Yale

Federated Learning

CPSC680: Trustworthy Deep Learning

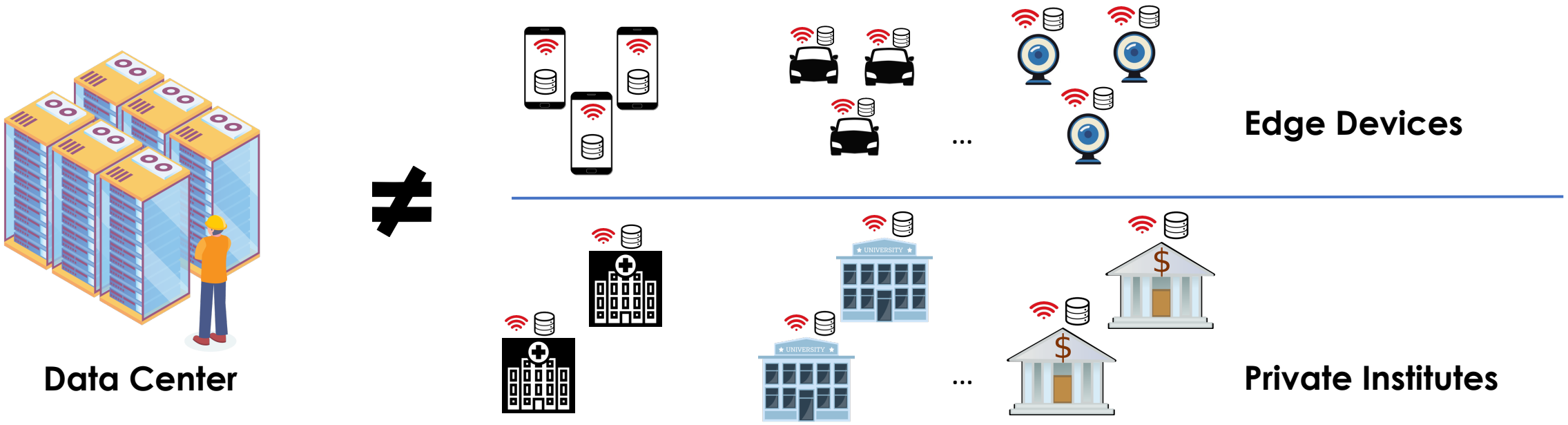
Rex Ying

Content

- Federated Learning
- Federated Learning Algorithms: FedSGD & FedAvg
- Challenges in Federated Learning
  - Data heterogeneity
  - Privacy-preserving

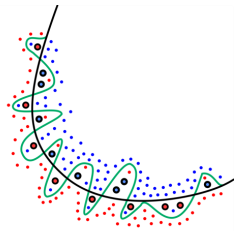
Centralized vs Decentralized Data

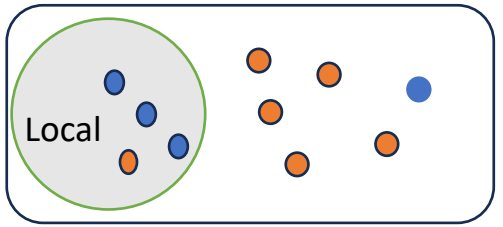
- Most settings to train Machine Learning models we have considered *so far* are centralized, where the data is stored ***centrally*** in a data center.
- However, in the real world, data is often generated and ***decentralized*** across many parties.
- **How can we learn from these decentralized data?**



Options

- **Don't use data** from multiple parties (no collaborations) to improve services
  - Local datasets may be too small to train a model
  - Local datasets may be biased



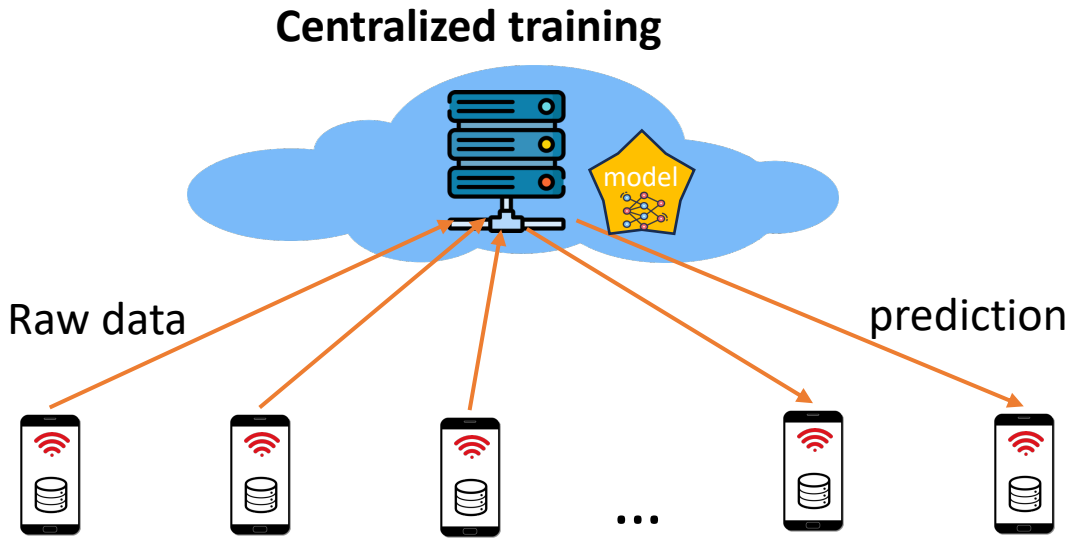


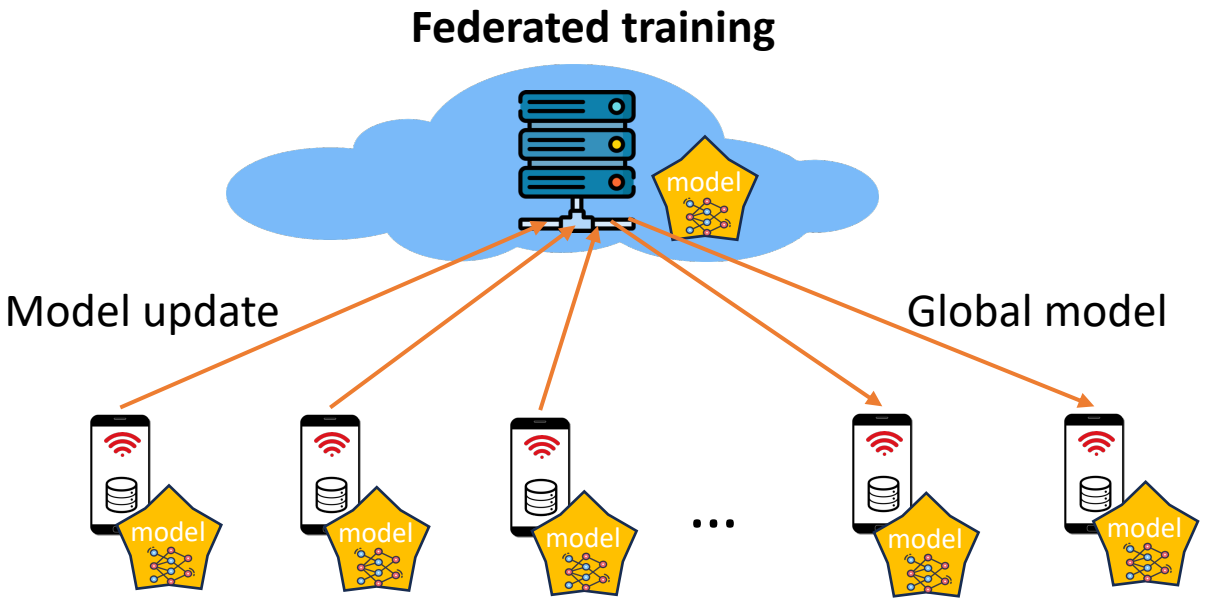
- **Centralize the data** from all parties to a data center.
  - Transmitting data could be prohibitively expensive
    - Self-driving cars may produce several TBs/day.
  - The sensitivity of data might be an issue
  - Privacy concerns

- **Innovate a new learning method** to learn from decentralized data
  → **Federated Learning** (today lecture)

What is Federated Learning?

- Federated Learning (FL) aims to **train an ML model collaboratively** across **multiple decentralized devices or servers** holding local data without explicitly exchanging data.



