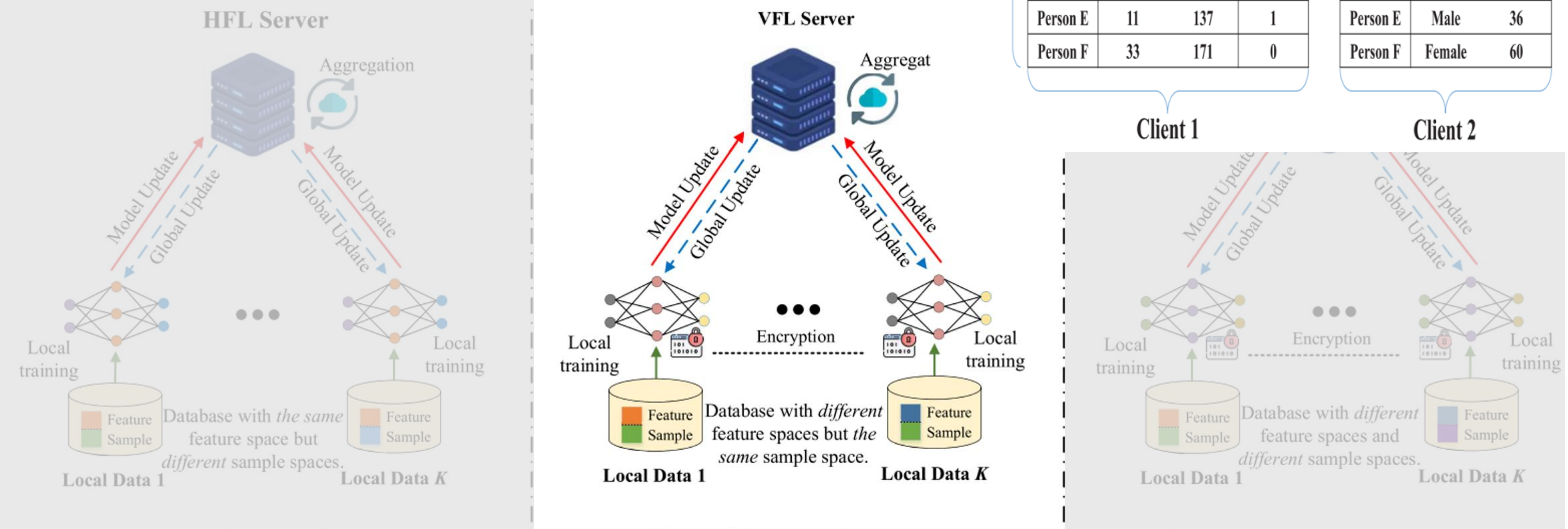
Samples

Name	Age	Height	Label
Person A	24	178	1
Person B	61	165	0
Person C	44	182	1
Person D	17	159	0
Person E	11	137	1
Person F	33	171	0

Client 1

Name	Sex	Weight
Person A	Male	78
Person B	Female	64
Person C	Male	89
Person D	Female	52
Person E	Male	36
Person F	Female	60

