# To the Winner Go the Spoils: Decentralized Machine Learning and the Fair Rewarding of Participation.

Arno Geimer

University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

# Federated Learning: *Privacy-Preserving Machine Learning at Scale*.

**Part 1**: Why Federated Learning?

Regulatory context, Use cases, Frameworks, and ongoing projects.

**Part 2**: Technical Foundations & Challenges.

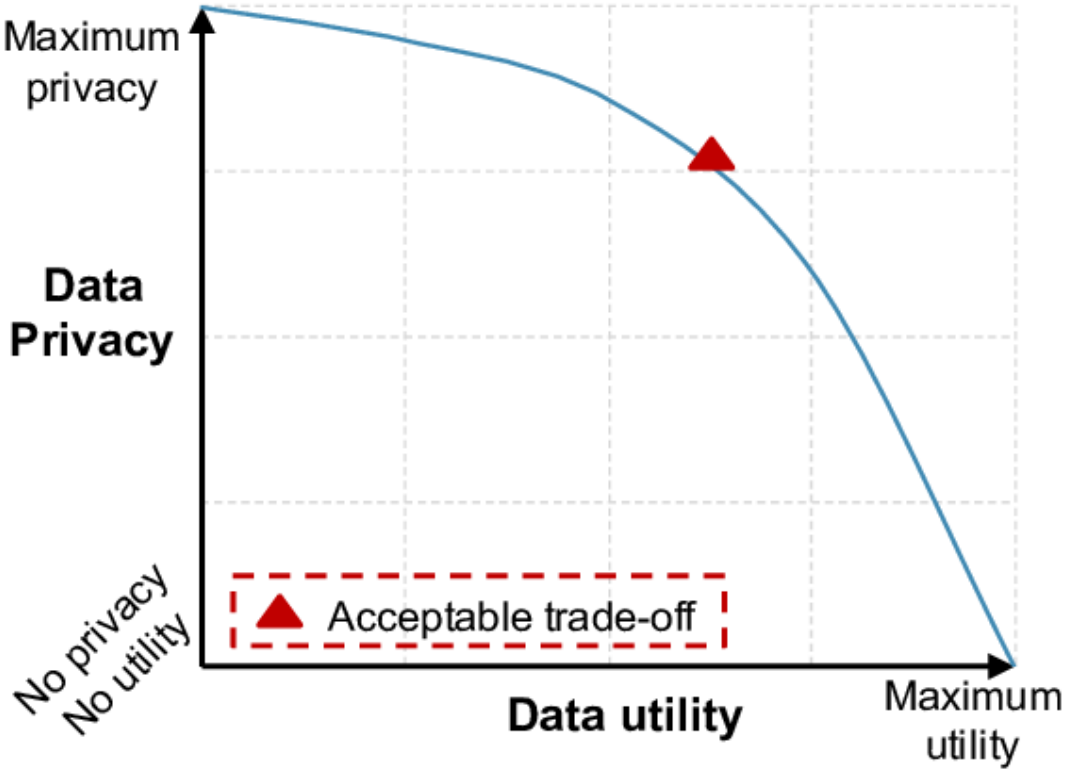
Enabling Federated Learning, improving performance, detecting malicious actors.

**Part 3**: Contribution Assessment.

Existing method, our observations, our approaches.

# Would you trust one central entity with ALL your personal data?

- More utility → weaker privacy guarantees.
- More privacy → less information for the model.
- How can we gain as much information while keeping data (relatively) secure?



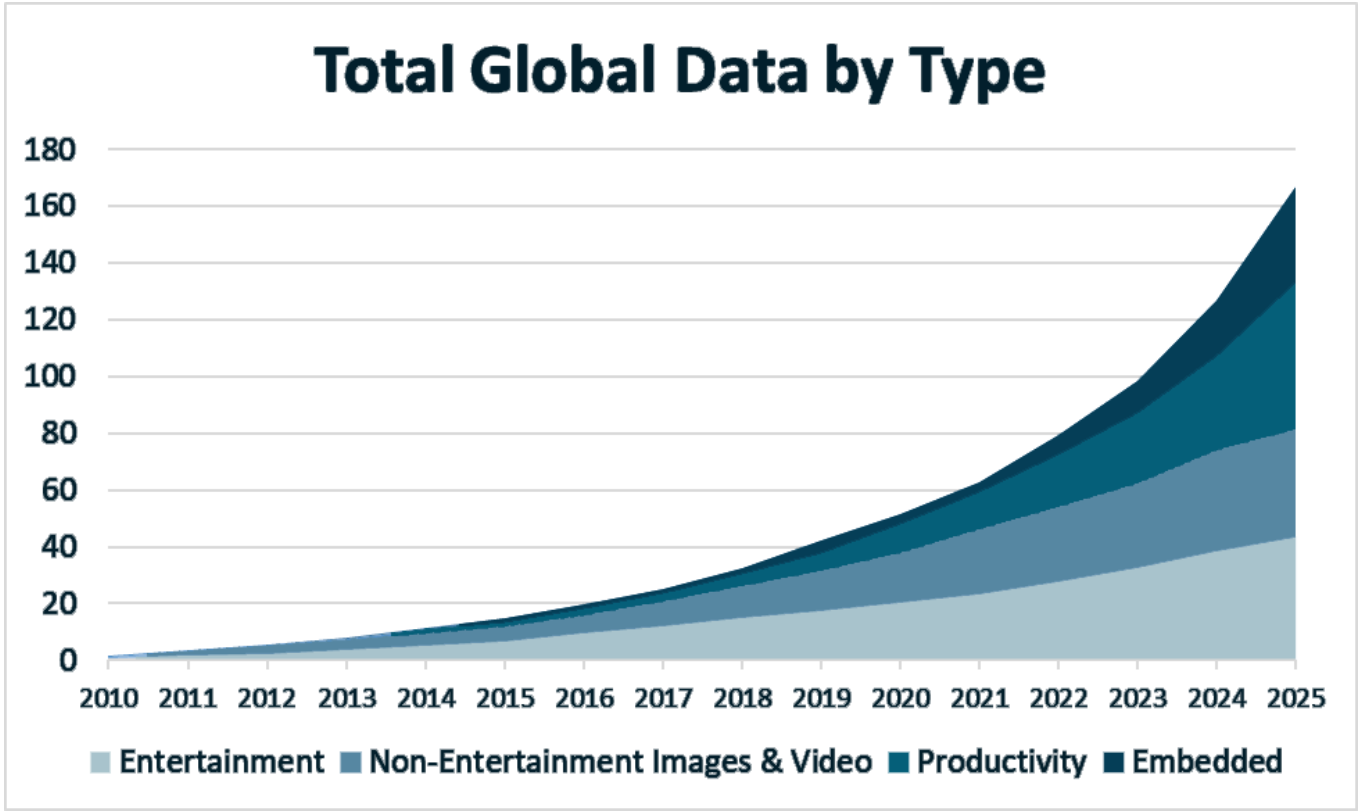

Liu, Fan, Zhiyong Cheng, Huilin Chen, Yinwei Wei, Liqiang Nie, and Mohan Kankanhalli. "Privacy-preserving synthetic data generation for recommendation systems." In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1379-1389. 2022.

# The Regulatory Context

- Mid-2010s: Surge in strict data-protection regulations following privacy awareness.
- Regulations give user control over personal data sale/use, the right to be forgotten and limit data transfer.
- As a result, data can no longer freely move. Companies need to adapt their data collection processes.

# The Exponential Growth of Data

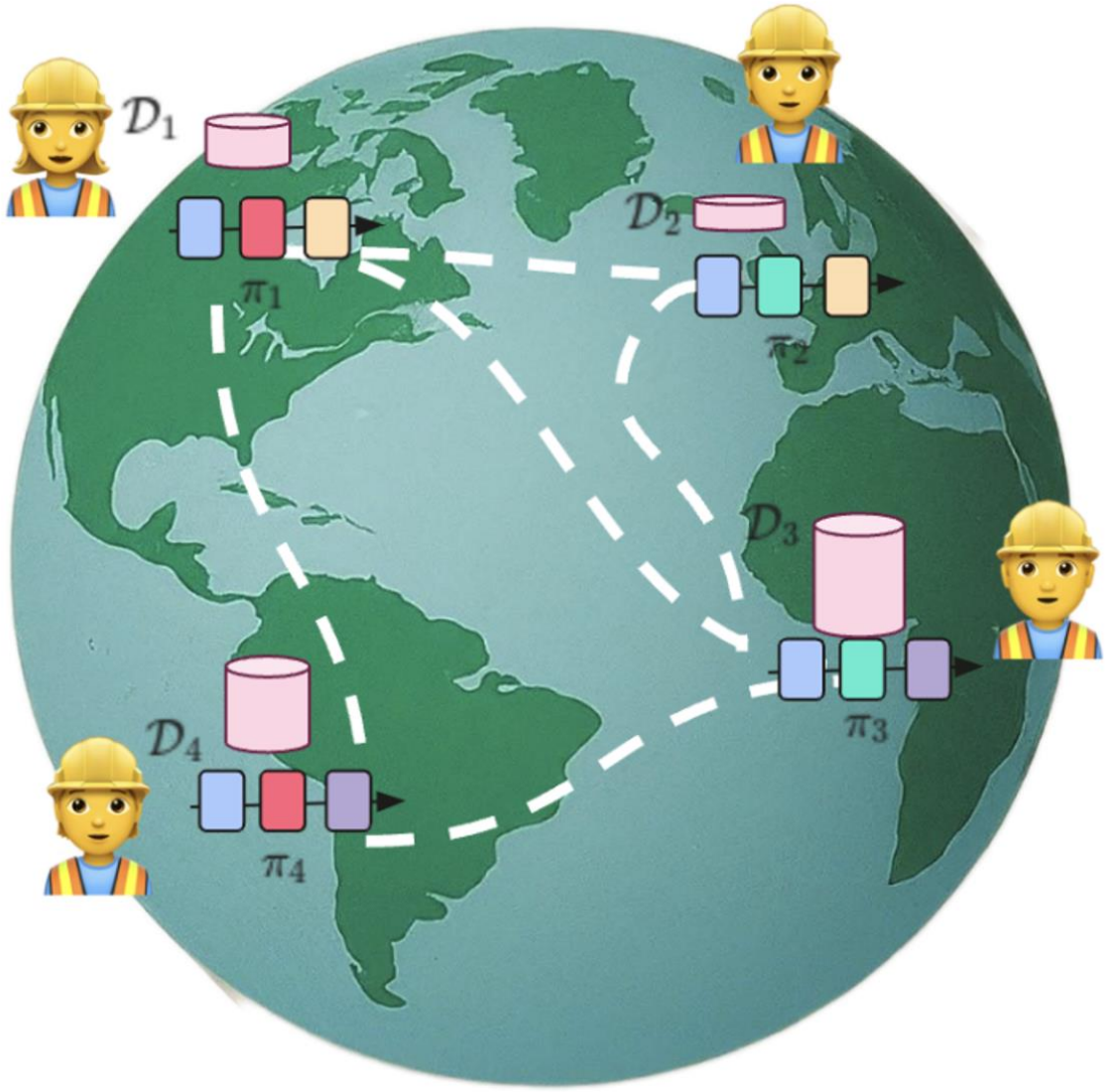
- Global data creation is doubling approximately every 2 years.
- It comes increasingly from personal and distributed devices.
- Infeasible to collect all data in a single storage location.



# What is Federated Learning?

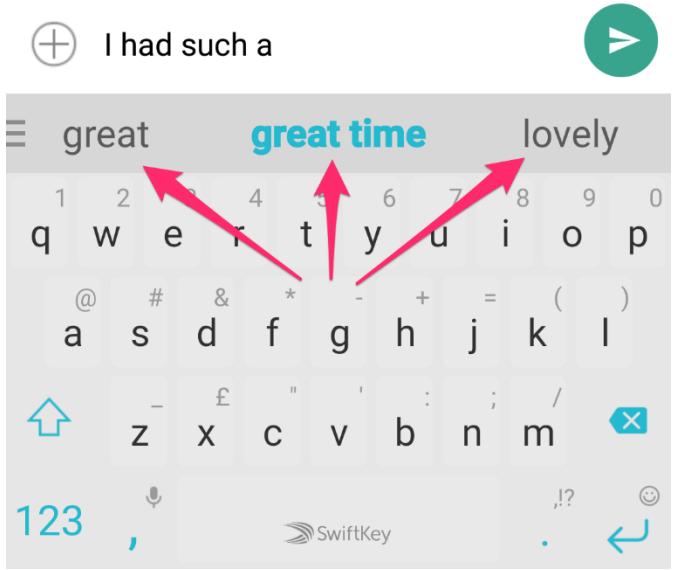
*“Bring code to the data, not data to the code”*

- Video
- A central server manages a collection of distributed data owners.
- All data stays local, all training is local.
- The result is a common model which contains information from all data owners.

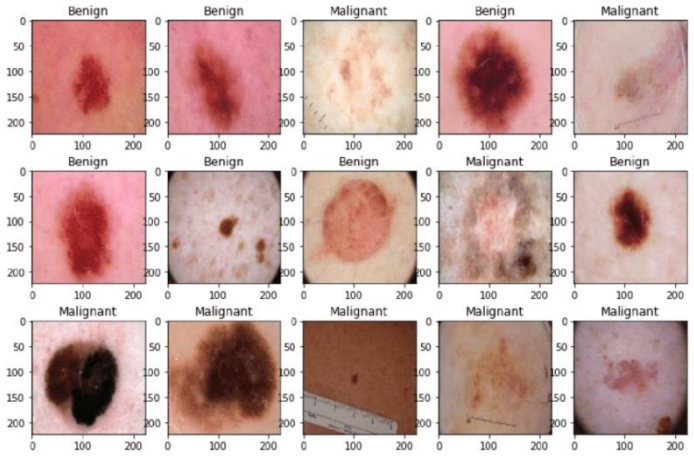


# Key Use Cases

- Mobile applications.
- Medical Applications.
- Fraud Prevention.
- Decentralized LLM finetuning.