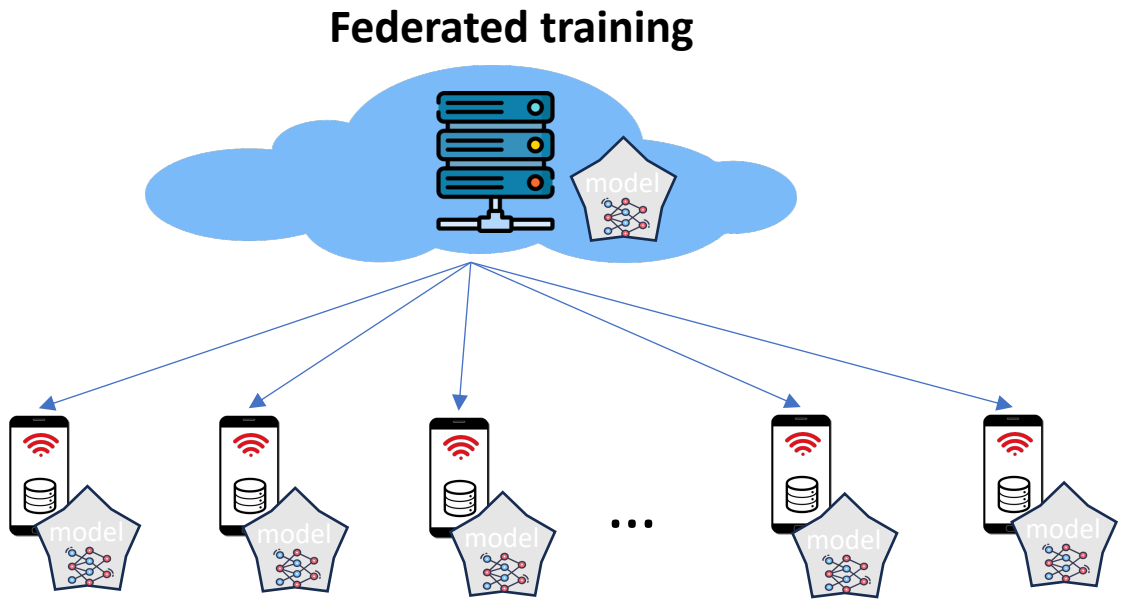


- **Advantages?**

✅ Quickly incorporate new data ✅ Reduce strain on network ✅ Privacy ✅ Work offline

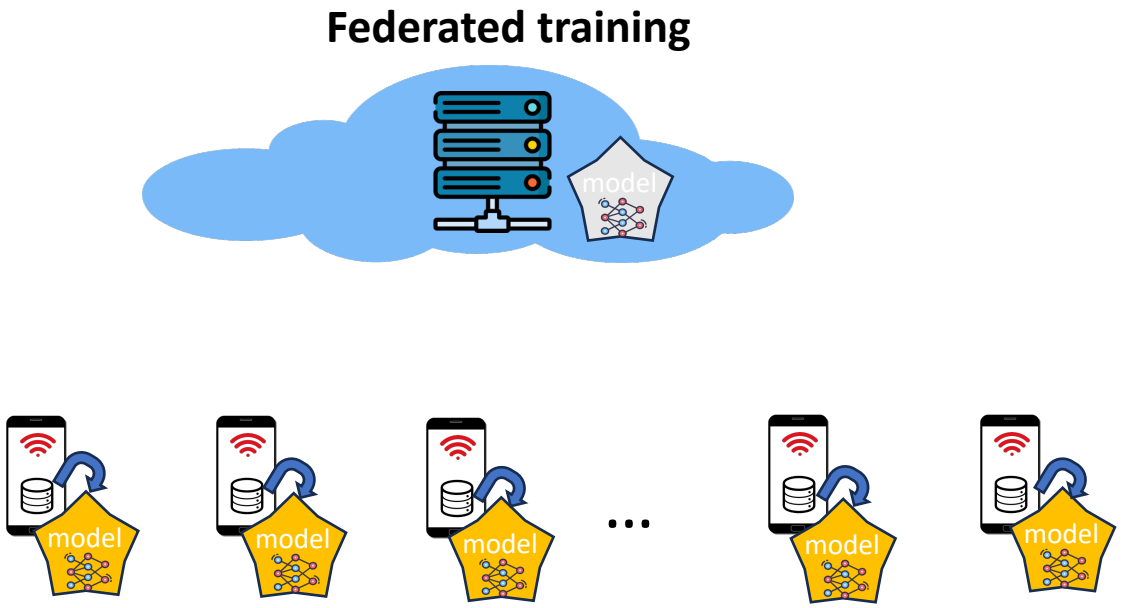
Federated Learning: Workflows

- Initialize/deploy the model to clients



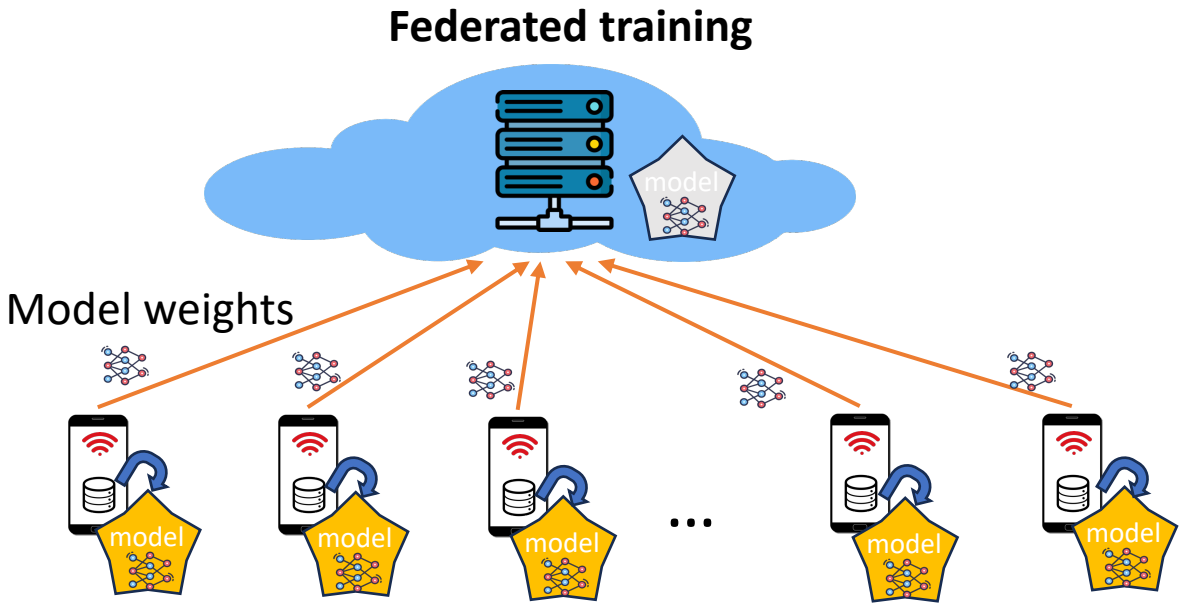
Federated Learning: Workflows

- Initialize/deploy the model to clients
- Each client updates its own model using its local dataset



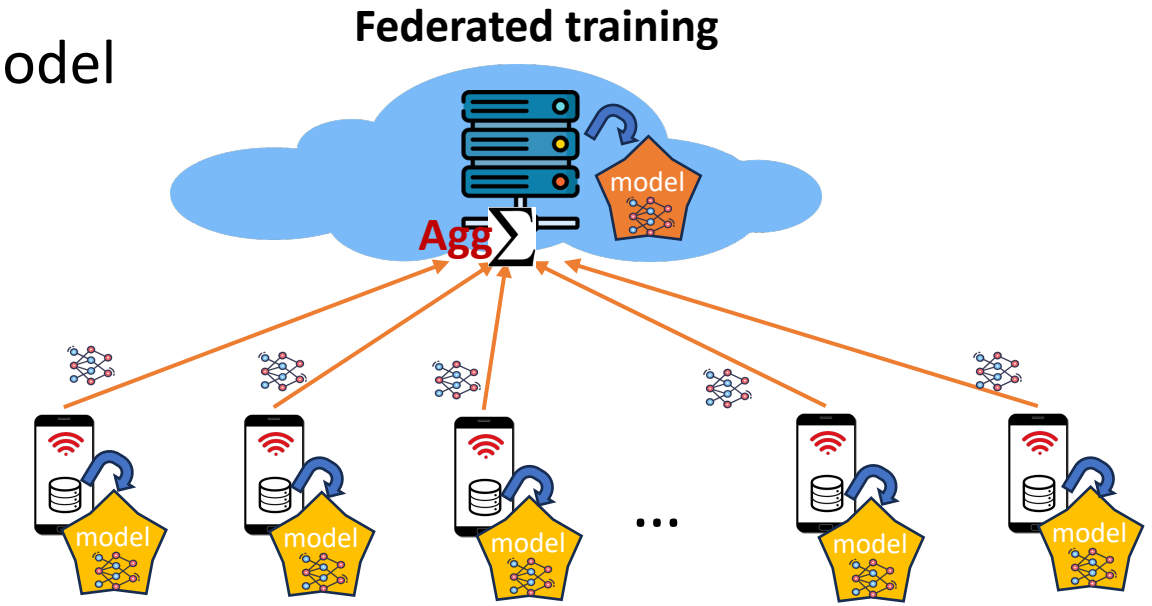
Federated Learning: Workflows

- Initialize/deploy the model to clients
- Each client updates its own model using its local dataset
- Clients share their model updates for aggregation