Client 2



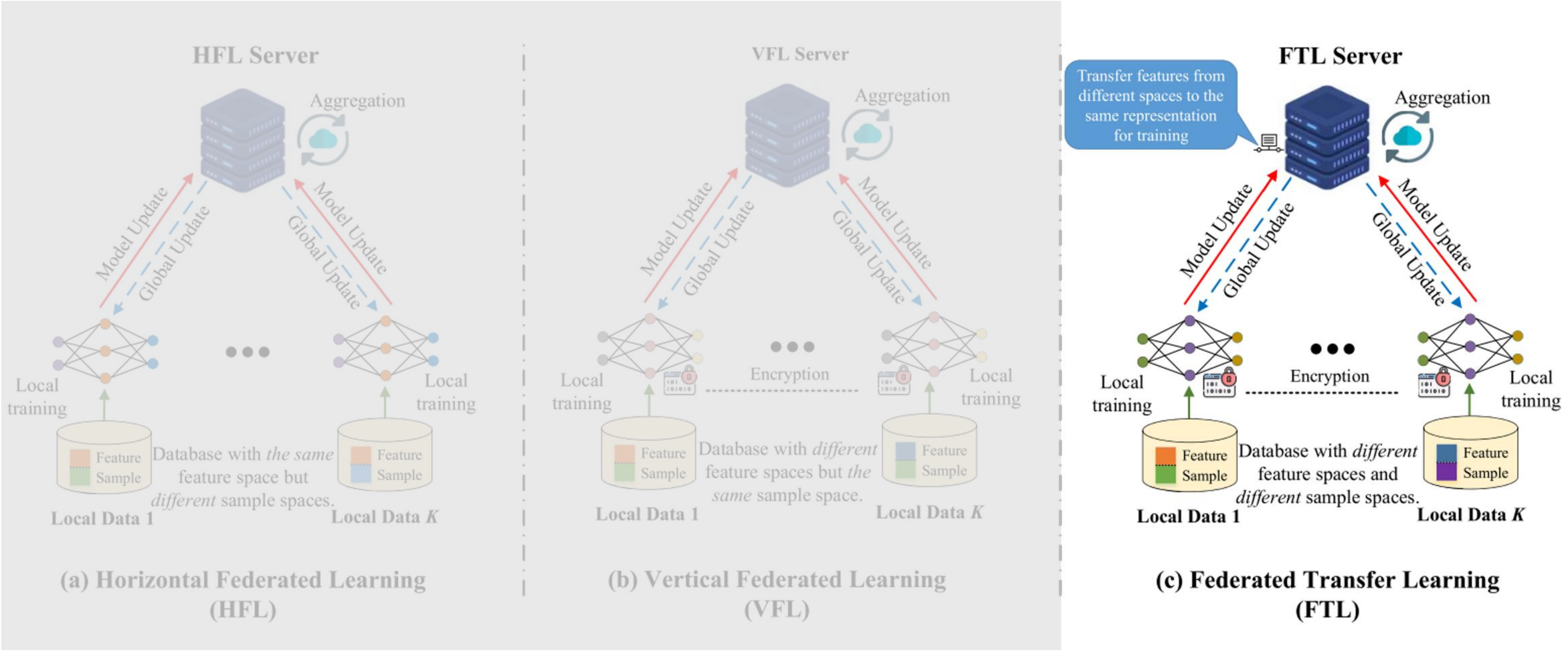
(a) Horizontal Federated Learning (HFL)

(b) Vertical Federated Learning (VFL)

(c) Federated Transfer Learning (FTL)