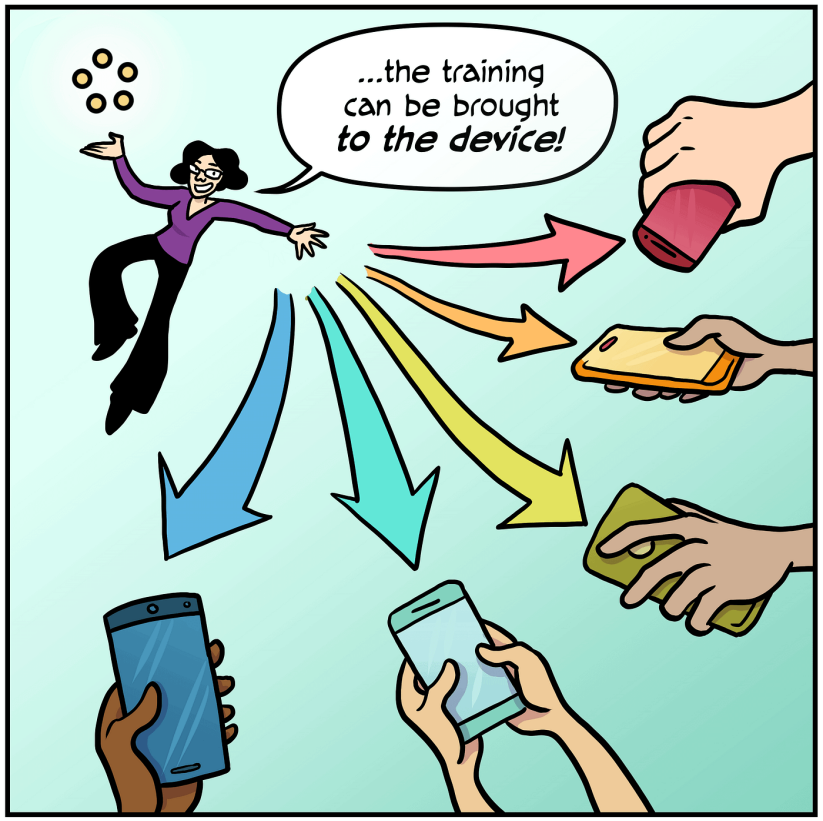







# Mobile Applications: Google Gboard, Apple Quicktype, Siri

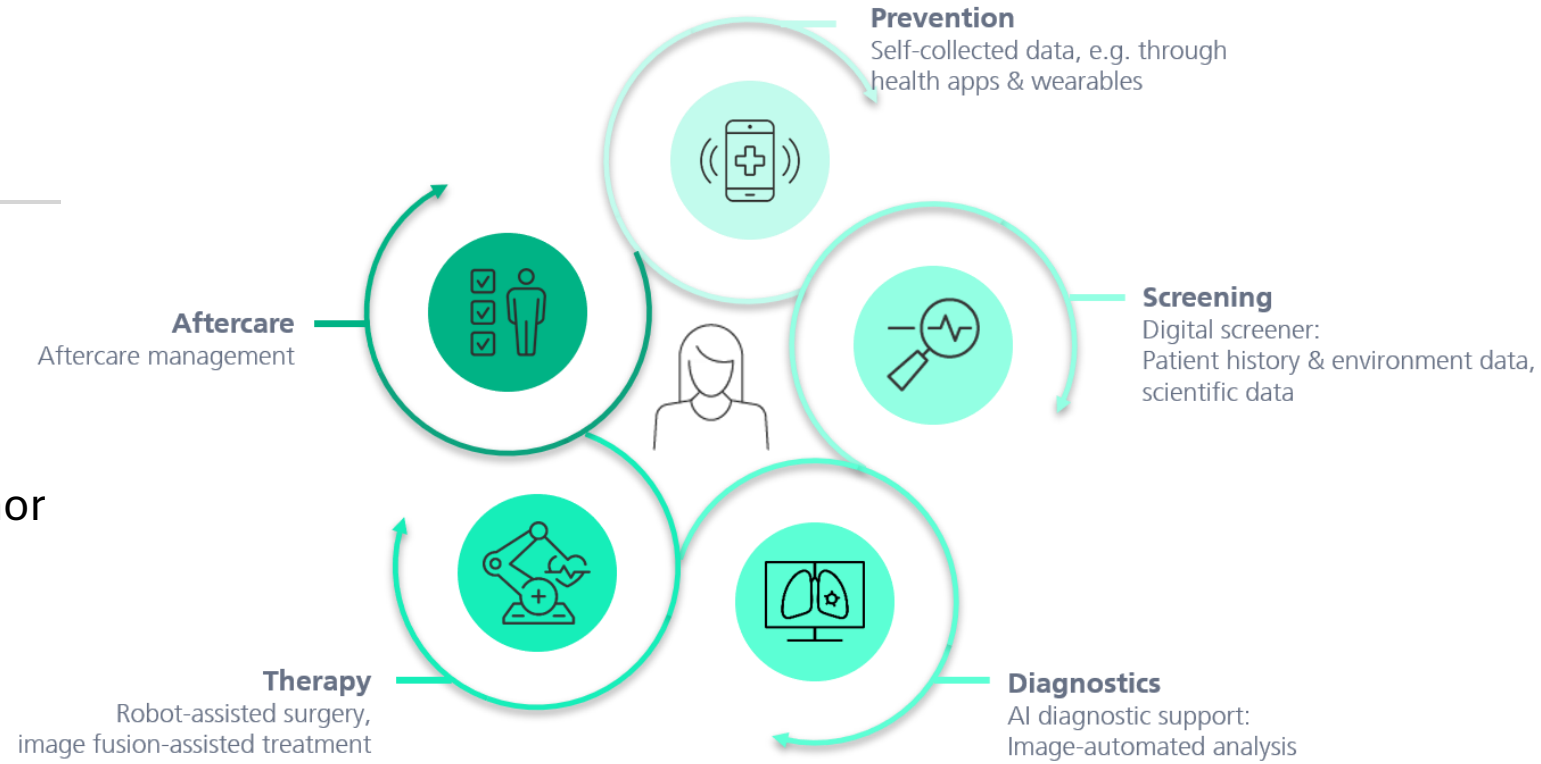
- **Devices**: billions of smartphones worldwide.
- **Use cases**: next-word prediction, emoji suggestions, personalized speech.
- No need to send personal messages and voice recordings to a central server.
- Frequent and diverse updates.




https://federated.withgoogle.com/

# Healthcare: Medical Image Analysis

- **Devices**: hospital servers, imaging machines
- **Use cases**: diagnostic models for tumor detection, rare disease detection.
- Increased statistical power.
- GDPR compliance.



# Large-Scale AI: Decentralized LLM Training

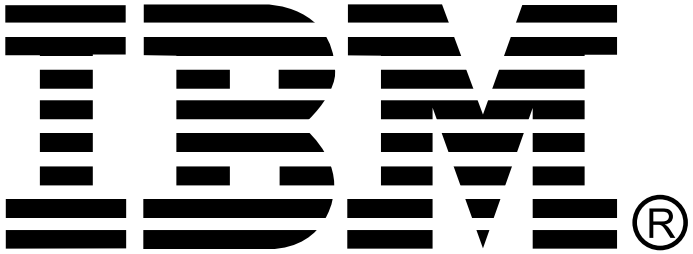
- **Devices**: Clusters, private data centers, cloud.
- **Use cases**: Train or fine-tune large language models across organizations.
- Proprietary data stays local .
- Shared foundation model improved without data leakage.

WeBank

NVIDIA

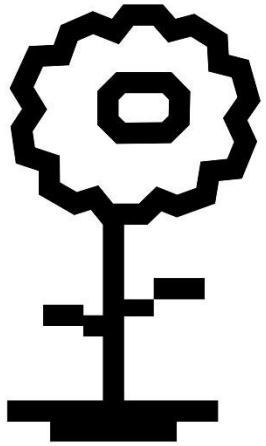
# Finance: Fraud Prevention

- **Devices**: Secure servers in banks, fintech systems, transaction endpoints.
- **Use cases**: Fraud detection models, suspicious behavior pattern identification.
- Protects sensitive financial records.
- Enhances detection by sharing insights across silos.






# Frameworks



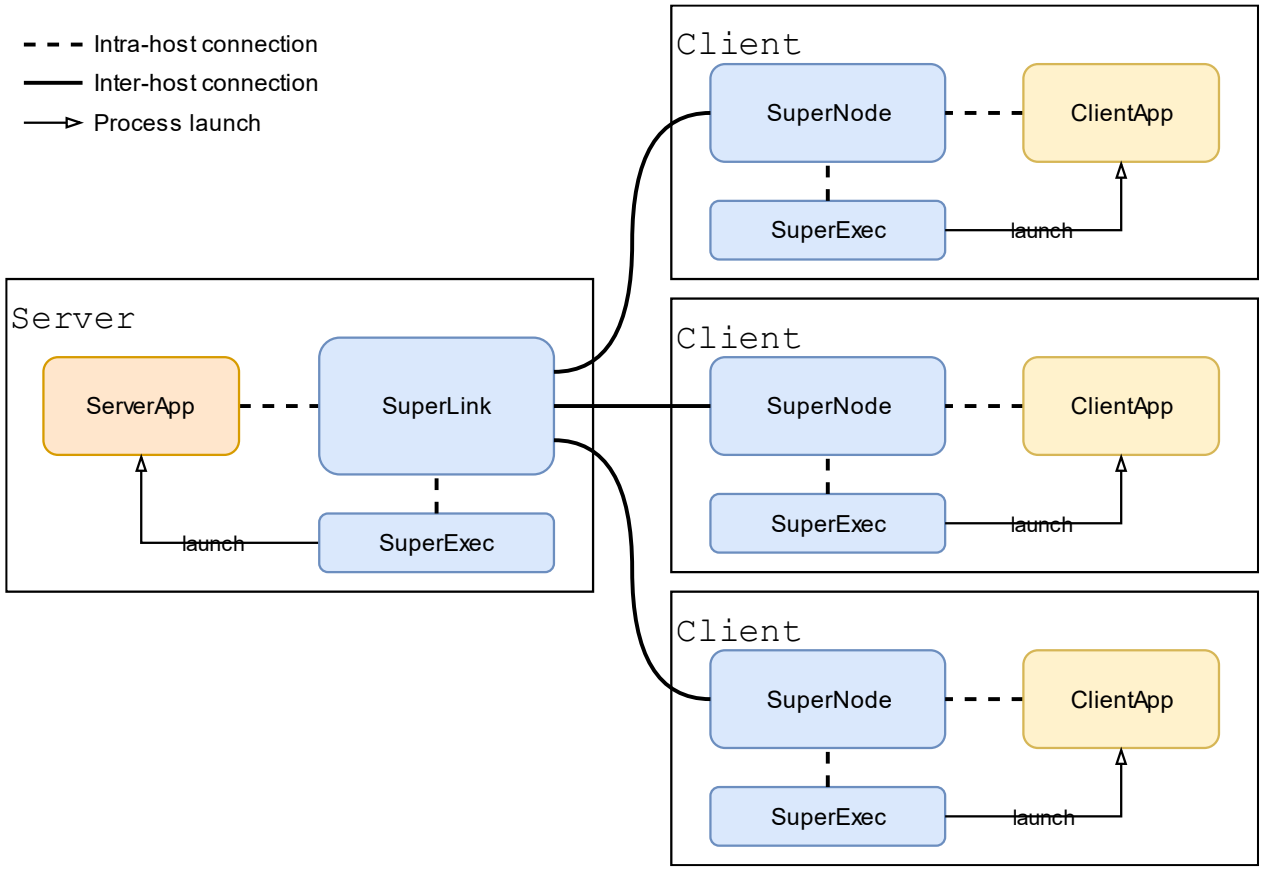
Flower: A Friendly Federated Learning Framework


- **Flower**: Open-source, lightweight, cross-platform.
- **Nvidia Clara FL**: Closed-source, healthcare-focused.
- **FATE** (WeBank): Open-source, strong security module integration.
- **TensorFlow Federated**: Open-source, research and prototyping library.


# The Flower framework

Flower: A Friendly Federated Learning Framework

- Clients and the Server are launched as separate instances.
- Communication through ray.
- Easy to deploy, run on different machines.



# Federated Learning: *Privacy-Preserving Machine Learning at Scale.*

**Part 1**: Why Federated Learning?

Regulatory context, Use cases, Frameworks, and ongoing projects.

**Part 2**: Technical Foundations & Challenges.

Enabling Federated Learning, improving performance, detecting malicious actors.

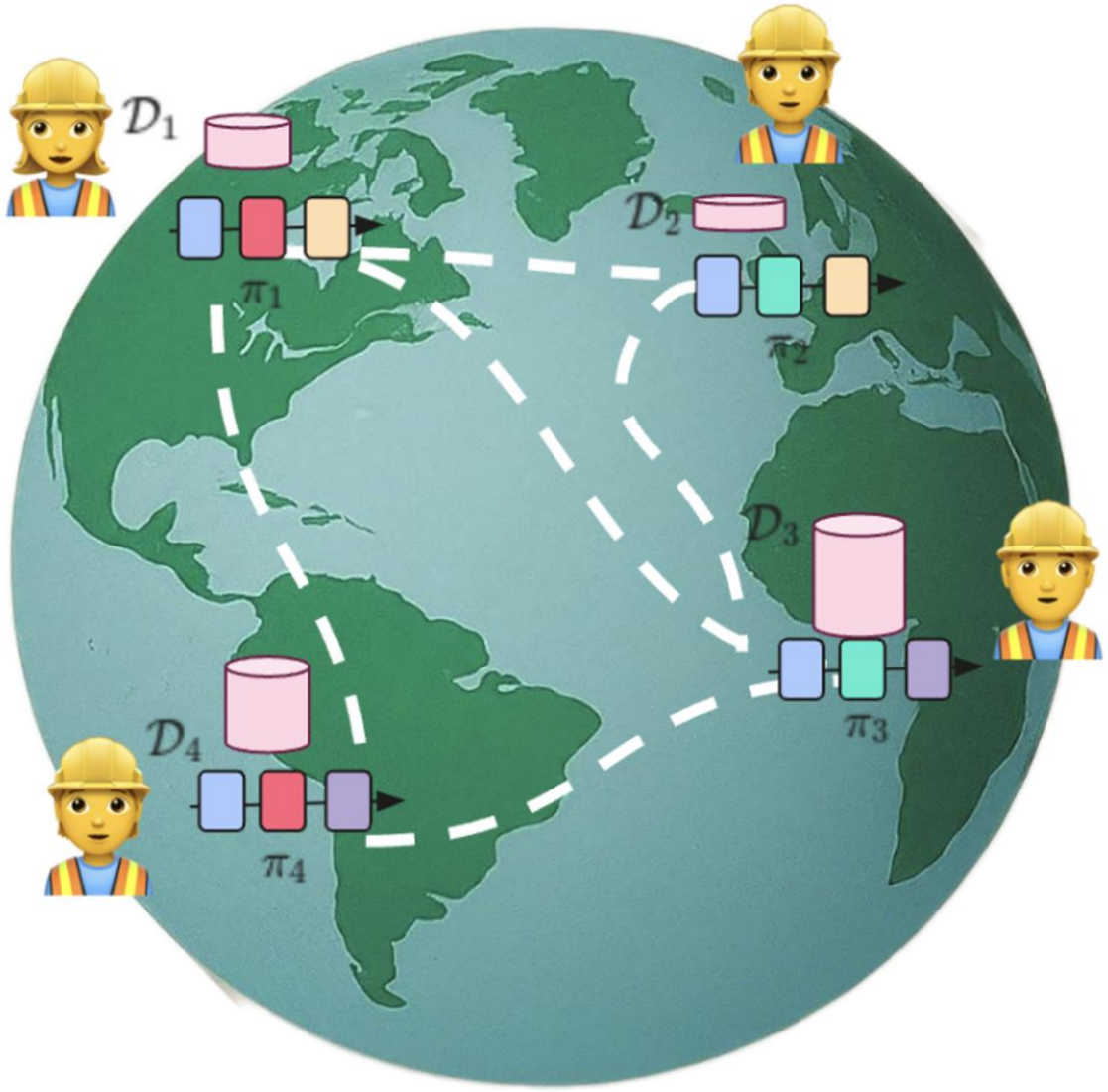
**Part 3**: Contribution Assessment.

Existing method, our observations, our approaches.

# What is Federated Learning?

*“Bring code to the data, not data to the code”*

- Video
- A central server manages a collection of distributed data owners.
- All data stays local, all training is local.
- The result is a common model which contains information from all data owners.



# Federated Averaging (FedAvg).

- Introduced by McMahan et al., 2017, Google.
- Repetitive loop between **server** and **clients**.
- Each client trains locally on its data.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273-1282. PMLR, 2017.

## The Server–Client Loop.

- Each round, the server sends the most up-to-date global model checkpoint to a random set of clients.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## The Server–Client Loop.

- Each round, the server sends the most up-to-date global model checkpoint to a random set of clients.
- Clients continue training from this checkpoint.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## The Server–Client Loop.

- Each round, the server sends the most up-to-date global model checkpoint to a random set of clients.
- Clients continue training from this checkpoint. They send their updated models back to the server.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## The Server–Client Loop.