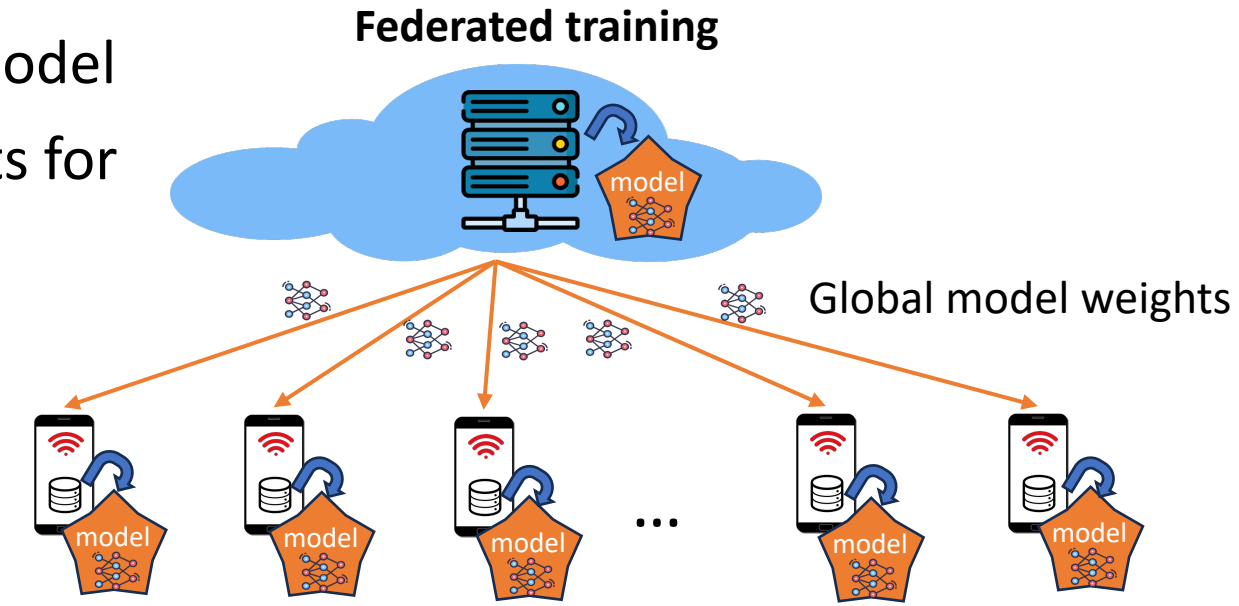
Federated Learning: Workflows

- Initialize/deploy the model to clients
- Each client updates its own model using its local dataset
- Clients share their model updates for aggregation
- Server aggregates the model into a global model



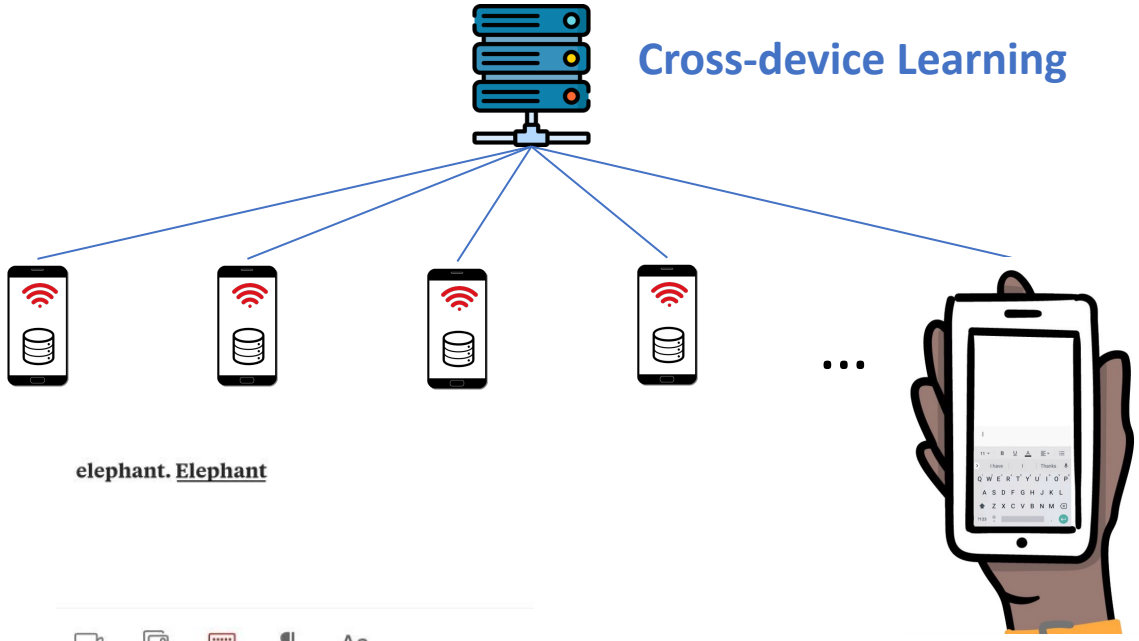
Federated Learning: Workflows

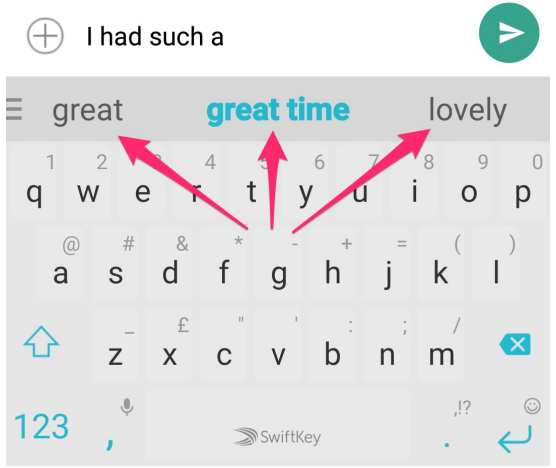
- Initialize/deploy the model to clients
- Each client updates its own model using its local dataset
- Clients share their model updates for aggregation
- Server aggregates the model into a global model
- Server sends back the global model to clients for inference

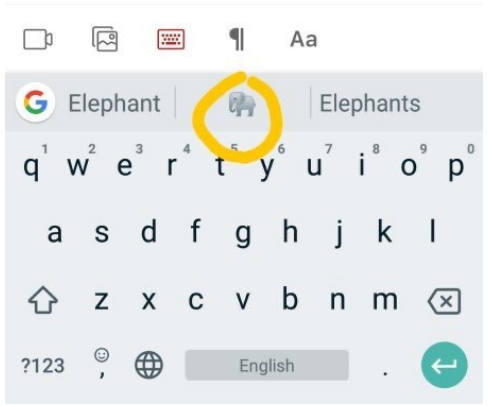


Real-world Applications of Federated Learning

- **Gboard** Google
  - **Next-word prediction**
  - **Emoji prediction**
  - **Action prediction**
    - Suggesting GIF, Stickers, Search queries





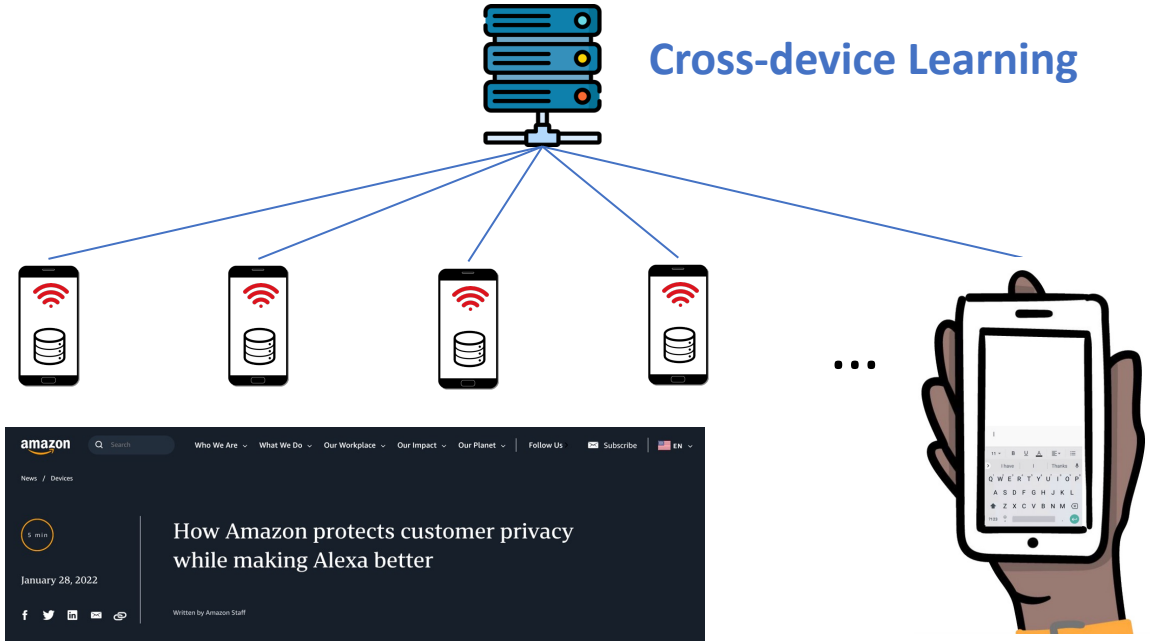


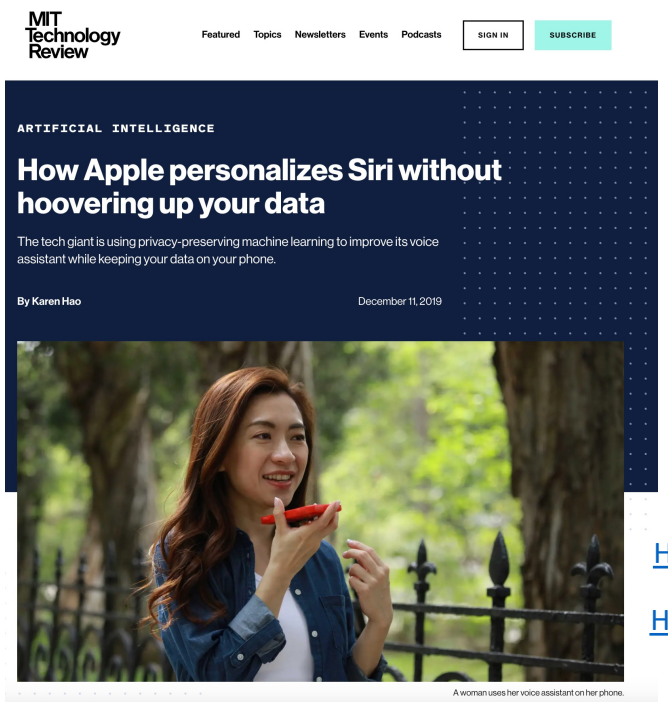
Real-world Applications of Federated Learning