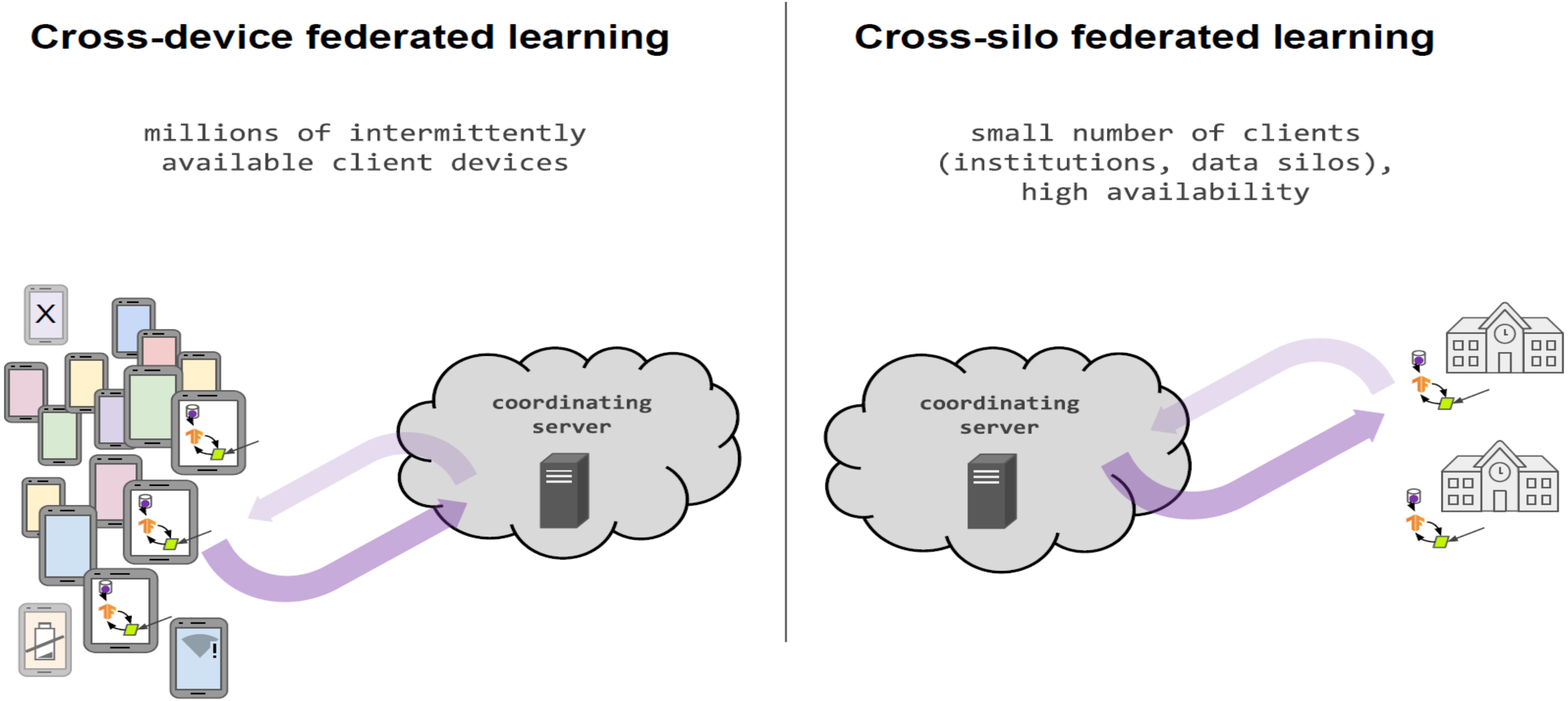
Types of FL

- **FL Models based on data partitioning**



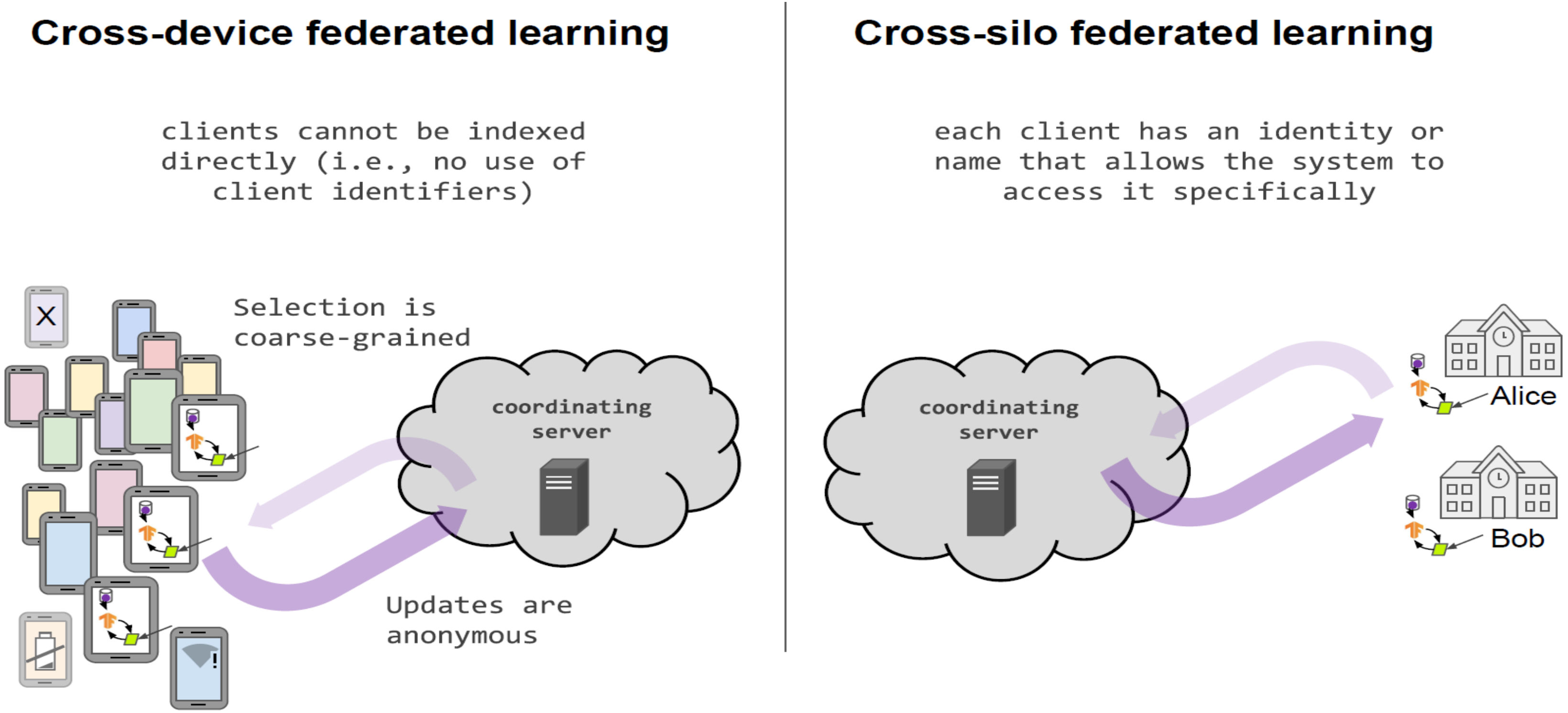
Types of FL

- **FL Models based on number and nature of clients**



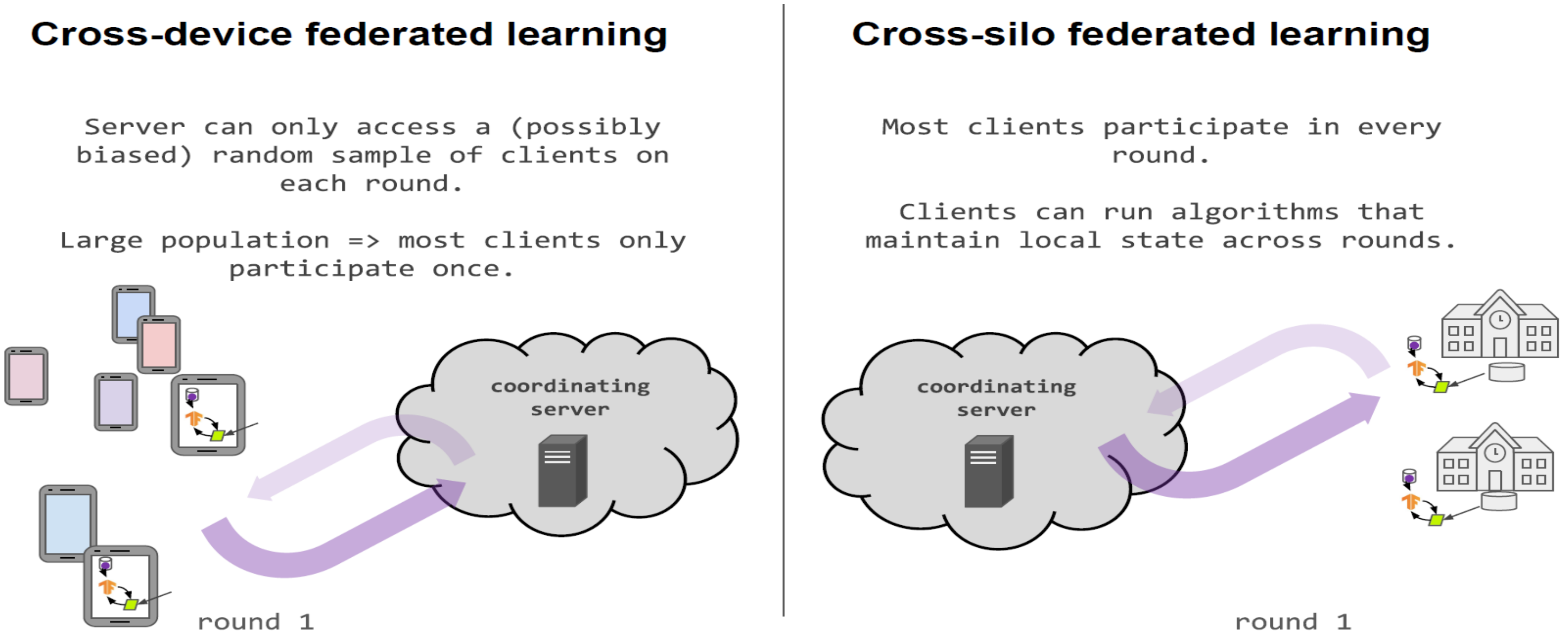
Types of FL

- **FL Models based on number and nature of clients**



Types of FL

- **FL Models based on number and nature of clients**



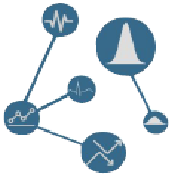
Challenges in Federated learning

expensive communication

- massive, slow networks

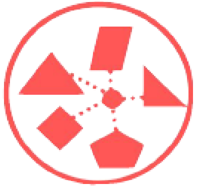
privacy concerns

- user privacy constraints



statistical heterogeneity

- unbalanced, non-IID data



systems heterogeneity

- variable hardware, connectivity, etc

Implementing FL

A reference list of popular federated learning repositories.

Name	Repository	License/Stars	Focus
TF Federated	https://github.com/tensorflow/federated	Apache 2.0 / 1.7k	R&D
FedJAX	https://github.com/google/fedjax	Apache 2.0 / 130	Research
Flower	https://github.com/adap/flower	Apache 2.0 / 529	Usability
FedML	https://github.com/FedML-AI/FedML	Apache 2.0 / 839	Research
PySyft	https://github.com/openmined/pysyft	Apache 2.0 / 7.7k	Privacy / R&D
IBM federated-learning-lib	https://github.com/IBM/federated-learning-lib	Custom / 244	Enterprise

Practical Implementation using Flower and Pytorch frameworks.

https://github.com/abdelwahed/FL_tutorial

Thank you

Dr Abdelwahed Khamis **Dr. Sara Khalifa**

Distributed Sensing Systems Research Group
Cyber Physical Systems Research Program
Data61, CSIRO