- Each round, the server sends the most up-to-date global model checkpoint to a random set of clients.
- Clients continue training from this checkpoint. They send their updated models back to the server.
- The central server generates a new global model checkpoint based on all received updates in this round.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

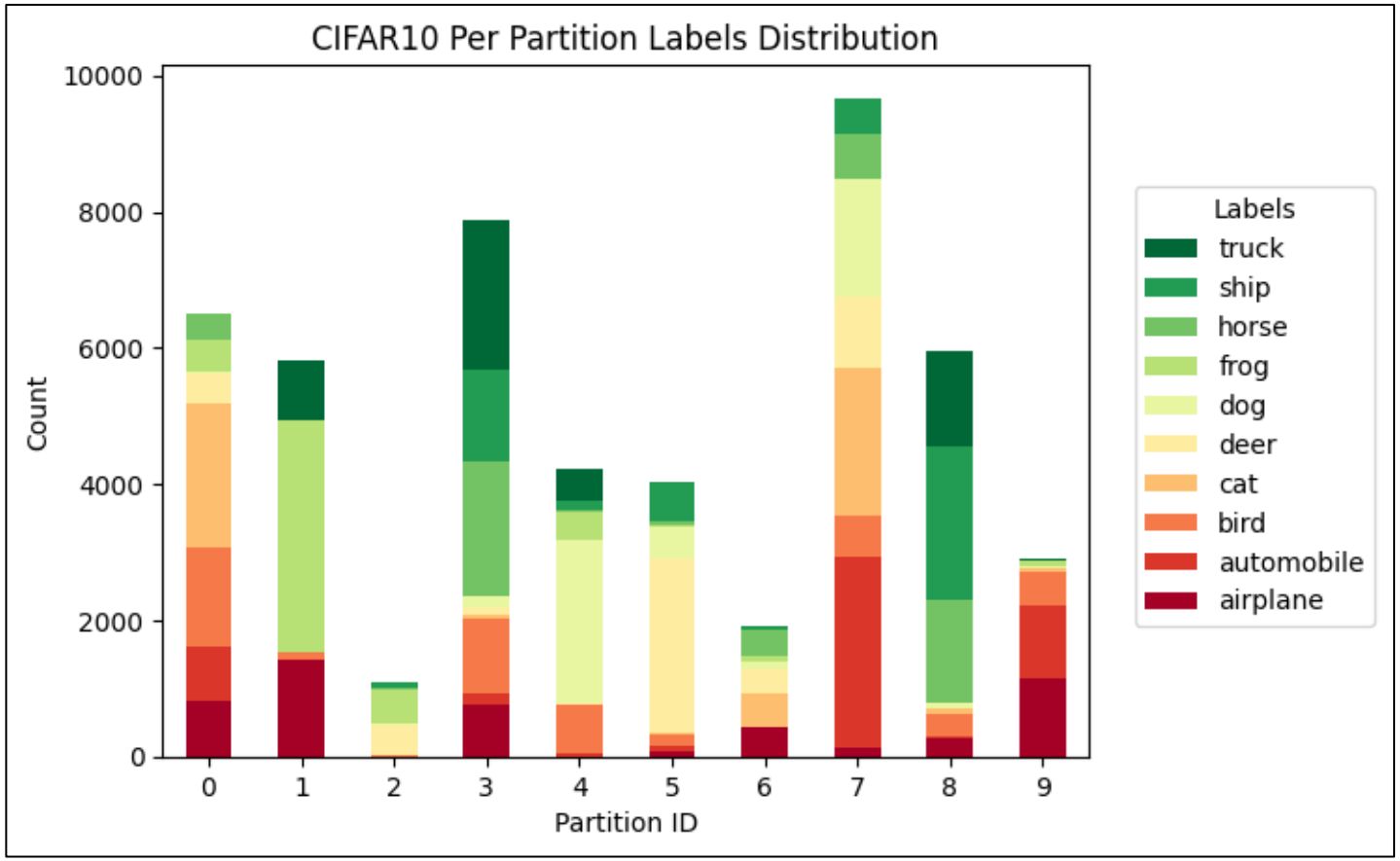
**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Key challenges in Federated Learning.

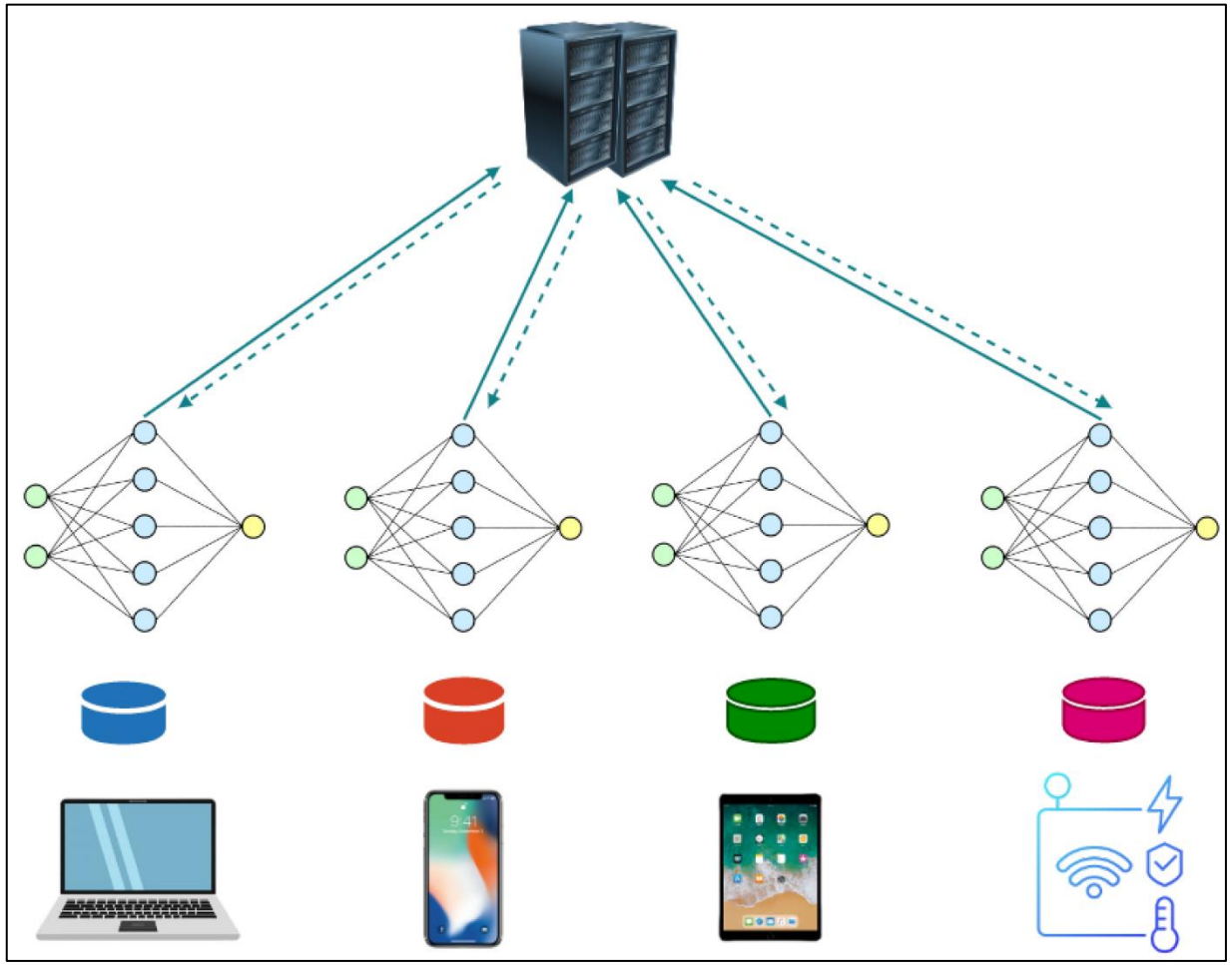
- **Data heterogeneity** in clients.




https://flower.ai/blog/2024-07-08-announcing-flower-datasets-0.2.0

# Key challenges in Federated Learning.

- **Data heterogeneity** in clients.
- Client **hardware heterogeneity**.



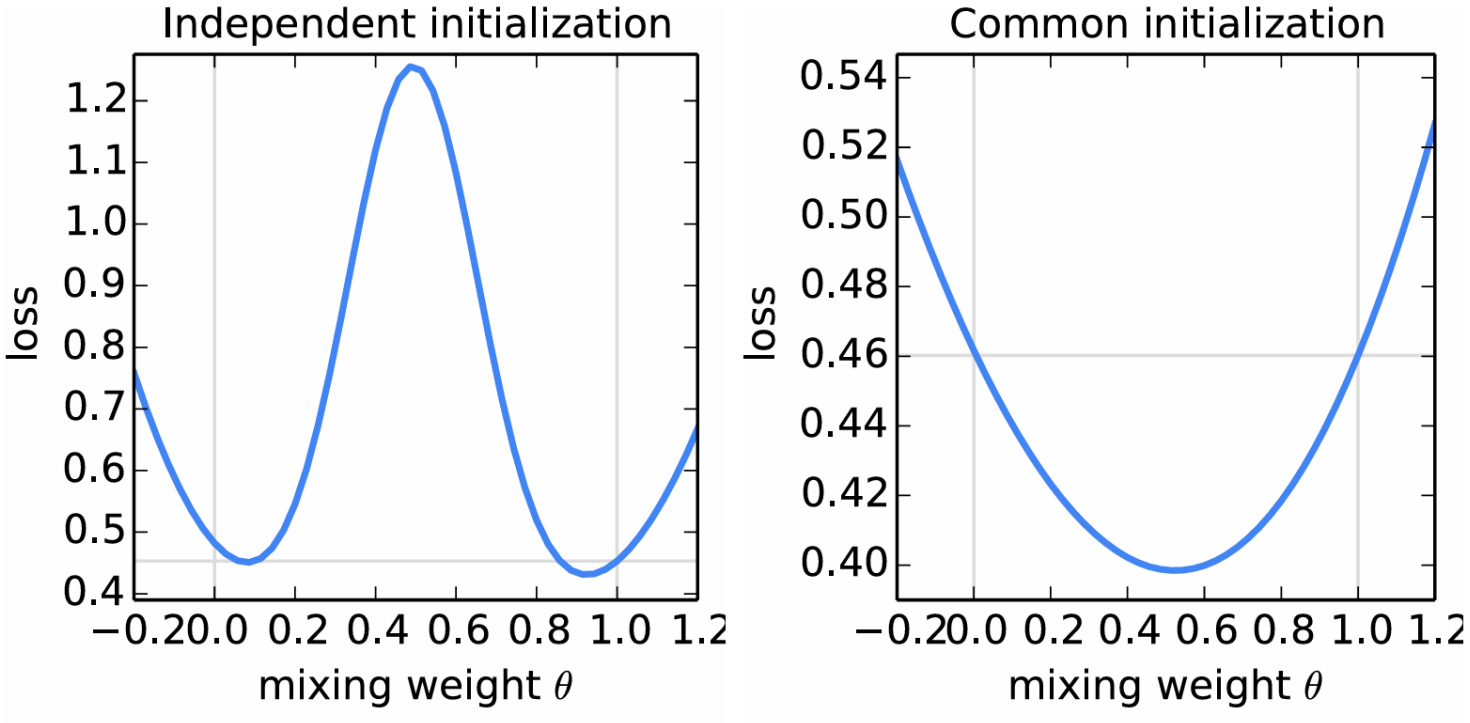
Barona López, L. I., & Borja Saltos, T. (2025). Heterogeneity challenges of federated learning for future wireless communication networks. *Journal of Sensor and Actuator Networks*, *14*(2), 37.

# Key challenges in Federated Learning.

- **Data heterogeneity** in clients.
- Client **hardware heterogeneity**.
- Model update **security**.

# Why Repetitive Training?

- Combining models trained from different initial conditions has been shown to produce an arbitrarily bad model.
- Gradual alignment leads to stable convergence and allows adaptation to new data.
- **Communication–accuracy trade-off**: Sending model updates is not free (time + resources).




Loss of the model generated as θw + (1-θ)w' on MNIST. Notice the y-axis scales. Common initialization produces a model which outperforms both parent models.

McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273-1282. PMLR, 2017.

## Key hyperparameters of Federated Averaging.

- Local epochs **E**:
  - High = Less communication, faster training, more drift.
  - Low = More communication, more stable but slower convergence.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Key hyperparameters of Federated Averaging.

- Local epochs **E**:
  - High = Less communication, faster training, more drift.
  - Low = More communication, more stable but slower convergence.

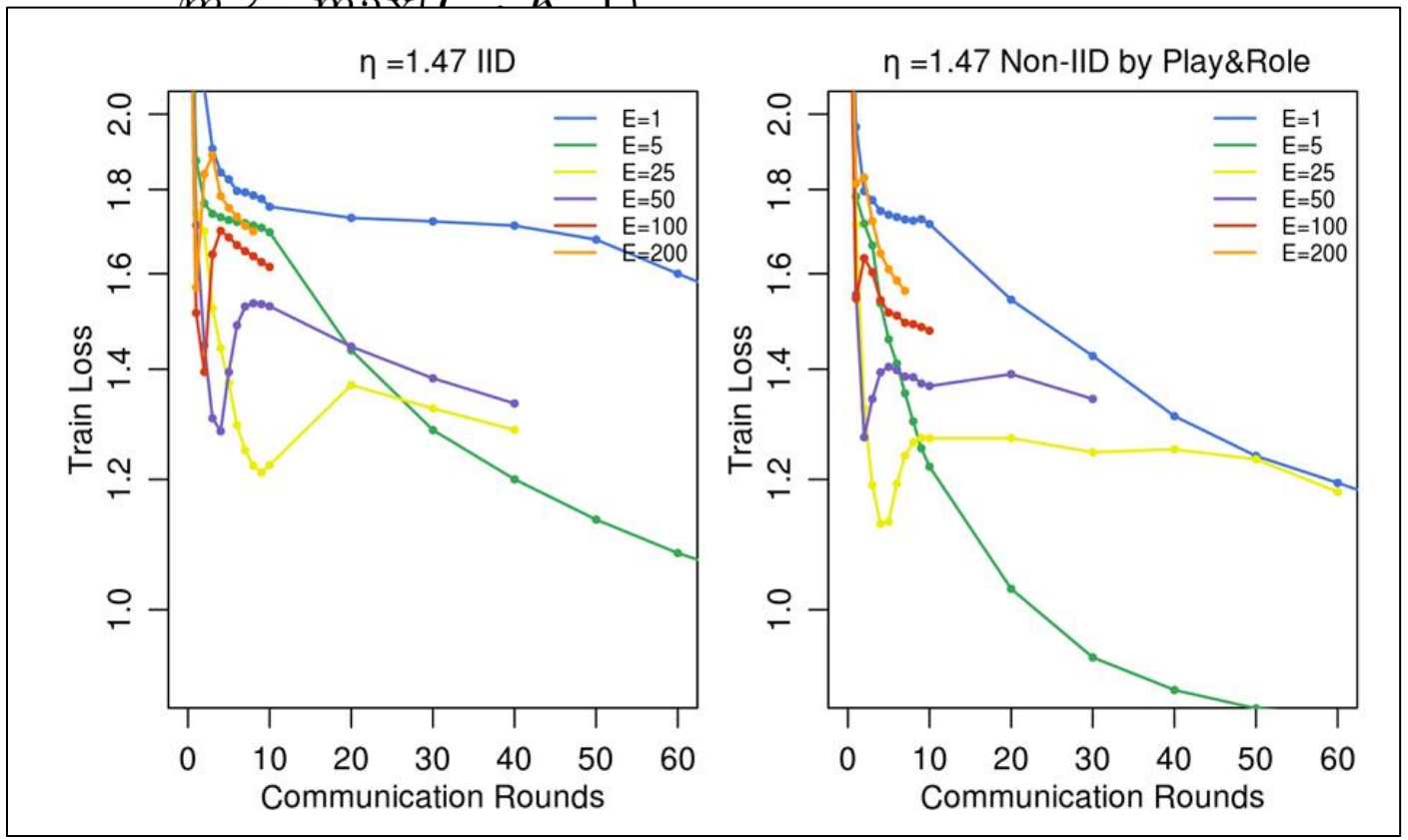
**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$




McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273-1282. PMLR, 2017.

## Key hyperparameters of Federated Averaging.

- Local epochs **E**:
  - High = Less communication, faster training, more drift.
  - Low = More communication, more stable but slower convergence.

- Client selection fraction **C**:
  - High = More diverse data, faster convergence, best possible global model.
  - Low = Faster training, low communication, slow convergence.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## Client Selection.

- Random sampling reduces bias.
- Blindly sampling from all available clients may not always be favorable.
- Alternatives: Clustered sampling, weighted sampling, contribution-based sampling.
- Not all clients might be available (smartphone on/off, plugged in).

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## Server Aggregation.

- FedAvg: weighted mean by dataset size.
- FedOpt: Instead of simple SGD, use adaptive optimization for aggregation.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Server Aggregation.