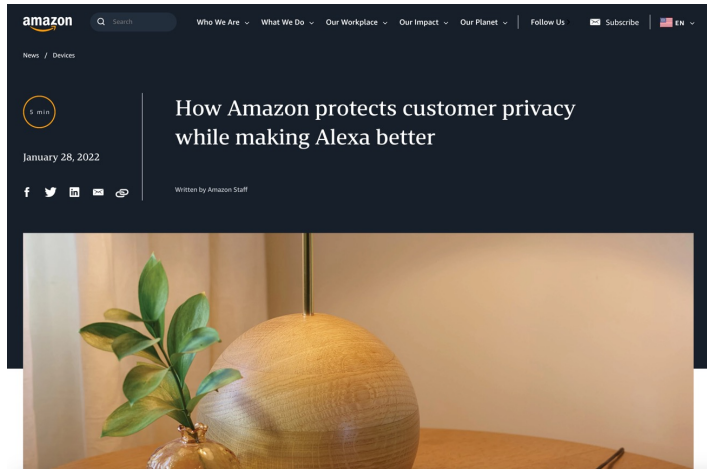
- **Gboard**
  - Next-word prediction
  - Emoji prediction
  - Action prediction
    - Suggesting GIF, Stickers, Search queries

- **Apple**
  - Personalized Siri

- **Amazon**
  - Alexa







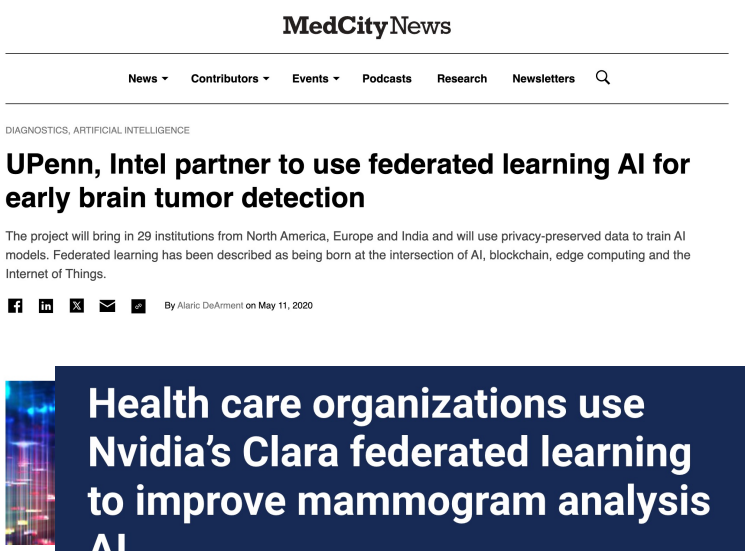
[How Apple personalizes Siri without hoovering up your data | MIT Technology Review]

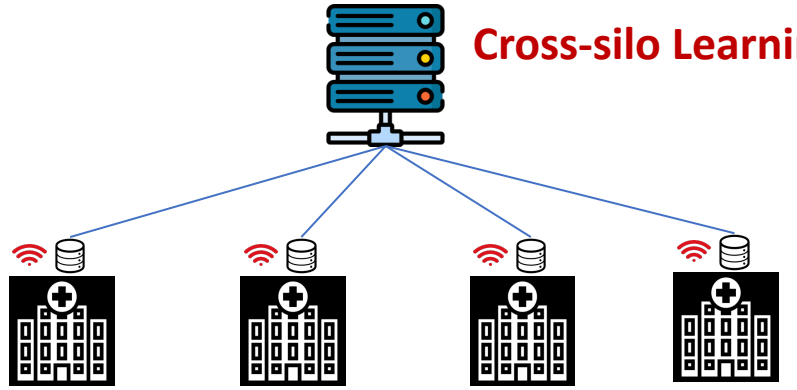
[How Amazon protects customer privacy while making Alexa better (aboutamazon.com)]

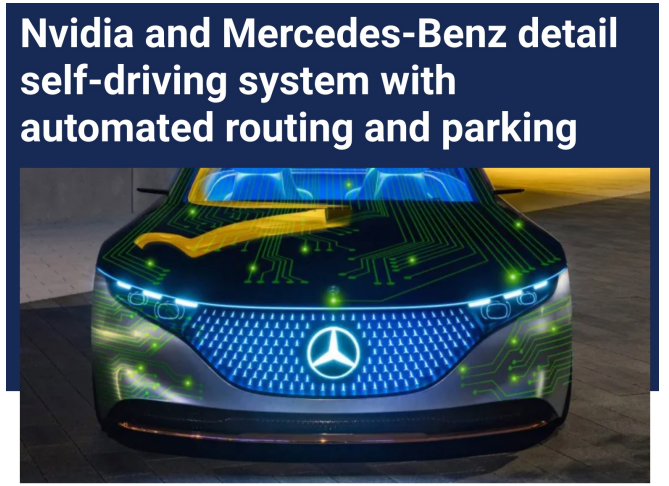
Real-world Applications of Federated Learning

- **Intel**
  - Brain tumor detection

- **Nvidia**
  - Mammogram analysis







[UPenn, Intel partner to use federated learning AI for early brain tumor detection - MedCity News]

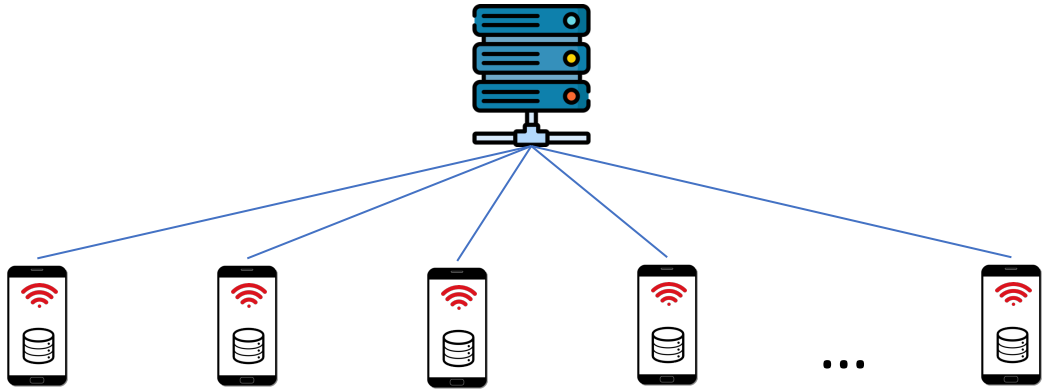
[Health care organizations use Nvidia's Clara federated learning to improve mammogram analysis AI | VentureBeat]

[Nvidia and Mercedes-Benz detail self-driving system with automated routing and parking | VentureBeat]

Cross-device vs. Cross-silo

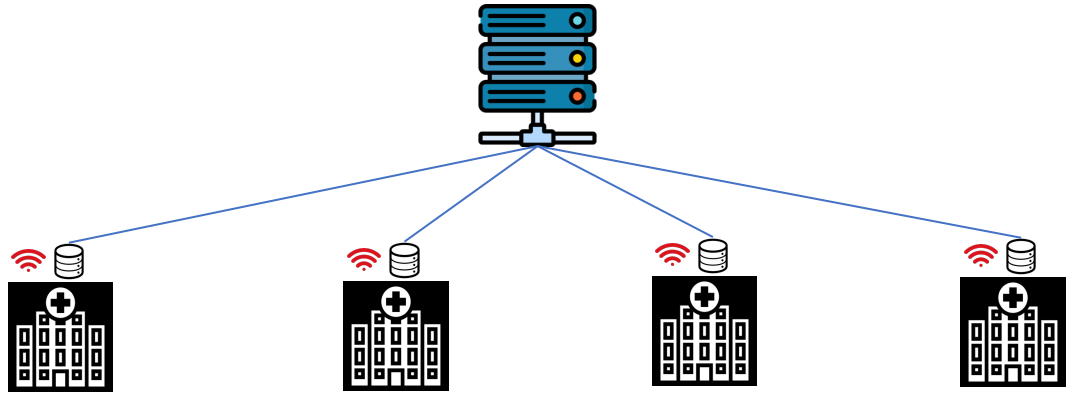
- Two main use cases of learning on decentralized data:

Cross-device Federated Learning



- Massive number of clients
- Each client holds a small dataset (maybe 1 data point)
- Limited availability and reliability
- Clients are anonymous, may be malicious

Cross-silo Federated Learning

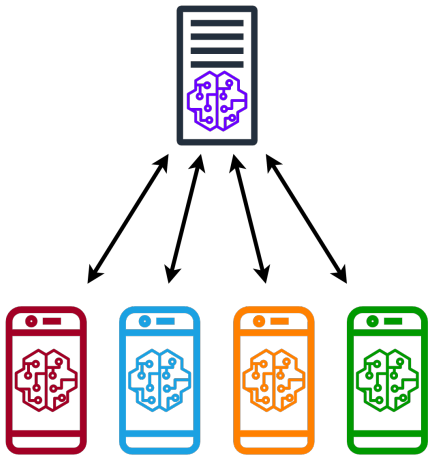


- Smaller number of clients
- Each client holds medium to large datasets
- Reliable parties, almost always available
- Clients are typically honest

Server Orchestrated vs. Fully Decentralized

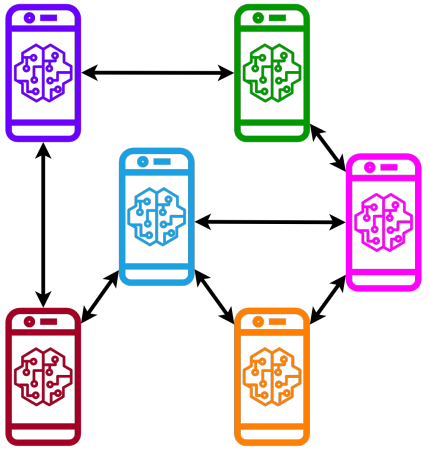
- Network settings

Server-orchestrated



- Server-client communication
- Global coordination and aggregation
- Failure at the server may cause bottlenecks

Fully Decentralized



- Peer-to-peer communication
- Aggregation at the local level, not globally
- Fault tolerance

Image credit: wiki

Content

- Federated Learning
- Federated Optimization: FedAvg
- Challenges in Federated Learning
  - Data heterogeneity
  - Privacy-preserving

Federated Optimization

- **Settings**
  - m clients
  - Each client holds a dataset $\mathcal{D}_k$ of n_k datapoints
  - Let $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_m$ be the joint datasets with $n = \sum_k n_k$ datapoints
- **Our goal** is to train the neural network on the dataset $\mathcal{D}$ by optimizing the loss function

$$\min_\theta F(\theta; \mathcal{D}) = \min_\theta \sum_k p_k \underbrace{F_k(\theta; \mathcal{D}_k)}_{\text{Optimized locally on devices}}$$

where θ is the model parameters

p_k is the weight for each client (e.g., $p_k = \frac{n_k}{n}$)

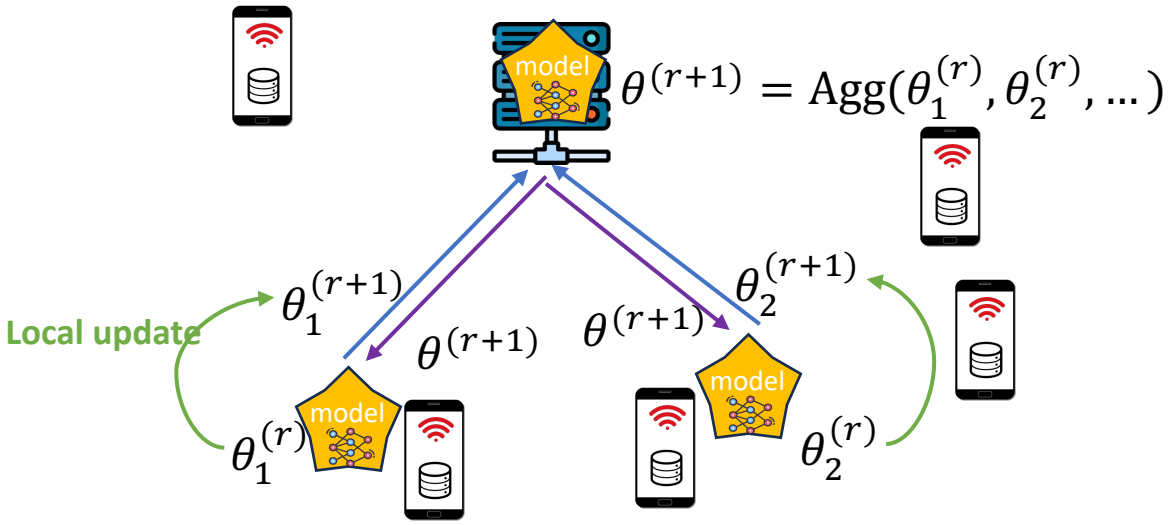
$F_k(\theta; \mathcal{D}_k)$ is typically an empirical risk minimization (ERM) objective

$$F_k(\theta; \mathcal{D}_k) = \sum_{(x,y)\in\mathcal{D}_k} \ell(h(x;\theta), y)$$

Federated Optimization: FedAvg

Server side

- M: number of clients joining the network
- ρ: client subsampling rate
- Initialize θ
- For each round $r = 0, 1 \ldots$ do
  - $S_r \leftarrow$ random set of $m = [\rho M]$ clients
  - For each client $k \in S_r$ do **(in parallel)**
    - $\theta_k^{(r+1)} \leftarrow \text{ClientUpdate}(k, \theta_k^{(r)})$
  - $\theta^{(r+1)} \leftarrow \sum_k p_k \theta_k^{(r+1)}$



ClientUpdate(k, θ)

- For each local step $1, \ldots, L$ do
  - $\mathfrak{B} \leftarrow$ mini-batch sampled from $\mathfrak{D}_k$
  - $\Delta\theta \leftarrow \Delta\theta - \eta \nabla F_k(\theta; \mathfrak{B})$
  - $\theta \leftarrow \theta + \Delta\theta$
- Send θ back to server

How does FedAvg differ from distributed SGD?

McMahan et al. "Communication-efficient learning of deep networks from decentralized data." AISTATS, 2017.

Federated Learning vs. Distributed Learning

Federated Learning

ClientUpdate(k, θ)

- For each local step $1, \dots, L$ do
  - $\mathfrak{B} \leftarrow$ mini-batch sampled from $\mathfrak{D}_k$
  - $\Delta\theta \leftarrow \Delta\theta - \eta \nabla F_k(\theta; \mathfrak{B})$
  - $\theta \leftarrow \theta + \Delta\theta$
- **Send θ** back to server

- Data is **naturally distributed**.
- The goal is to **train without explicitly transferring data samples.**
- Data is usually **non-IID and imbalanced**
- If L=1, FedAvg → FedSGD

Distributed Learning

ClientUpdate(k, θ)

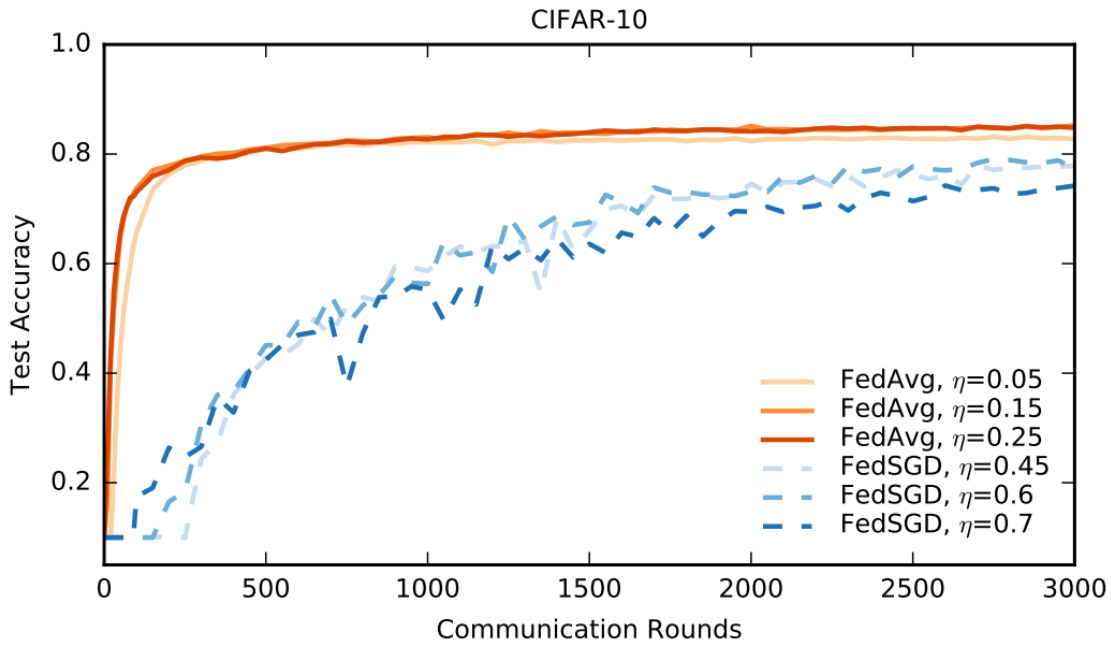
- For each local step $1, \dots, L$ do (typically, $L = 1$)
  - $\mathfrak{B} \leftarrow$ mini-batch sampled from $\mathfrak{D}_k$
  - $\Delta\theta \leftarrow \Delta\theta - \eta \nabla F_k(\theta; \mathfrak{B})$
- **Send $\Delta\theta$** back to server

- Data is **centrally stored**.
- 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

Why do we want multiple local updates (L > 1)?