- FedAvg: weighted mean by dataset size.
- FedOpt: Instead of simple SGD, use adaptive optimization for aggregation.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

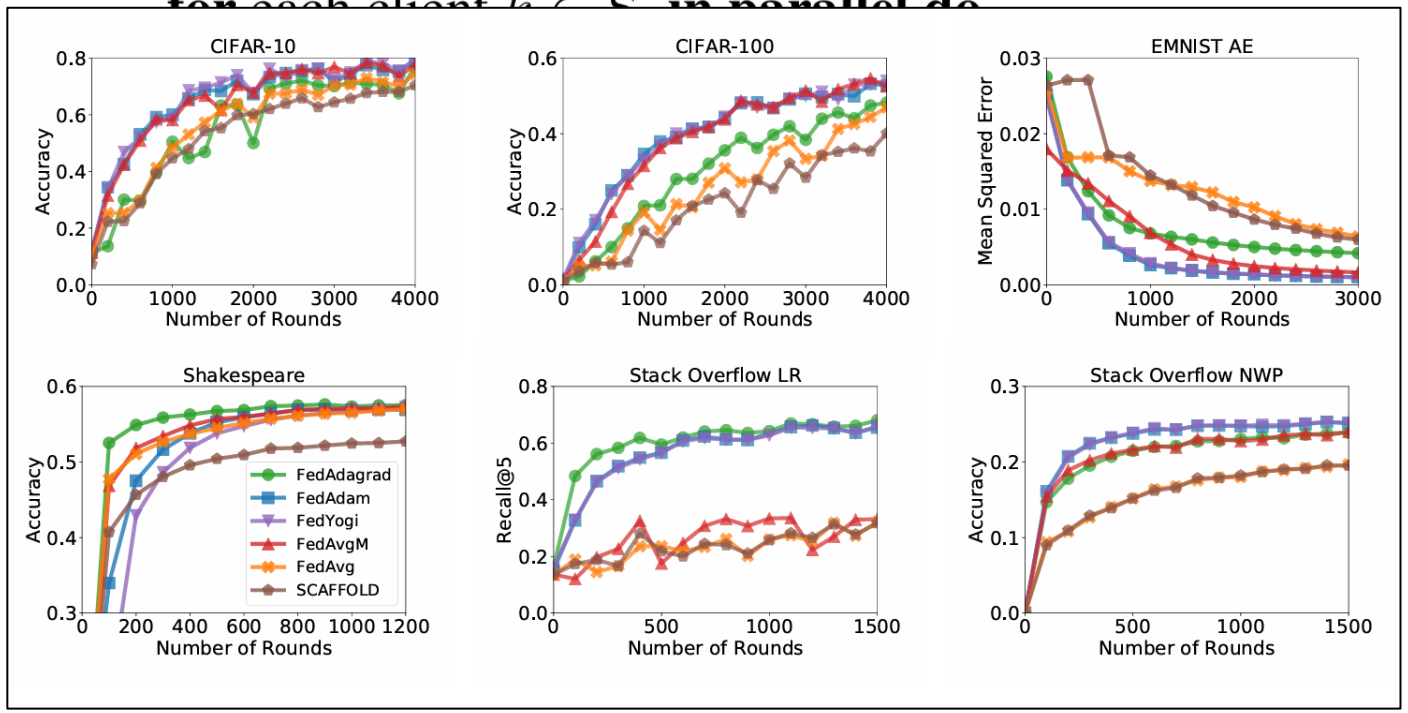
initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**




McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273-1282. PMLR, 2017.

## Server Aggregation.

- FedAvg: weighted mean by dataset size.
- FedOpt: Instead of simple SGD, use adaptive optimization for aggregation.
- Other methods use the median, a trimmed average, or other variance-reducing aggregations.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

## Client Update Strategies.

- FedAvg: All clients run the same local training process.
- FedProx: Partial work and local regularization > Improved performance on heterogeneous data.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

Li, Tian, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." *Proceedings of Machine learning and systems* 2 (2020): 429-450.

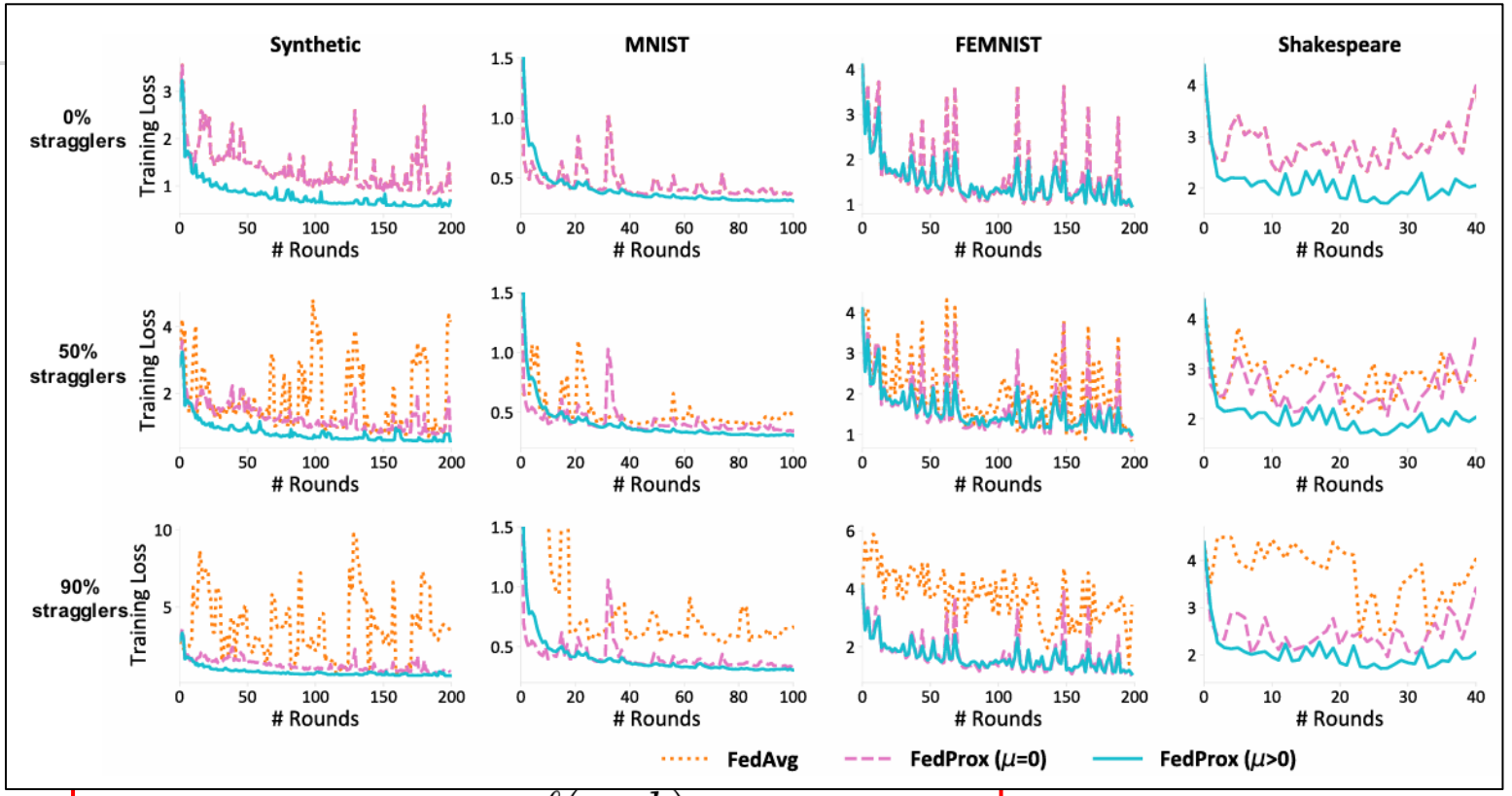
# Client Update Strategies.

- FedAvg: All clients run the same local training process.
- FedProx: Partial work and local regularization > Improved performance on heterogeneous data.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$




$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server


Li, Tian, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." *Proceedings of Machine learning and systems* 2 (2020): 429-450.

## Client Update Strategies.

- FedAvg: All clients run the same local training process.
- FedProx: Partial work and local regularization > Improved performance on heterogeneous data.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

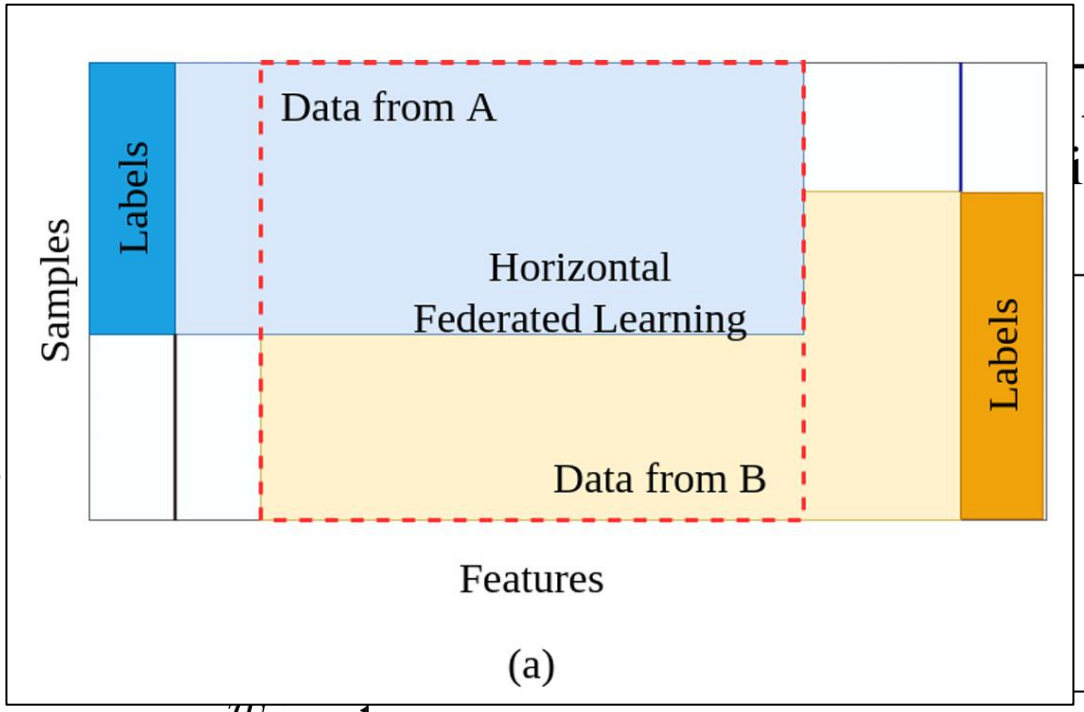
**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Client Update Strategies.

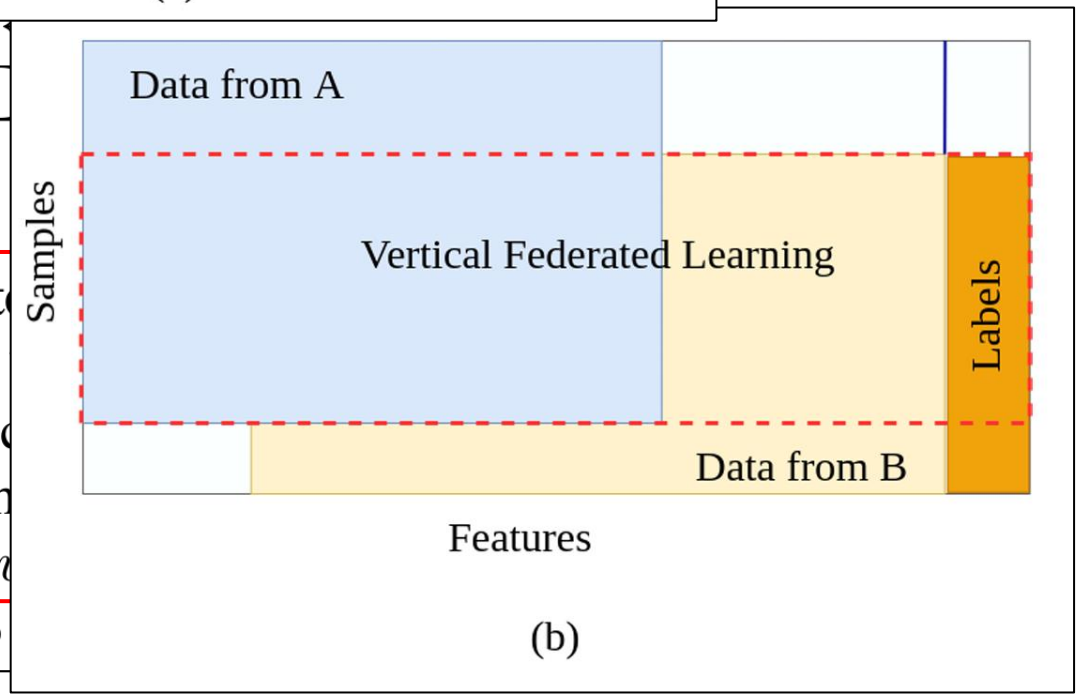
- FedAvg: All clients run the same local training process.
- FedProx: Partial work and local regularization > Improved performance on heterogeneous data.
- Vertical Federated Learning: Different feature spaces in client data > Different local training.




$K$ clients are
is the number

$w_{t+1}$
$m_t \leftarrow \sum$
$w_{t+1} \leftarrow$

**ClientUpdat**
$\mathcal{B} \leftarrow$ (split
**for** each lo
**for** batch
$w \leftarrow$
return $w$ to




Trindade, Silvana, Luiz F. Bittencourt, and Nelson LS da Fonseca. "Management of resource at the network edge for federated learning." *arXiv preprint arXiv:2107.03428* (2021).

## Security in Federated Learning.

- **Eavesdropping**: Client updates can be intercepted, revealing potentially private information.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Security in Federated Learning.

- **Eavesdropping**: Client updates can be intercepted, revealing potentially private information.
- **Counter-measure**: Encryption of model updates.
- Drawback: Computationally expensive.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of l

**Ser**

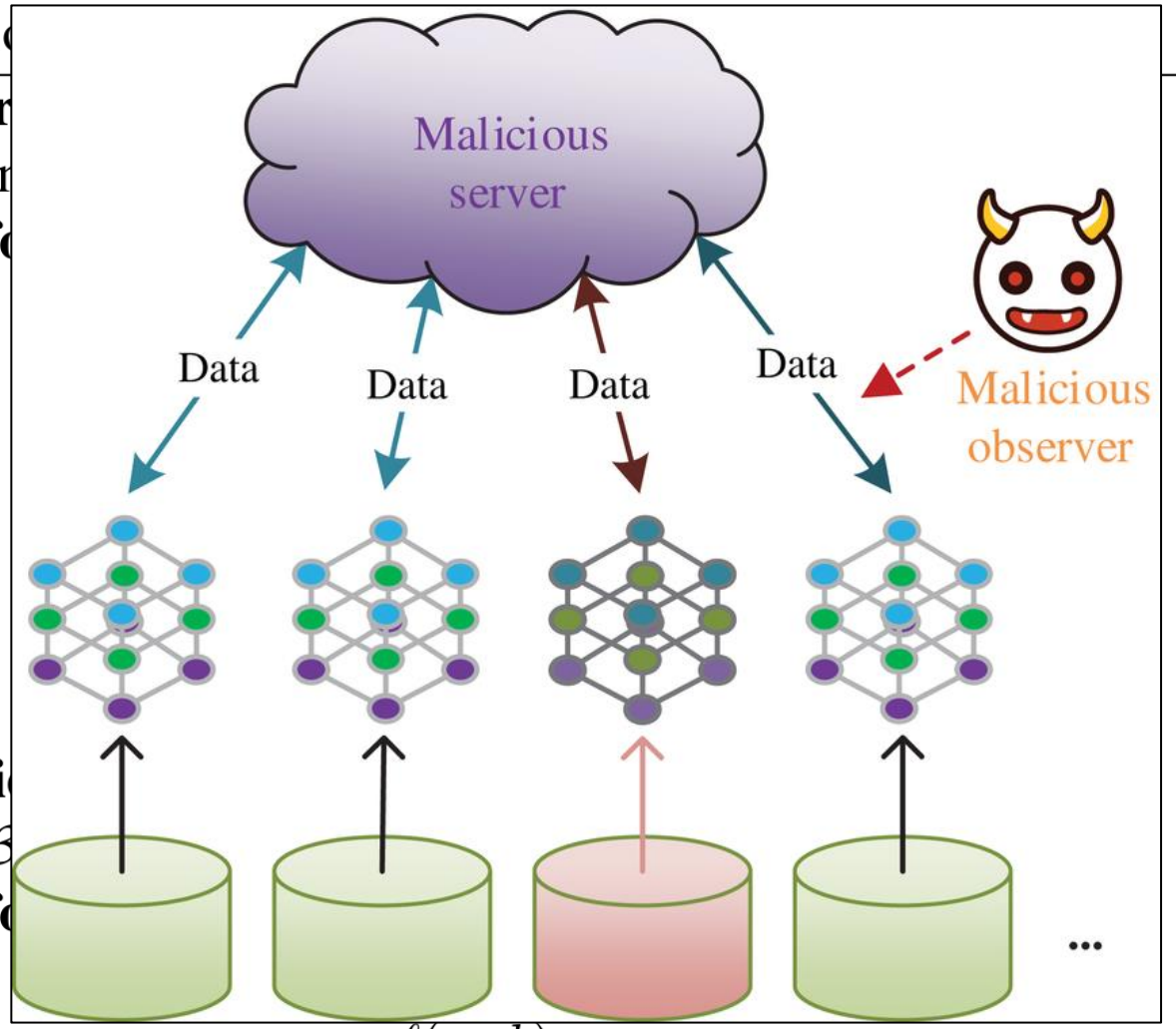
i

**f**

**Cli**

$\mathcal{B}$

**f**




$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

Yu, Bin, Wenjie Mao, Yihan Lv, Chen Zhang, and Yu Xie. "A survey on federated learning in data mining." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 12, no. 1 (2022): e1443.