- Leveraging local computation
- Integrate updates more quickly (instantly use local gradients).

Federated Optimization: FedAvg



McMahan et al. "Communication-efficient learning of deep networks from decentralized data." AISTATS, 2017.

- FedAvg with L > 1 helps models **converge more quickly**, thus reducing the number of communication rounds **($\sim$100x compared to FedSGD)**
- Usually generalize better than FedSGD with large mini-batch.
- **However, local updates may also hurt the performance if not properly turned (especially for non-IID data)**

(Will be discussed in next section)

Content

- Federated Learning
- Federated Optimization: FedAvg
- Challenges in Federated Learning
  - Privacy-preserving
  - Data heterogeneity

Challenges in Federated Learning

- **Expensive communication** 
  - transmitting large volumes of model updates between numerous devices and a central server, which can be bandwidth-intensive and slow

- **System heterogeneity** 
  - Each party may have different hardware available with varying network constraints.

- **Statistical heterogeneity** 
  - Data across parties are usually unbalanced, non IID

- **Privacy concerns** 
  - Data used in federated learning is typically sensitive

 **This Lecture**

[Advances and Open Problems in Federated Learning (arxiv.org)]

[Federated Learning: Challenges, Methods, and Future Directions (arxiv.org)]

Content

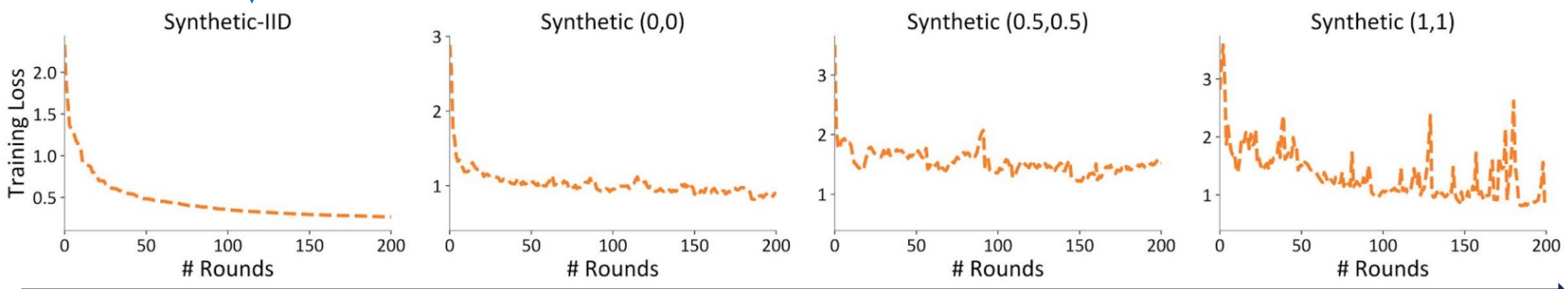
- Federated Learning
- Federated Optimization: FedAvg
- Challenges in Federated Learning
  - Data heterogeneity
  - Privacy-preserving

Heterogeneity in Federated Learning

- When local datasets are non-IID, more local updates may lead to slower convergence, reduced stability, or even divergence
- Synthetic (α, β)
  - (X_k, Y_k) where $Y_k = \text{argmax}(\text{softmax}(W_k X_k + b_k))$
  - $X_k \sim \mathcal{N}(v_k, \Sigma), v_k \sim \mathcal{N}(B_k, 1), B_k \sim \mathcal{N}(0, \beta)$
  - $W, b \sim \mathcal{N}(u_k, 1),\ u_k \sim \mathcal{N}(0, \alpha)$

Data in all clients come from the **same** distribution

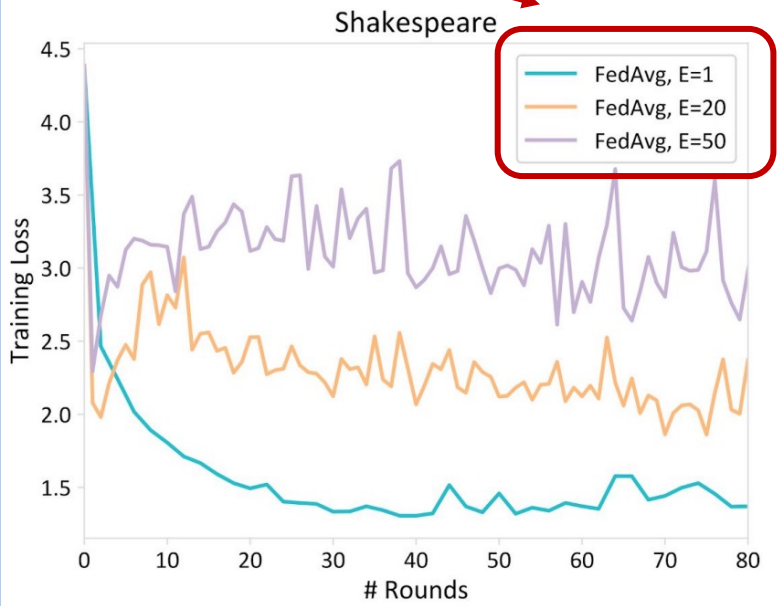
Client data are drawn from distributions **far** from each other



Effect of data heterogeneity on convergence

The model becomes harder to converge as the data heterogeneity increases

E is the number of local updates



How to explain this phenomenon?

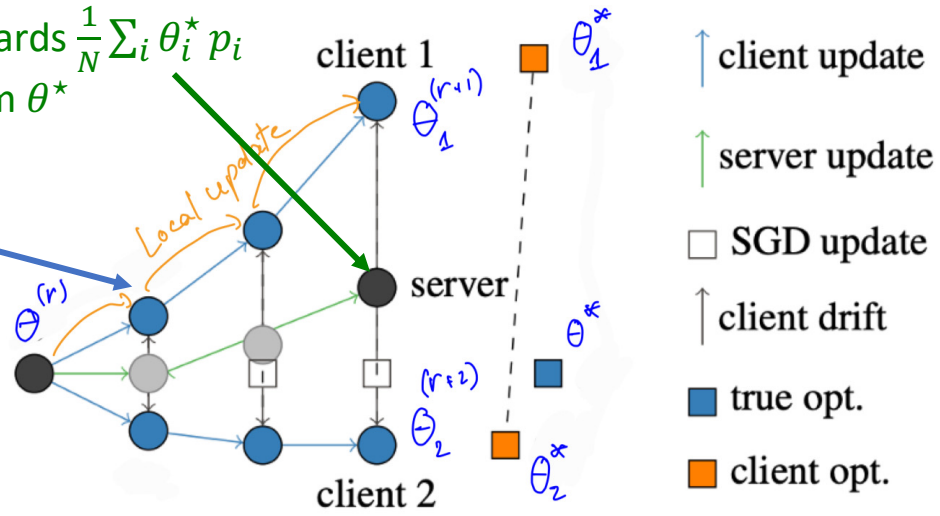
Li et al, Federated optimization in heterogeneous networks, MLSys 2020

Heterogeneity in Federated Learning

- When local datasets are non-IID, **FedAvg suffers from client drifts.**

The server updates move towards $\frac{1}{N}\sum_i \theta_i^\star p_i$ instead of to the true optimum $\theta^\star$

The local updates move towards the individual client optima $\theta_i^\star$



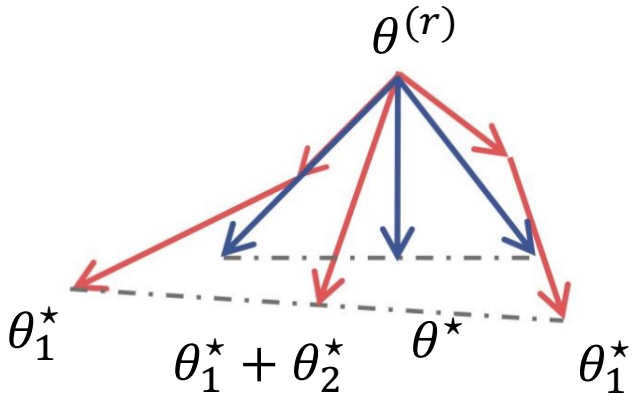
Karimireddy et al. "Scaffold: Stochastic controlled averaging for federated learning." ICML, 2020.

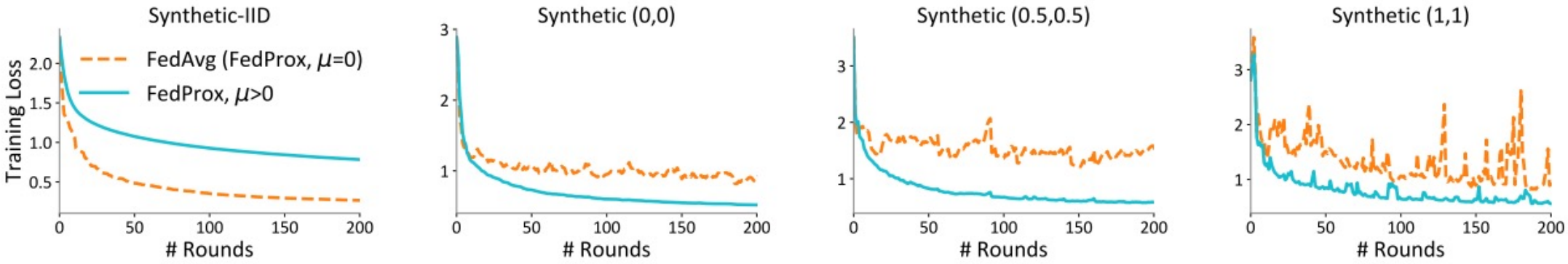
- Over-optimizing the model on the client side may introduce more bias to the global model, leading to a significant drift from the true optimal weights.