## Security in Federated Learning.

- **Model inversion attack**: Reconstruct training data from gradients, test membership of training data.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

# Security in Federated Learning.

- **Model inversion attack**: Reconstruct training data from gradients, test membership of training data.
- **Counter-measure** : Differential privacy, larger batch sizes.
- Drawback: Negative impact on global model performance.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

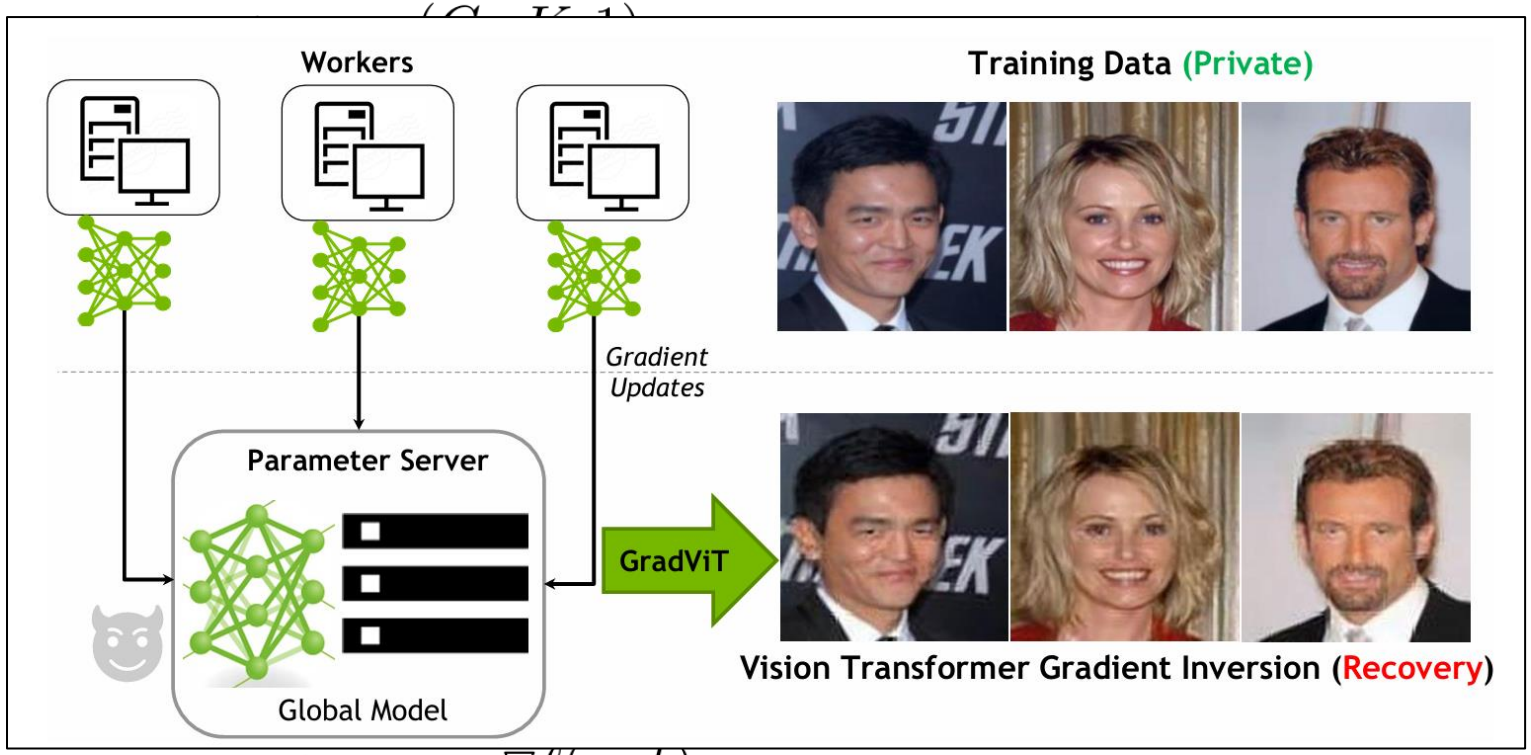
**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server




Hatamizadeh, Ali, Hongxu Yin, Holger R. Roth, Wenqi Li, Jan Kautz, Daguang Xu, and Pavlo Molchanov. "Gradvit: Gradient inversion of vision transformers." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10021-10030. 2022.

## Security in Federated Learning.

- **Model poisoning**: Introduce malicious data to change global model behaviour.

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

initialize $w_0$

**for** each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of $m$ clients)

**for** each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)

**for** each local epoch $i$ from 1 to $E$ **do**

**for** batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return $w$ to server

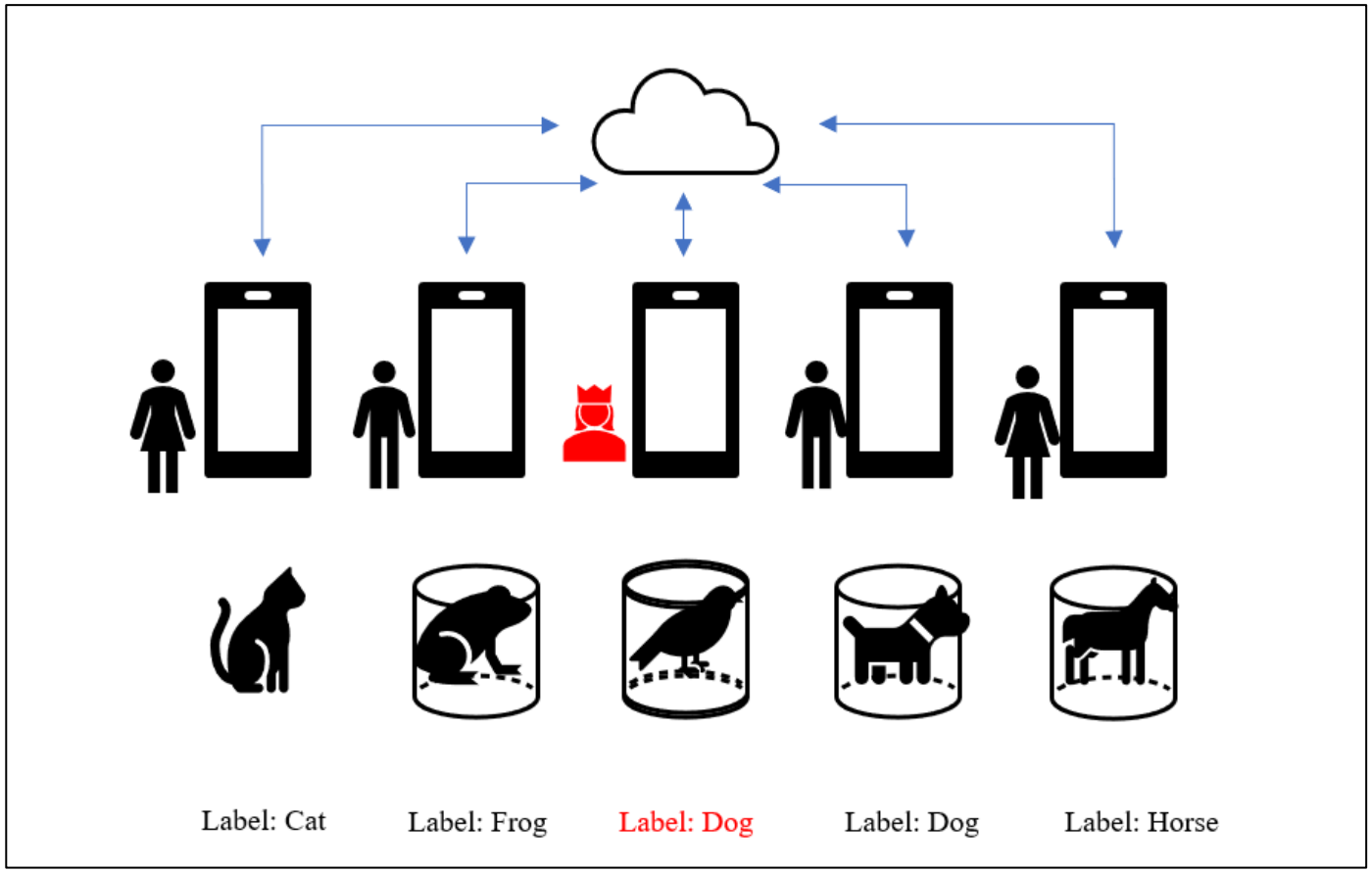
# Security in Federated Learning.

- **Model poisoning**: Introduce malicious data to change global model behaviour.
- **Counter-measure**: Poisoning detection methods, client selection.
- Drawback: False positives, safety vs performance trade-off.

https://github.com/michaelthung-illumina/Label_Flipping_Attack

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**

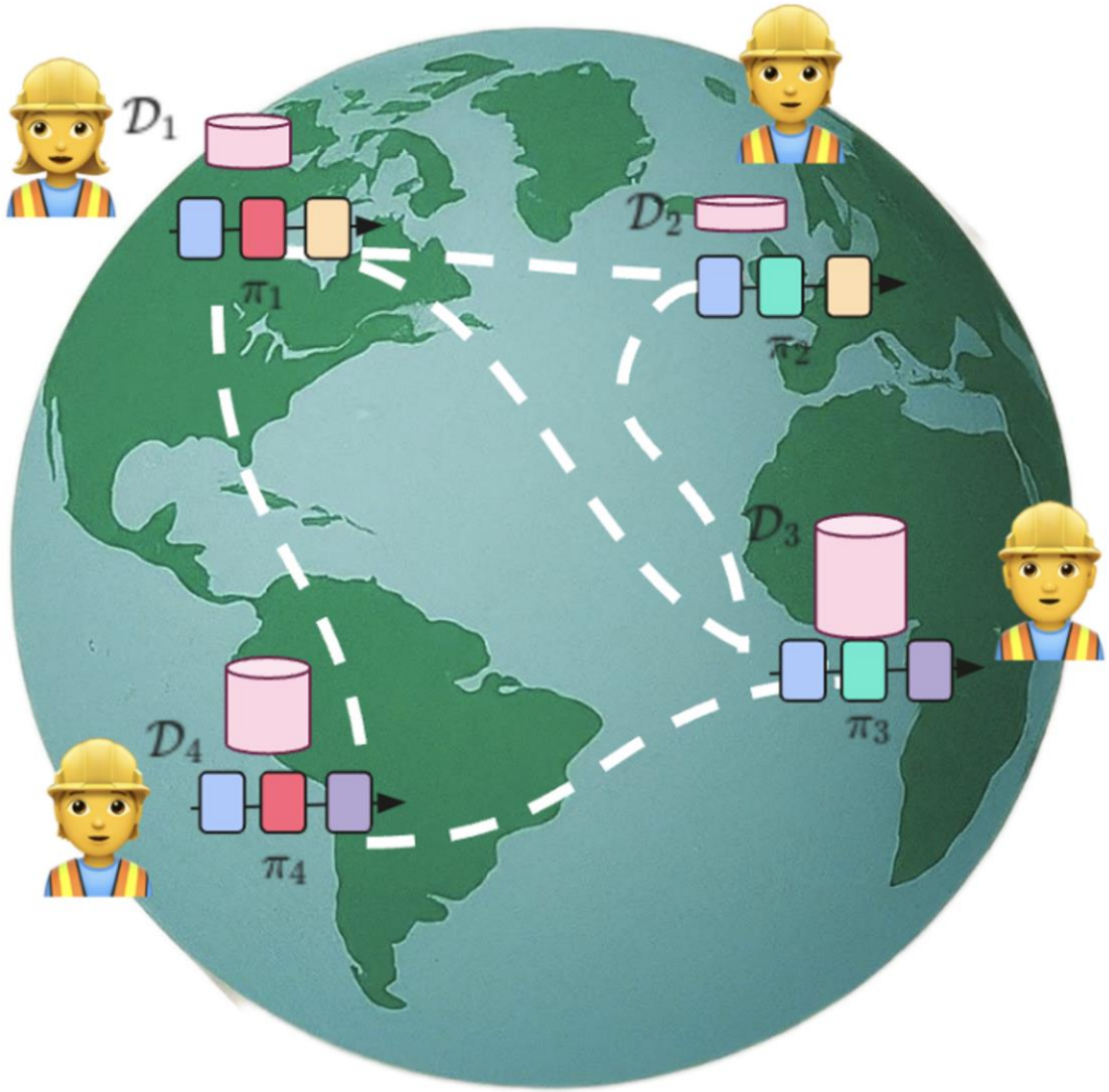



$$w \leftarrow w - \eta \nabla \ell(w; b)$$

return $w$ to server

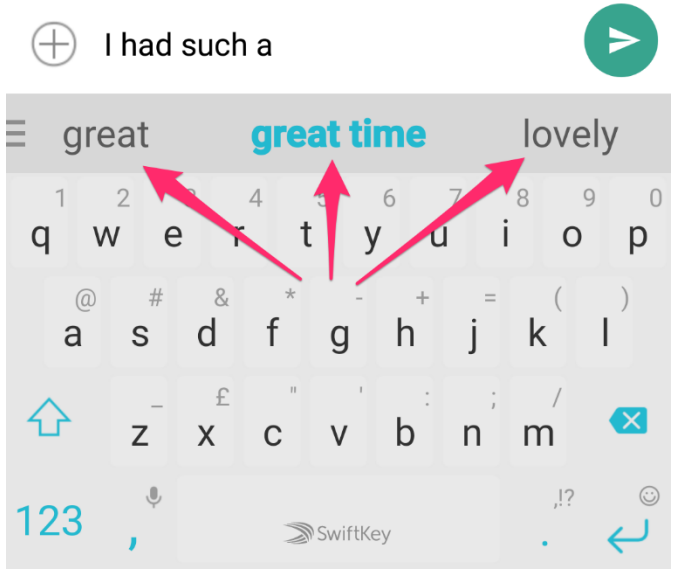
# Federated Learning: Summary.

- Decentralized training on multiple data sources.

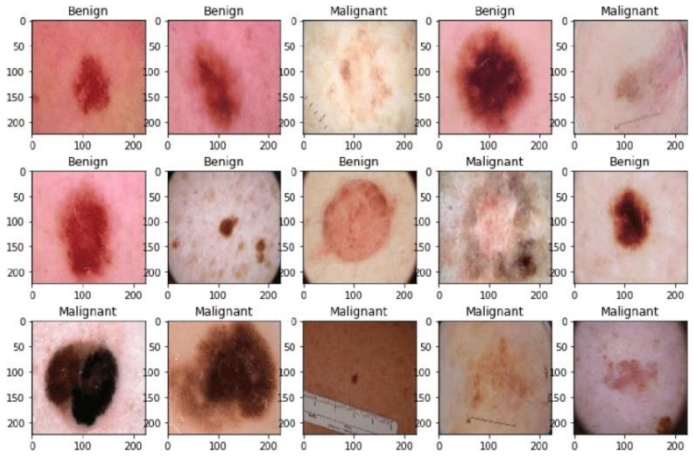


# Federated Learning: Summary.

- Decentralized training on multiple data sources.
- Applications in healthcare, consumer devices, finance.

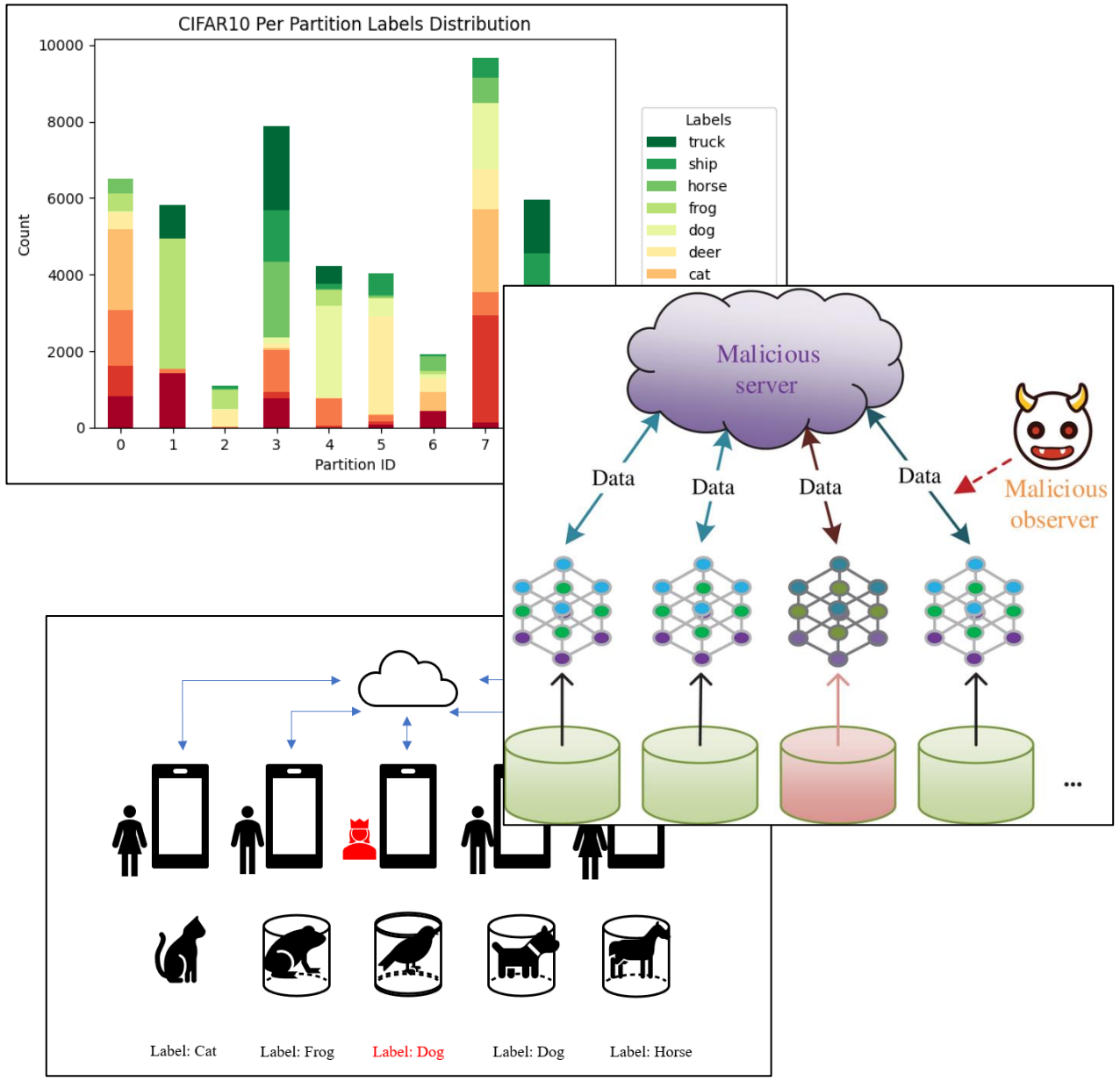







# Federated Learning: Summary.

- Decentralized training on multiple data sources.
- Applications in healthcare, consumer devices, finance.
- Technical challenges include data heterogeneity, communication cost and security of model updates.



# Federated Learning: *Privacy-Preserving Machine Learning at Scale*.

**Part 1**: Why Federated Learning?

Regulatory context, Use cases, Frameworks, and ongoing projects.

**Part 2**: Technical Foundations & Challenges.

Enabling Federated Learning, improving performance, detecting malicious actors.

**Part 3**: Contribution Assessment.

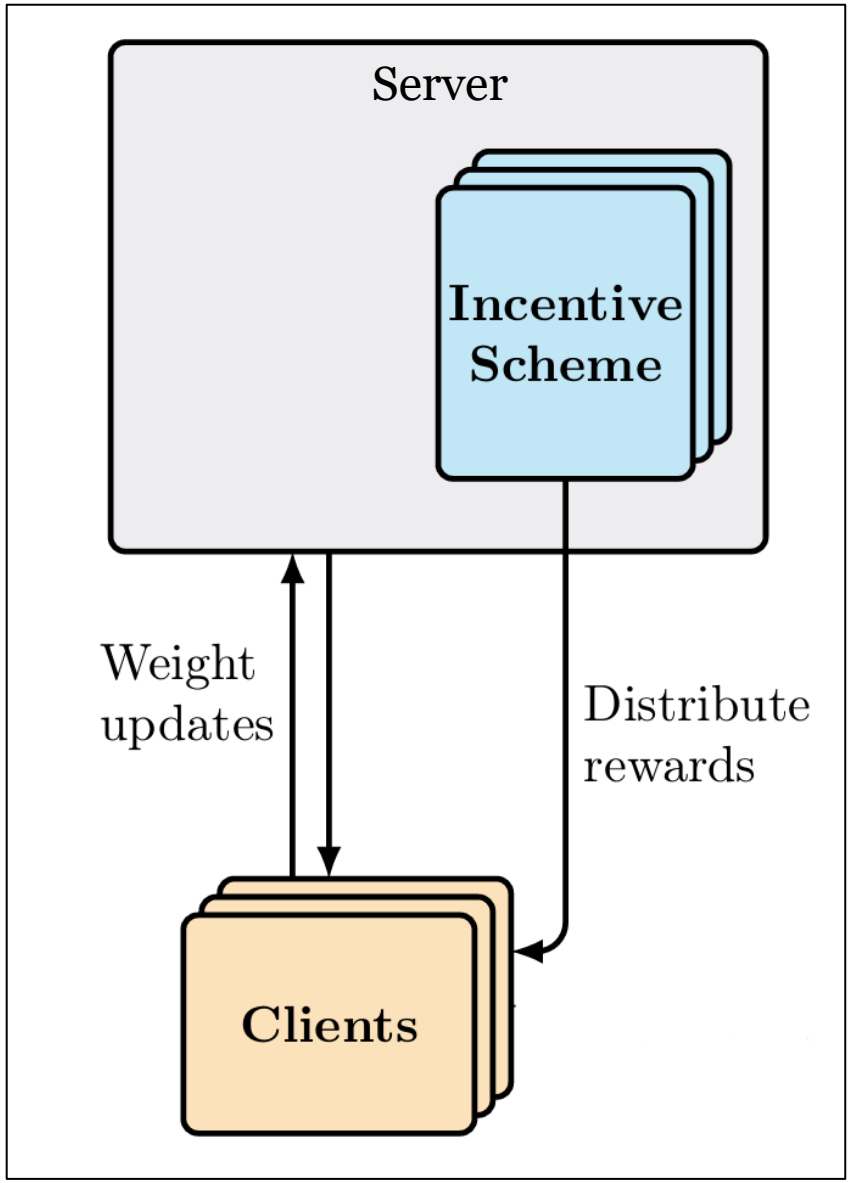
Existing method, our observations, our approaches.

# Contribution Assessment in Federated Learning: Motivation.

- **Fair reward allocation**: (Monetary) incentives for participation.
- **Accountability**: Free rider, malicious client detection.
- **Transparency**: Compliance with regulations, audits.

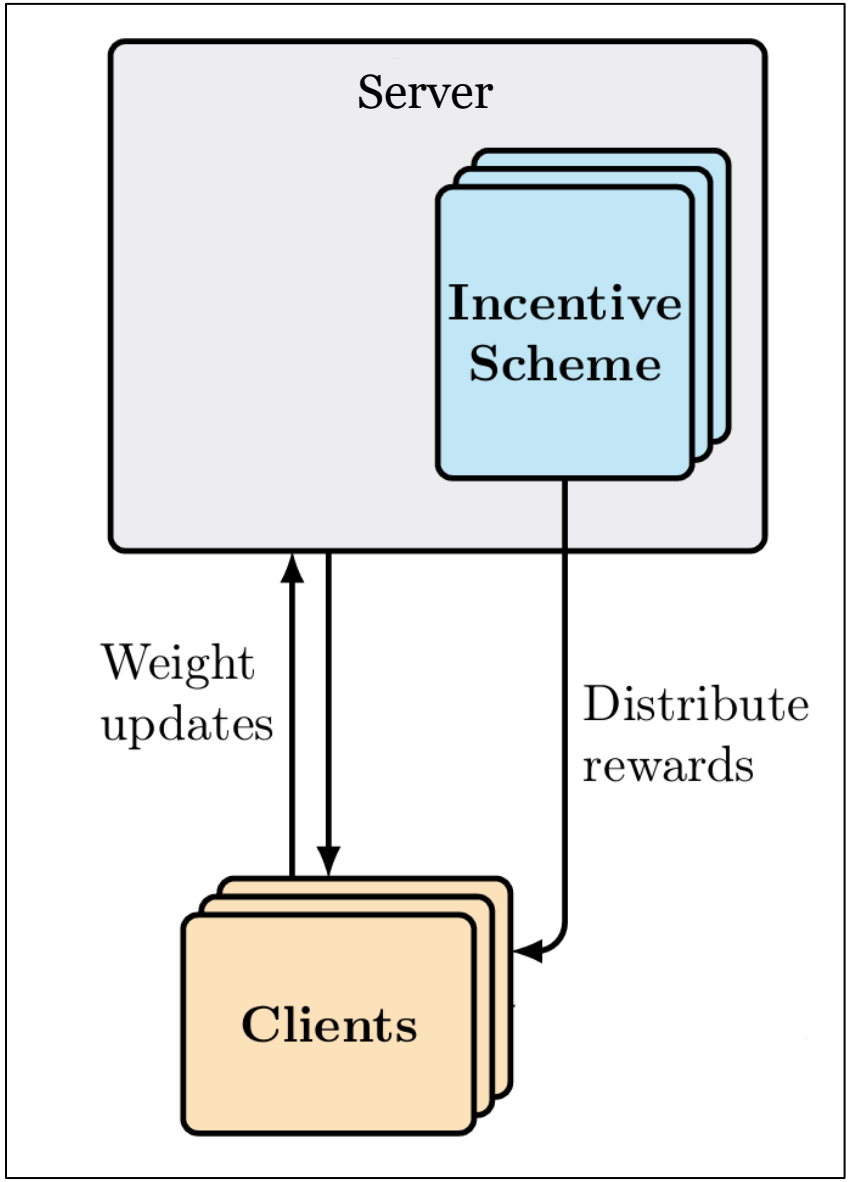
## Example Scenarios:

- **Healthcare**: Spread of COVID-19: High value of datasets which include cases.
- **Fraud detection**: Different sizes of datasets: More data = more impact?
- **Mobile devices**: Some devices provide higher quality user signals.
- **Decentralized LLM finetuning**: Malicious actors can be detected due to bad contribution.



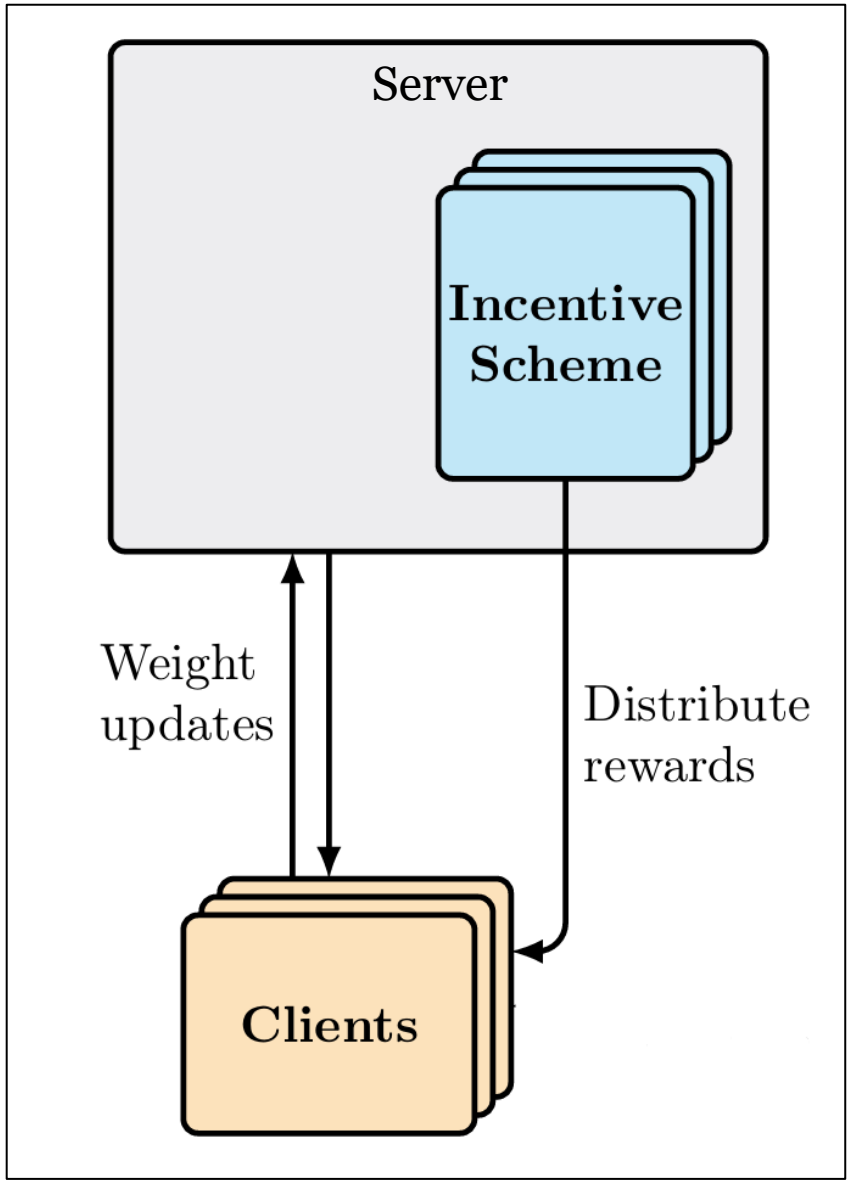
## Contribution calculation methods:

- **Fixed**: Determined pre-federation. Examples: Dataset size, company revenue.
- **Similarity measures**: Allows for grouping of clients. More useful for outlier detection than rewarding of clients.
- **Leave-one-out retraining**: Measure influence of single datapoints on final model. Requires re-training the model.
- **Computation-based contribution**: Game-theoretic influence computation methods.