Heterogeneity in Federated Learning: FedProx

- **FedProx**: add a proximal regularization term to constrain the local updates to not drift too much from the previous round.

$$\min_{\theta_k} F_k(\theta_k; \mathcal{D}_k) + \frac{\mu}{2} ||\theta_k - \theta^{(r)}||_2$$





Effect of data heterogeneity on convergence

Another approach: [SCAFFOLD [KARIMIREDDY ET AL., 2020]]

Li et al, Federated optimization in heterogeneous networks, MLSys 2020

Personalized Federated Learning

- In some applications (e.g, Siri of Apple or Alexa of Amazon), you may want to personalize and adapt the global model to data and external conditions from each device.
- Learning a global (one-fit-all) model may require a large model, and the training may be slow and hard to converge as **each client may want to optimize the model differently**.

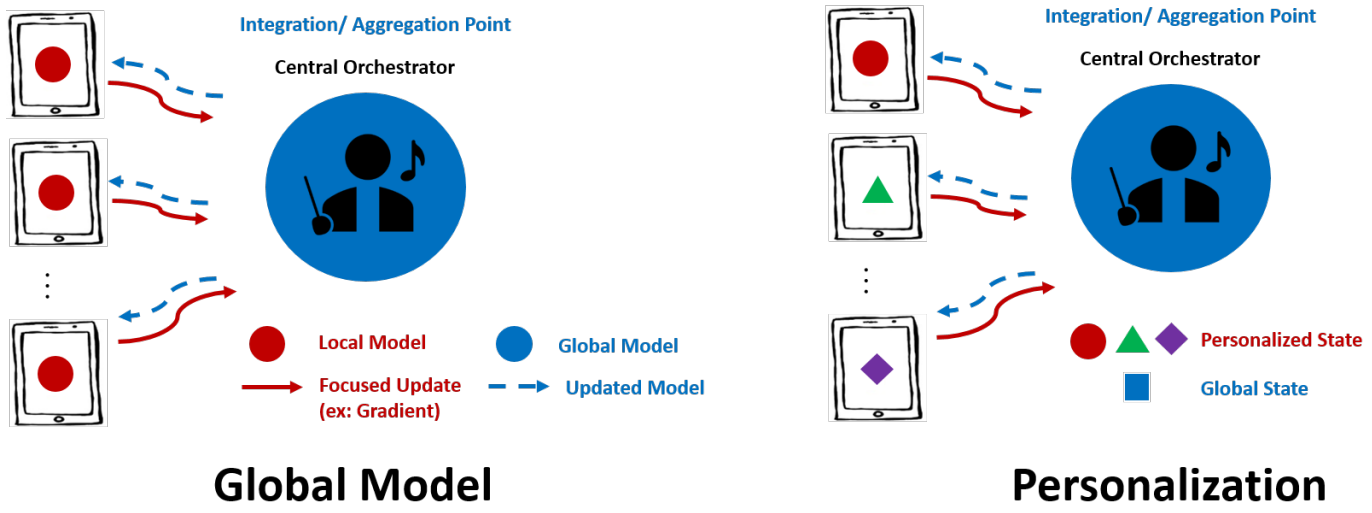


Image credit: [Internet of Federated Things – Home for Internet of Federated Things (umich.edu)]

Personalized FL via Multi-task learning

- Given the heterogeneity of data, we can **view federated learning as multi-task learning** (MTL) and each device/client's training objective as a separate 'task.'
- Our goal is to train a global model to **improve the generalization** by using training **signals from related tasks (clients)**.

$$\min_{\theta,\Omega} F(\theta_1, \dots, \theta_m, \Omega; \mathcal{D}) = \min_{\theta,\Omega} \left[\sum_k p_k F_k(\theta_k; \mathcal{D}_k) \right] + \left[\sum_{k<l} R(\theta_k, \theta_l, \Omega_{k,l}) \right]$$

Model parameters

Task relationship parameters

Task loss

Regularizer, similar task has similar parameters

- θ, Ω can be found by **alternating optimization schemes**
- k, l are the client indices

Smith, Virginia, et al. "Federated multi-task learning." NIPS (2017).

Personalized FL: Other Solutions

- **Fine-tuning**
  - We learn a global model first and then fine-tune the global model on the local datasets
- **Meta learning (initialization-based)**
  - Learn initialization over multiple tasks, then train locally
- ...

Fallah et al. "Personalized federated learning: A meta-learning approach." ,2020.

Tan, Alysa Ziying, et al. "Towards personalized federated learning." IEEE Transactions on Neural Networks and Learning Systems (2022).

Content

- Federated Learning
- Federated Optimization: FedAvg
- Challenges in Federated Learning
  - Data heterogeneity
  - Privacy-preserving

Privacy in Federated Learning

- We often need to deal with sensitive data in federated learning settings
  → Privacy is a crucial factor
- Federated learning exhibits **an additional attack surface** as the server and clients have to share model weights through networks.
- Malicious participants can easily join the network. Even **the orchestrator can also be dishonest**.
  → **Privacy risks**