# Computation-based contribution calculation methods: Challenges.

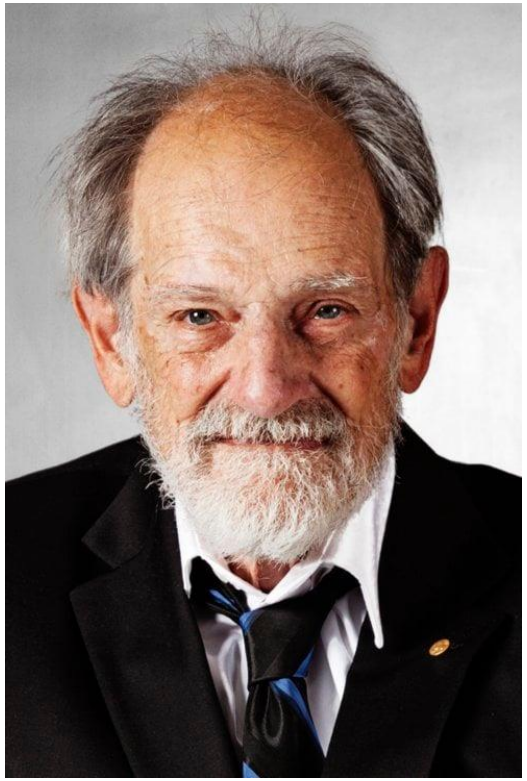
- **Scalability**: Potentially hundreds of different clients.
- Must handle **non-IID data**: Evaluation should not be biased towards certain clients.
- **Robustness** against manipulation: Clients might try to influence final values.
- **Evaluation difficulties**: Often needs access to a test set to evaluate model updates.



## Shapley-value based evaluation:

**Game-theoretic approach**: Fairly distribute the gain amongst a group of players who cooperated.

Used in explainable AI (XAI), resource allocation, financial applications.



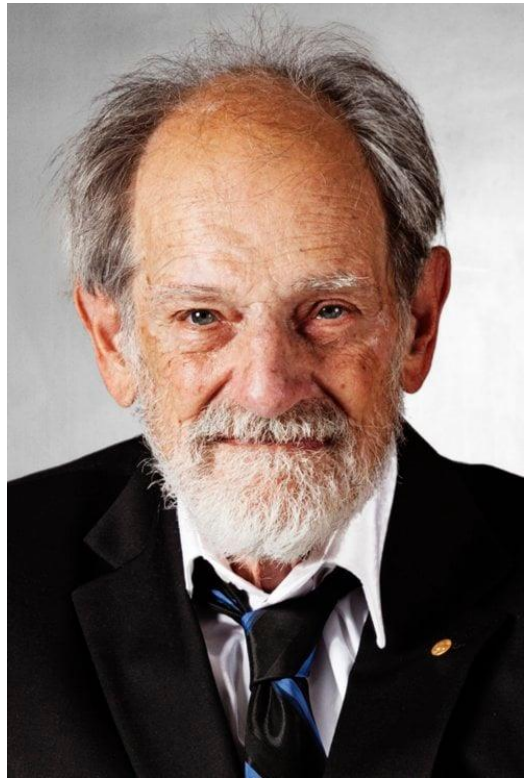
Lloyd Shapley, 1923 - 2016

$$\varphi_k = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{k\}) - v(S)\right)$$

## Shapley-value based evaluation:

**Game-theoretic approach**: Fairly distribute the gain amongst a group of players who cooperated.

Used in explainable AI (XAI), resource allocation, financial applications.



Lloyd Shapley, 1923 - 2016

$$\varphi_k = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{k\}) - v(S)\right)$$

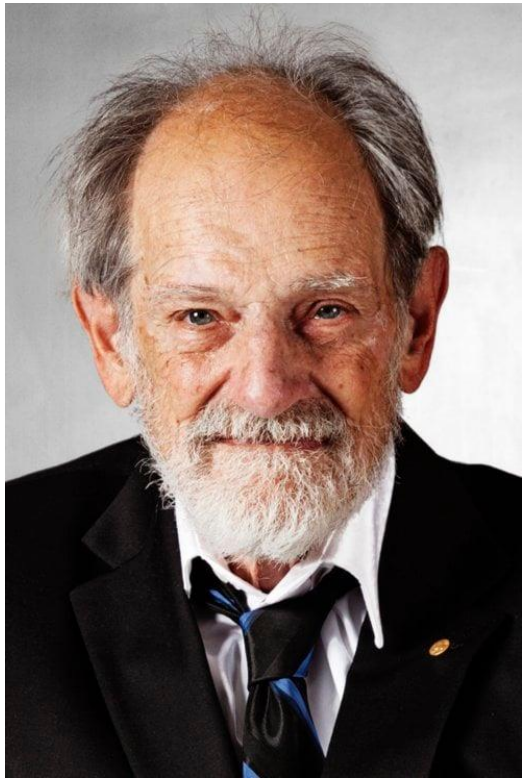
Iterate over all subsets not containing $k$.

⇐ Complexity: $O(2^n)$

## Shapley-value based evaluation:

**Game-theoretic approach**: Fairly distribute the gain amongst a group of players who cooperated.

Used in explainable AI (XAI), resource allocation, financial applications.



Lloyd Shapley, 1923 - 2016

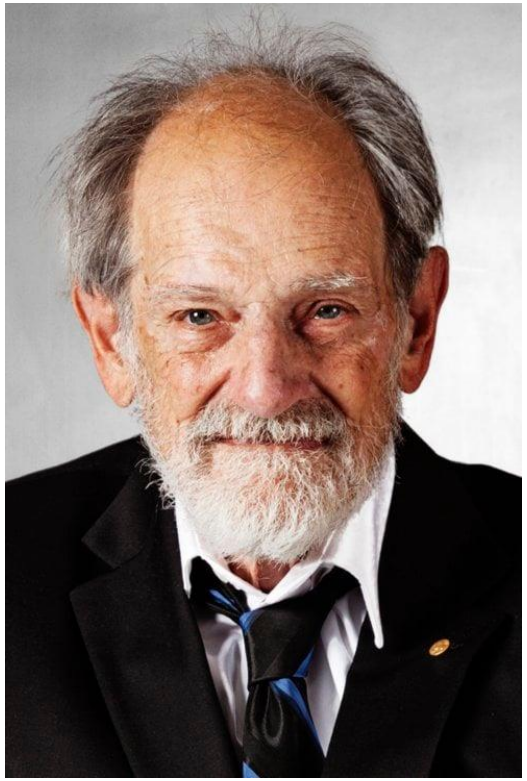
$$\varphi_k = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{k\}) - v(S)\right)$$

Weighting factor.

## Shapley-value based evaluation:

**Game-theoretic approach**: Fairly distribute the gain amongst a group of players who cooperated.

Used in explainable AI (XAI), resource allocation, financial applications.



Lloyd Shapley, 1923 - 2016

$$\varphi_k = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{k\}) - v(S)\right)$$

Value of *k* to the cooperation.

## Shapley-value based evaluation:

Only contribution method which satisfies **efficiency** (sum of all = total gain), **symmetry** (same value = same contribution), **linearity** and **null player** (no value = no gain).

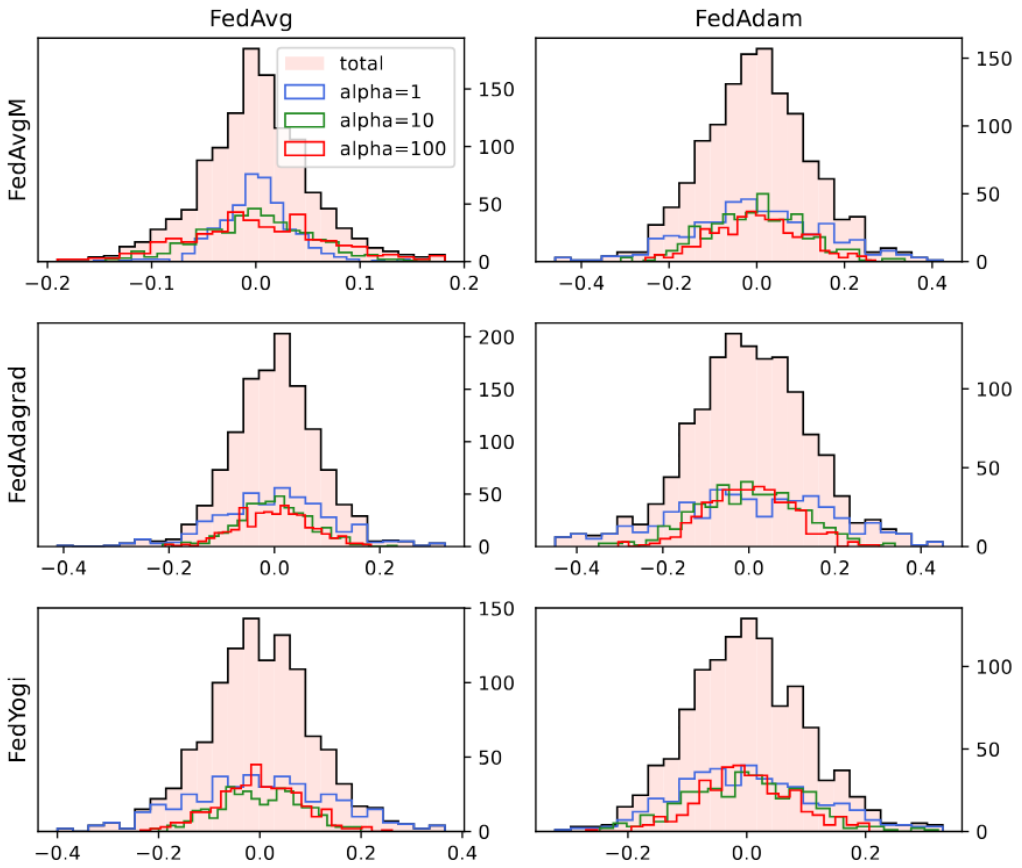
**Federated Learning**: Evaluate value of model updates on a per-round basis.

$$\varphi_k^i = \sum_{S \subseteq N \setminus \{k\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{k\}) - v(S)\right)$$

Value of client update *k* to the global model.

# Our research.

- **Research question**: Are Shapley values aggregation-strategy-invariant?
- **Our approach**: Test Shapley values of the same clients using different aggregation strategies over several datasets. Compare to a set baseline.
- **Our results**: Contributions can differ greatly based on the chosen aggregation strategy.

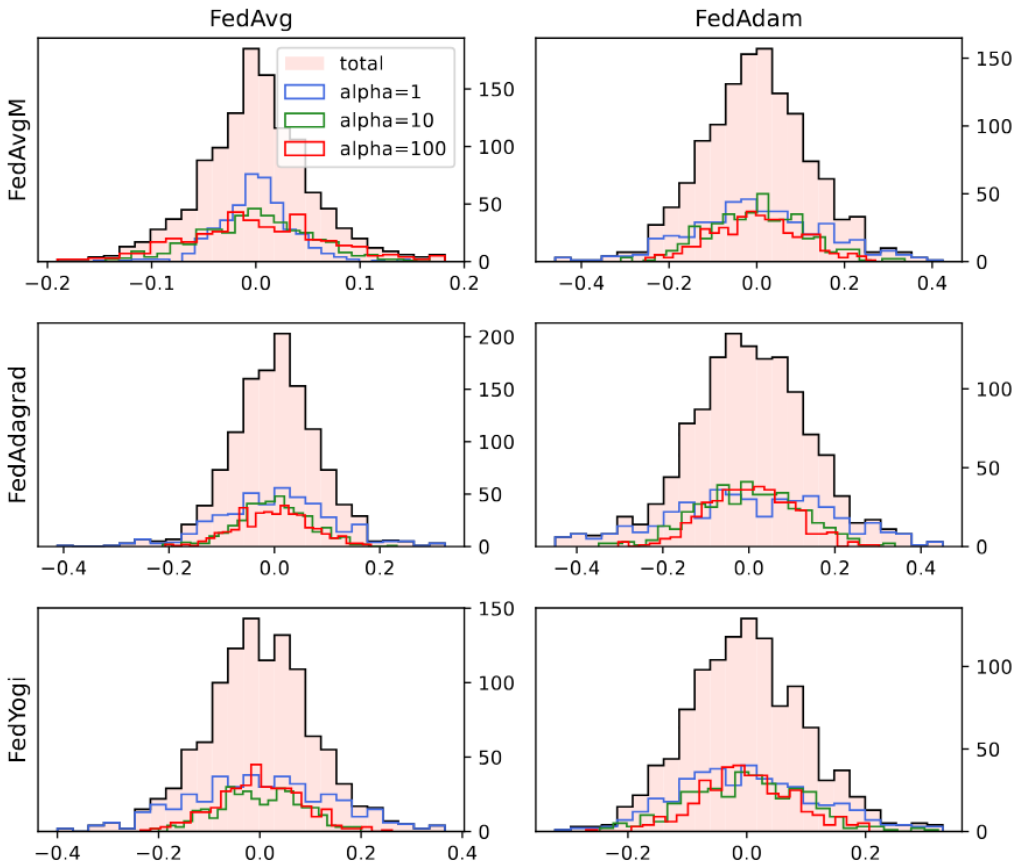



Distribution of the per-client difference in contributions across pairs of aggregation strategies (in % of total reward).

Geimer, Arno, Beltran Fiz, and Radu State. "On the Volatility of Shapley-Based Contribution Metrics in Federated Learning." *IJCNN 2025*.

# Experimental details.

- **Framework**: Flower.
- **Datasets**: MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100 with a simple CNN.
- **Federation hyperparameters**: 5 resp. 3 clients per federation. 3 values for epochs, 3 degrees of heterogeneity in data distribution
- 8 different aggregation strategies.
- Total of  20.000 different FL runs.




Distribution of the per-client difference in contributions across pairs of aggregation strategies (in % of total reward).

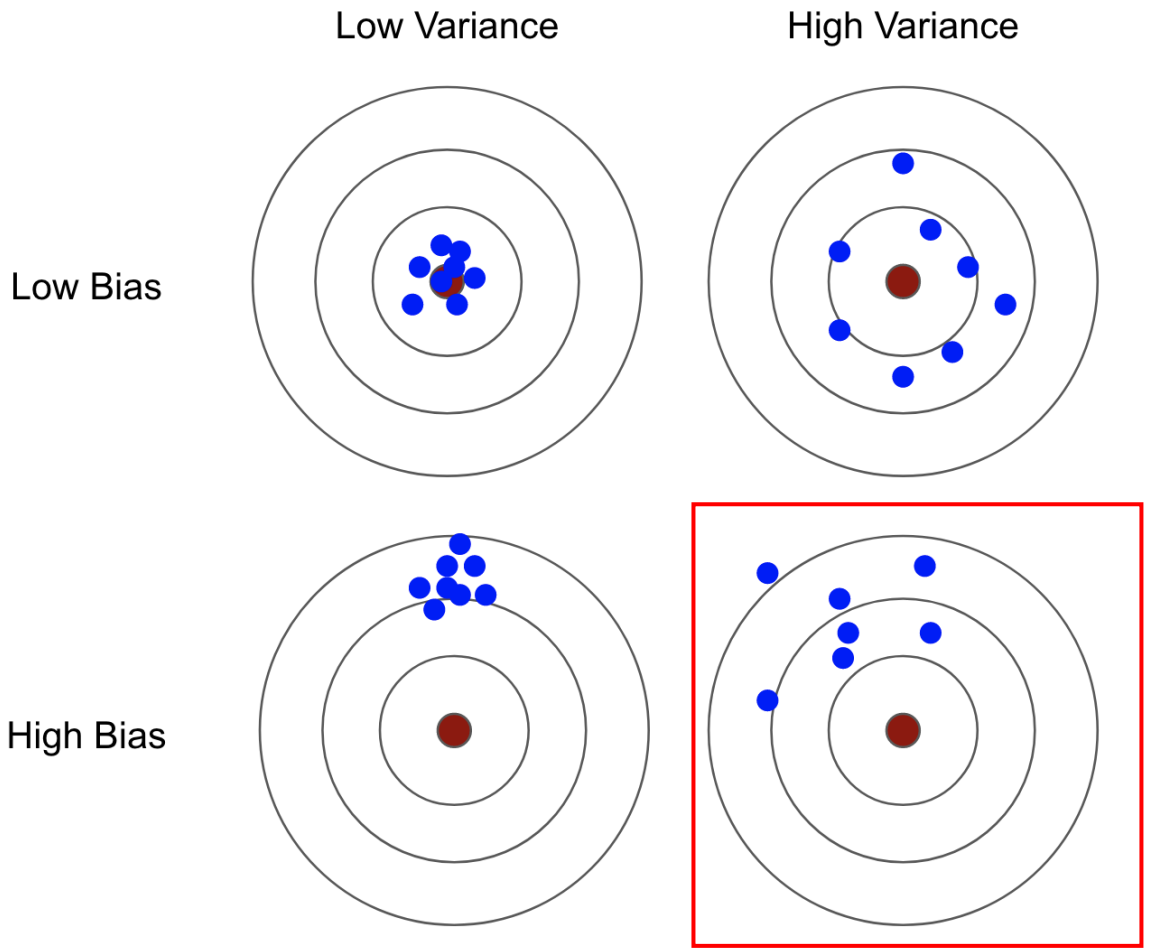
Geimer, Arno, Beltran Fiz, and Radu State. "On the Volatility of Shapley-Based Contribution Metrics in Federated Learning." *IJCNN 2025*.