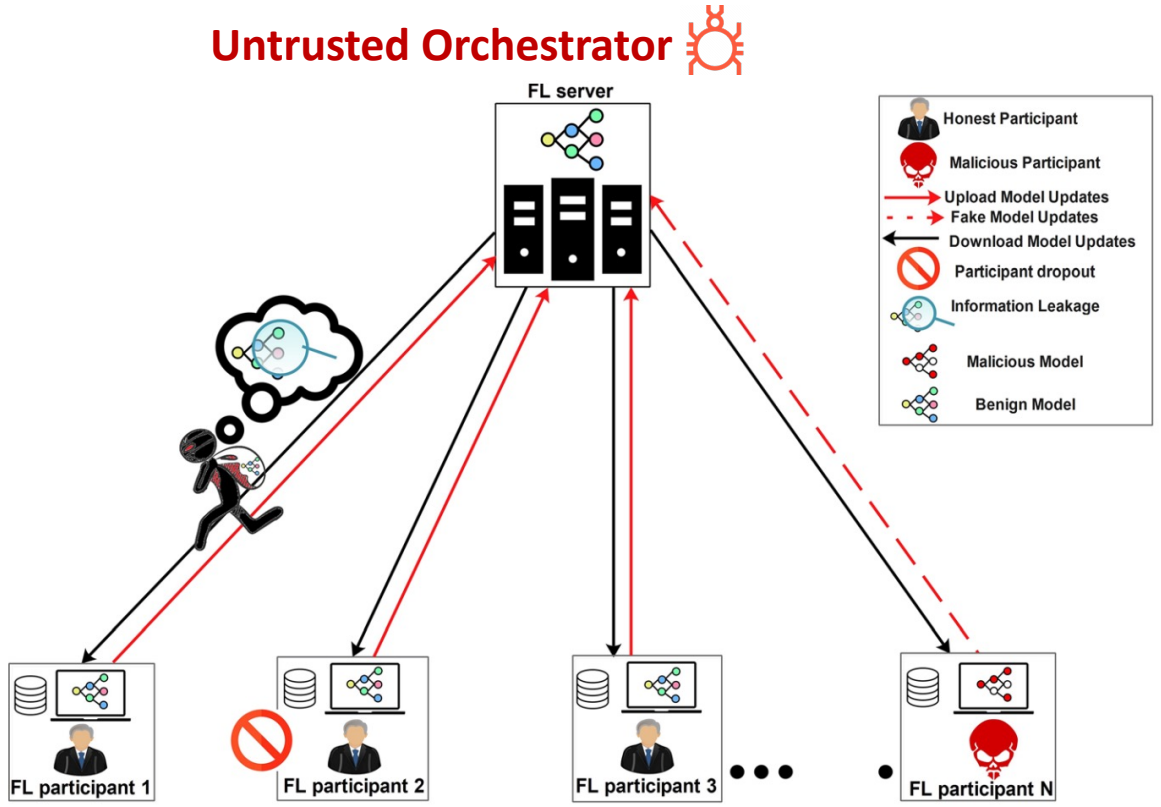


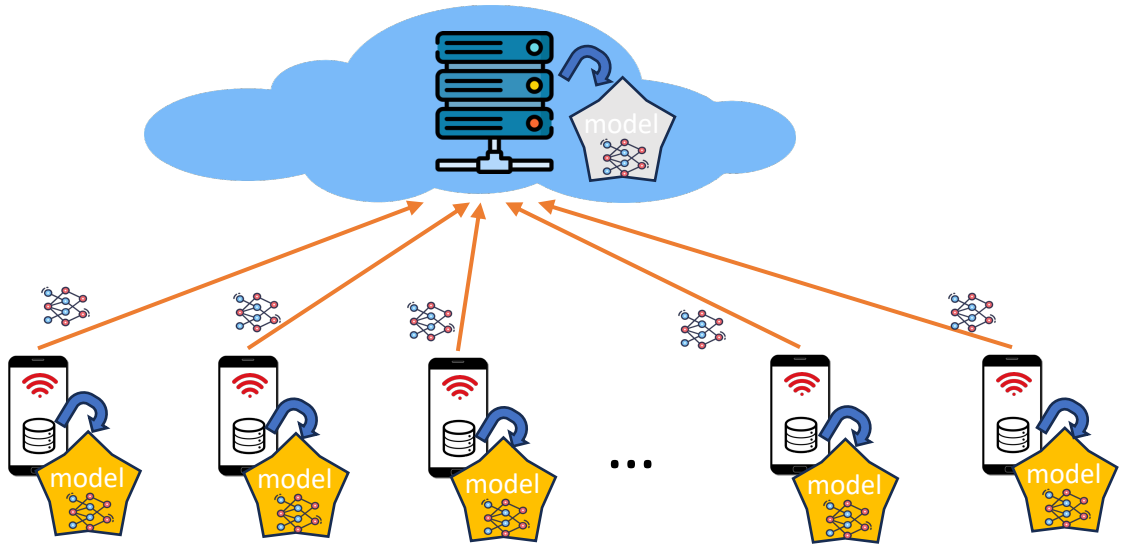
Image credit: Qammar et al. "Federated learning attack surface: taxonomy, cyber defences, challenges, and future directions." Artificial Intelligence Review, 2022.

Naive Aggregation

Server side

…

- For each client $k \in S_r$ do
  - $\theta_k^{(r+1)} \leftarrow \text{ClientUpdate}(k, \theta_k^{(r)})$
- $\theta^{(r+1)} \leftarrow \sum_k p_k \theta_k^{(r)}$



- Clients need to send the model weights to server.
- With exposure to model weights, adversaries can attack the model using **membership inference attacks** and **model extraction attacks** (as discussed in the privacy lecture).

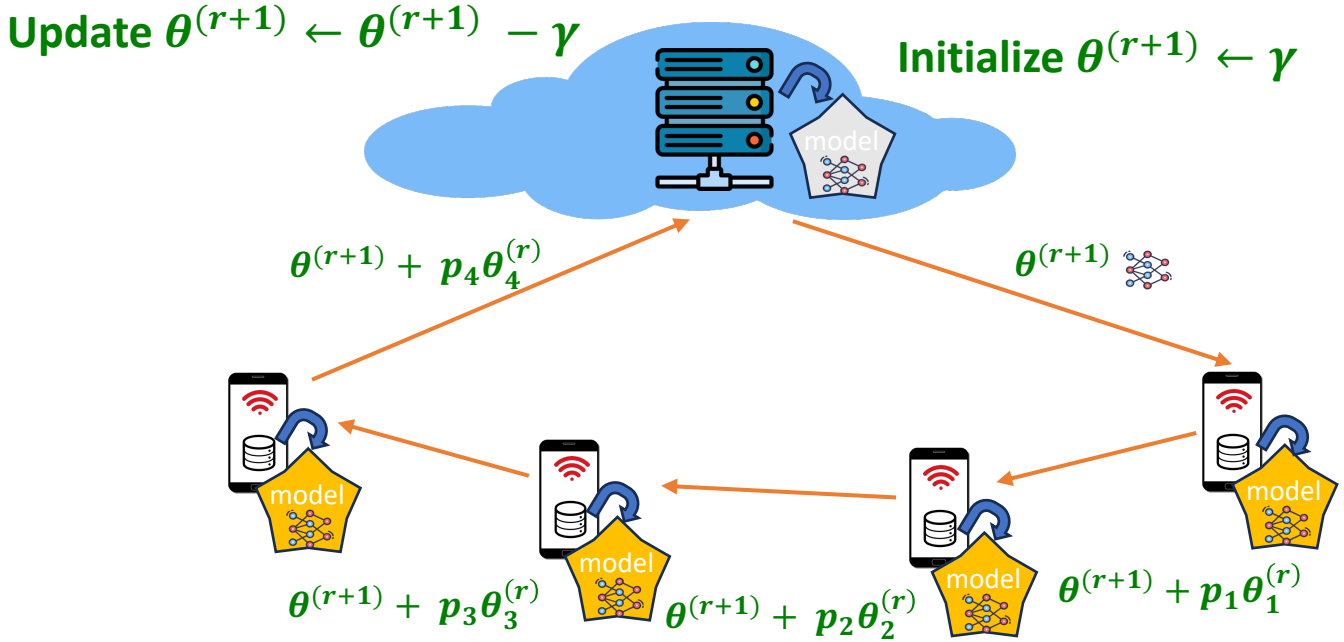
How to participate in federated networks without explicitly sharing our own model weights?

Secure Aggregation

Server side

…

- **Initialize aggregated model $\theta^{(r+1)} \leftarrow \gamma$ random weights**
- For each client $k \in \mathcal{S}_r$ do ~~(in parallel)~~
  - **Send $\theta^{(r+1)}$ to client k**
  - **Client updates $\theta^{(r+1)} \leftarrow \theta^{(r+1)} + p_k \theta_k^{(r+1)}$ locally**
  - **Client sends $\theta^{(r+1)}$ to the next client**
- **Update model with $\theta^{(r+1)} \leftarrow \theta^{(r+1)} - \gamma$**



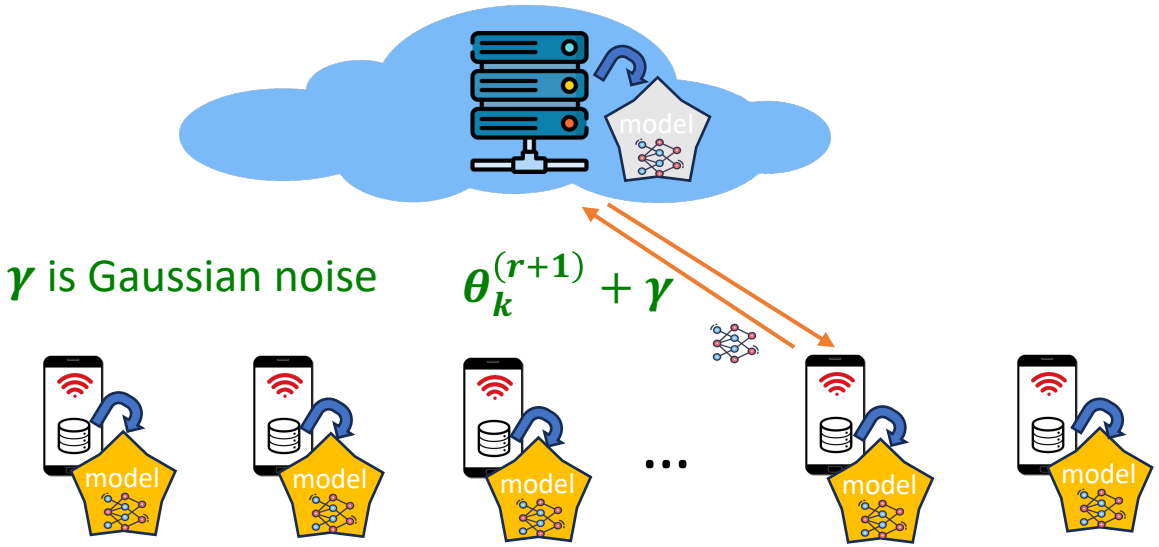
- **Drawback**: we cannot aggregate in parallel anymore
- **Solutions**: Using more advanced communication protocols
  - Pairwise Diffie-Hellman Key Agreement (further reading)

Differential Privacy Aggregation

Server side

…

- For each client $k \in \mathcal{S}_r$ do
  - $\theta_k^{(r+1)} \leftarrow \text{ClientUpdate}(k, \theta_k^{(r)})$
- $\boldsymbol{\theta^{(r+1)} \leftarrow \sum_k p_k \theta_k^{(r+1)}}$ ⟸ **Inject DP constraint**



- **Solution**: Each client adds a Gaussian noise $\boldsymbol{\gamma}$ to $\boldsymbol{\theta_k^{(r+1)}}$ (similar to DP-ERM) before sending it back to the server.
- **Drawbacks**: the resulting average has poor accuracy unless the number of clients (M) is very large. Given a fixed privacy guarantee, the performance gap with centralized settings is $\mathcal{O}(\sqrt{M})$
- **Improvements**:
  - Secure aggregation and secure shuffling can close the gap.
  - Gossip Noise for Private Averaging: Let devices communinate and share pairwise-cancelling Gaussian noise

Summary

- Federated learning (FL) is the technique of **learning ML models from decentralized data**.
- FL is **highly interdisciplinary**, drawing upon ML, computational optimization, privacy, security, networking, systems, and hardware technology.
- **FedAvg with local updates can significantly reduce communication rounds** needed for convergence.
- A primary benefit of FL is the **bolstering of data privacy and security** as sensitive data remains on the user's device.
- This is **an active research topic** with many ongoing challenges, inspiring the publication of hundreds of papers annually to address these issues.