Euclidean distance to baseline over all experiments.

| $e$ | Strategy \ $\alpha$ | CIFAR-10 1 | CIFAR-10 10 | CIFAR-10 100 | CIFAR-100 1 | CIFAR-100 10 | CIFAR-100 100 | MNIST 1 | MNIST 10 | MNIST 100 | FMNIST 1 | FMNIST 10 | FMNIST 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | FedAvg | 12.63 | 6.13 | **5.99** | **5.31** | 5.89 | 6.88 | 12.62 | 7.65 | 7.5 | 13.06 | 7.13 | 5.71 |
| | FedAvgM | 12.55 | 6.59 | 7.25 | 5.64 | **5.65** | 6.9 | 12.82 | **7.31** | 7.73 | 12.97 | 6.68 | 5.48 |
| | FedAdagrad | 20.72 | 8.61 | 6.44 | 12.32 | 7.29 | **5.56** | 13.84 | 8.69 | 7.66 | 14.37 | 6.9 | 6.29 |
| | FedAdam | 12.7 | 10.1 | 7.14 | 7.26 | 7.21 | 7.09 | **10.2** | 9.93 | 9.31 | 13.32 | 8.26 | 7.23 |
| | FedYogi | **10.39** | 8.21 | 6.49 | 7.22 | 9.17 | 8.18 | 15.22 | 8.15 | **4.67** | 16.53 | 8.09 | 5.57 |
| | FedMedian | 12.78 | 6.18 | 6.73 | 5.76 | 6.45 | 7.35 | 11.43 | 7.86 | 7.07 | **12.71** | **6.62** | **5.26** |
| | FedTrimAvg | 12.6 | **6.03** | 6.5 | 5.91 | 6.21 | 6.05 | 12.39 | 7.33 | 7.46 | 13.04 | 6.82 | 5.43 |
| | Krum | 35.77 | 19.28 | 10.24 | 51.03 | 18.99 | 11.68 | 16.23 | 9.71 | 7.41 | 17.73 | 10.35 | 7.07 |
| 5 | FedAvg | 13.1 | 6.49 | 7.38 | 6.09 | **5.69** | 5.8 | 13.17 | 8.69 | 9.44 | 12.73 | 6.99 | **6.24** |
| | FedAvgM | 12.85 | 6.37 | 7.09 | 5.84 | 5.84 | 6.79 | 12.74 | 8.7 | 8.56 | **12.31** | **6.3** | 7.11 |
| | FedAdagrad | 19.54 | 8.58 | 6.82 | 12.98 | 6.55 | 7.18 | 14.52 | 9.43 | 7.83 | 13.64 | 6.33 | 7.32 |
| | FedAdam | 14.27 | 13.52 | 11.65 | 9.55 | 9.82 | 11.27 | 13.96 | 12.99 | 13.37 | 13.6 | 10.48 | 9.05 |
| | FedYogi | **9.7** | 6.83 | **6.2** | **5.58** | 6.94 | **4.69** | 12.49 | **6.04** | **5.18** | 13.71 | 7.27 | 5.64 |
| | FedMedian | 12.47 | **5.91** | 6.88 | 6.35 | 5.83 | 6.82 | **12.1** | 7.67 | 9.18 | **12.31** | 6.67 | 7.23 |
| | FedTrimAvg | 13.32 | 6.56 | 6.71 | 6.32 | 5.95 | 6.58 | 12.97 | 8.68 | 9.36 | 12.37 | 6.5 | 6.78 |
| | Krum | 33.57 | 18.88 | 10.83 | 38.61 | 15.48 | 9.04 | 16.8 | 10.82 | 10.87 | 17.4 | 10.76 | 8.37 |
| 10 | FedAvg | 13.91 | **6.29** | **7.48** | 7.29 | **5.18** | 7.13 | 13.49 | 10.47 | 10.39 | 12.31 | 7.21 | 8.01 |
| | FedAvgM | 13.32 | 6.44 | 7.82 | 7.61 | 5.63 | **6.28** | 13.0 | 9.31 | 10.62 | 12.48 | 7.05 | 8.93 |
| | FedAdagrad | 19.28 | 8.15 | 8.22 | 13.31 | 8.32 | 6.84 | 12.67 | **8.1** | **7.26** | 12.48 | 8.01 | 8.31 |
| | FedAdam | 14.77 | 12.65 | 8.66 | 11.65 | 7.39 | 8.21 | 14.86 | 11.35 | 11.26 | 13.86 | 12.45 | 11.34 |
| | FedYogi | **10.85** | 8.64 | 8.09 | 7.74 | 6.54 | 8.38 | 15.41 | 12.58 | 9.18 | 12.26 | 9.63 | 8.99 |
| | FedMedian | 12.44 | 7.02 | 9.64 | 6.77 | 6.56 | 8.25 | **12.36** | 9.8 | 10.6 | 12.42 | 8.65 | 8.28 |
| | FedTrimAvg | 13.31 | 6.8 | 7.91 | **6.45** | 5.52 | 6.54 | 12.81 | 9.54 | 10.39 | **12.11** | **6.77** | **7.5** |
| | Krum | 29.6 | 18.43 | 13.73 | 30.62 | 15.27 | 10.45 | 16.98 | 13.64 | 12.93 | 17.2 | 12.57 | 10.61 |

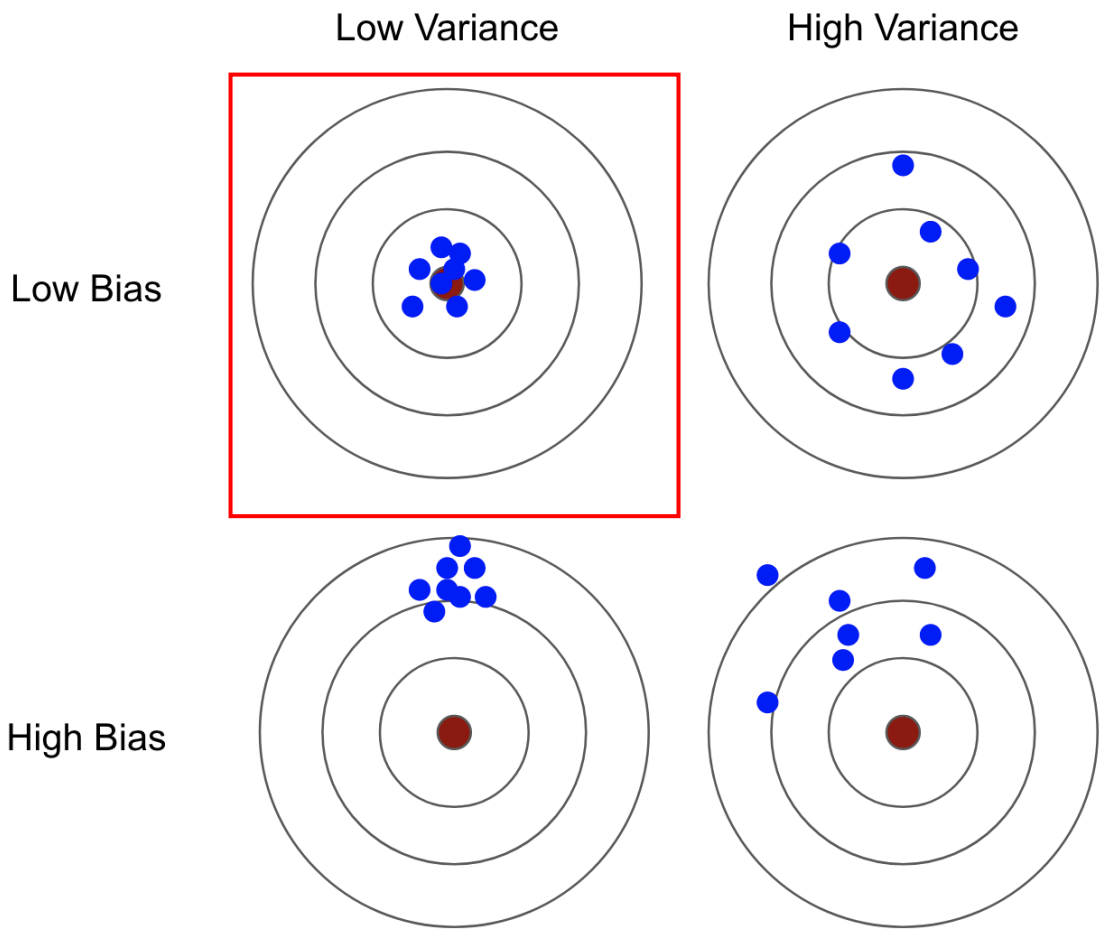
# Problem Statement.

- **Observation**: Big variance between aggregation strategies, no good approximation of the baseline.
- Can we decrease the bias? If not, can we at least decrease variance between different runs?



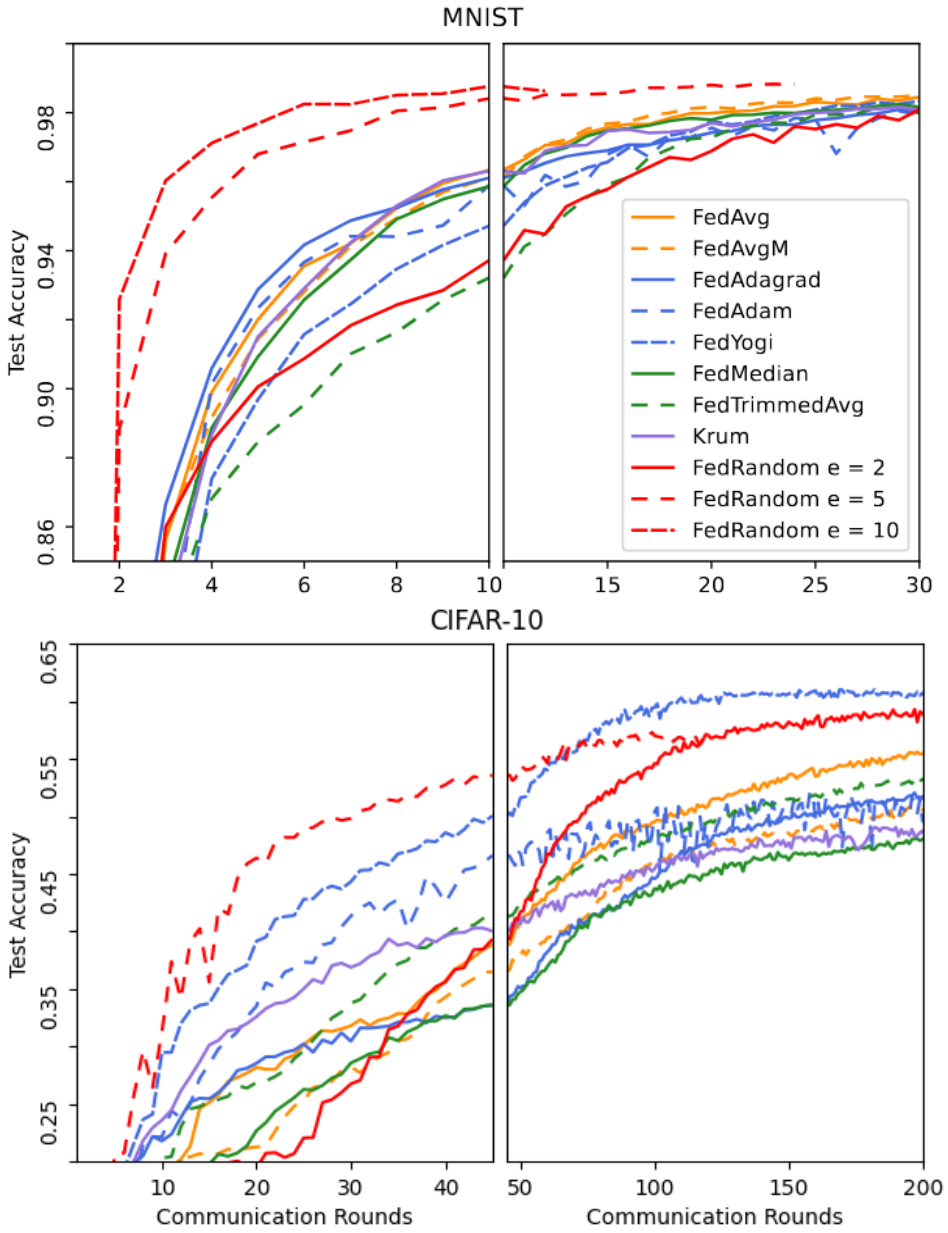
# Problem Statement.

- **Observation**: Big variance between aggregation strategies, no good approximation of the baseline.
- Can we decrease the bias? If not, can we at least decrease variance between different runs?



# Proposed method.

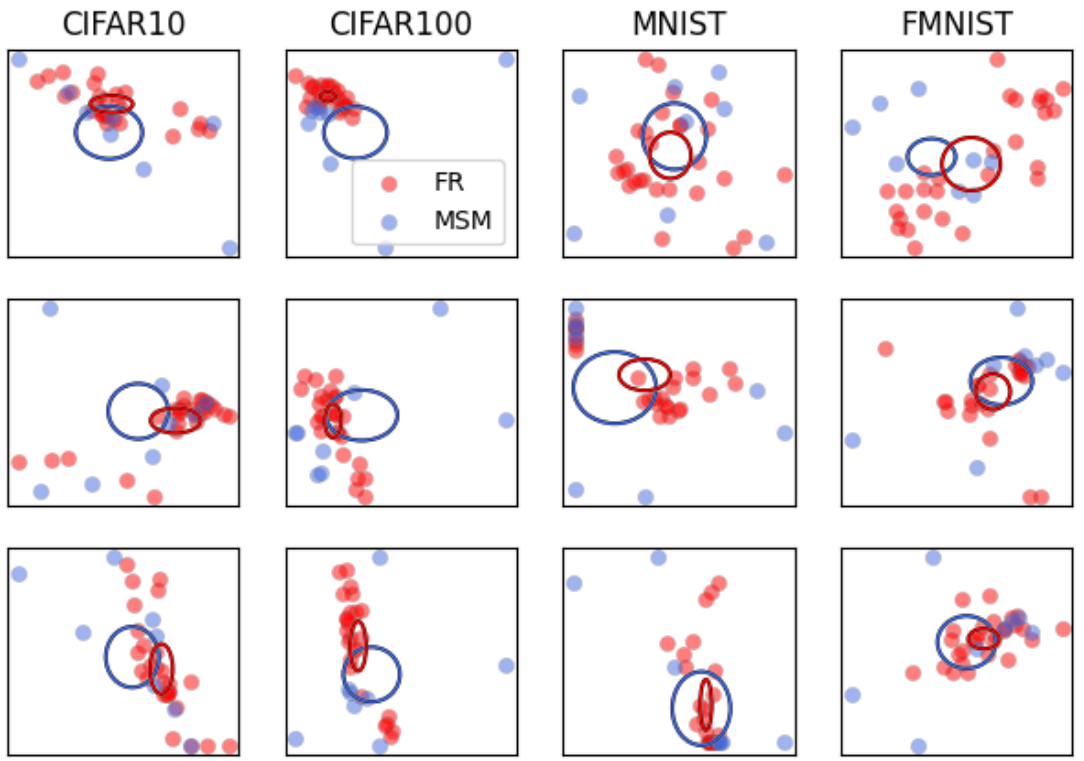
- If we consider different aggregation strategies as noisy samples, can we gain additional information from statistical properties?
- Can we find a method to produce more samples, increasing soundness of our results?
- **FedRandom**: Samples a random aggregation strategy each round.




Accuracy of different aggregation strategies on two datasets.

# Results

- **FedRandom** drastically reduces (94% of cases) sample variance. Thus, the contribution values we get from different runs lie closer together than in the baseline aggregation strategies.

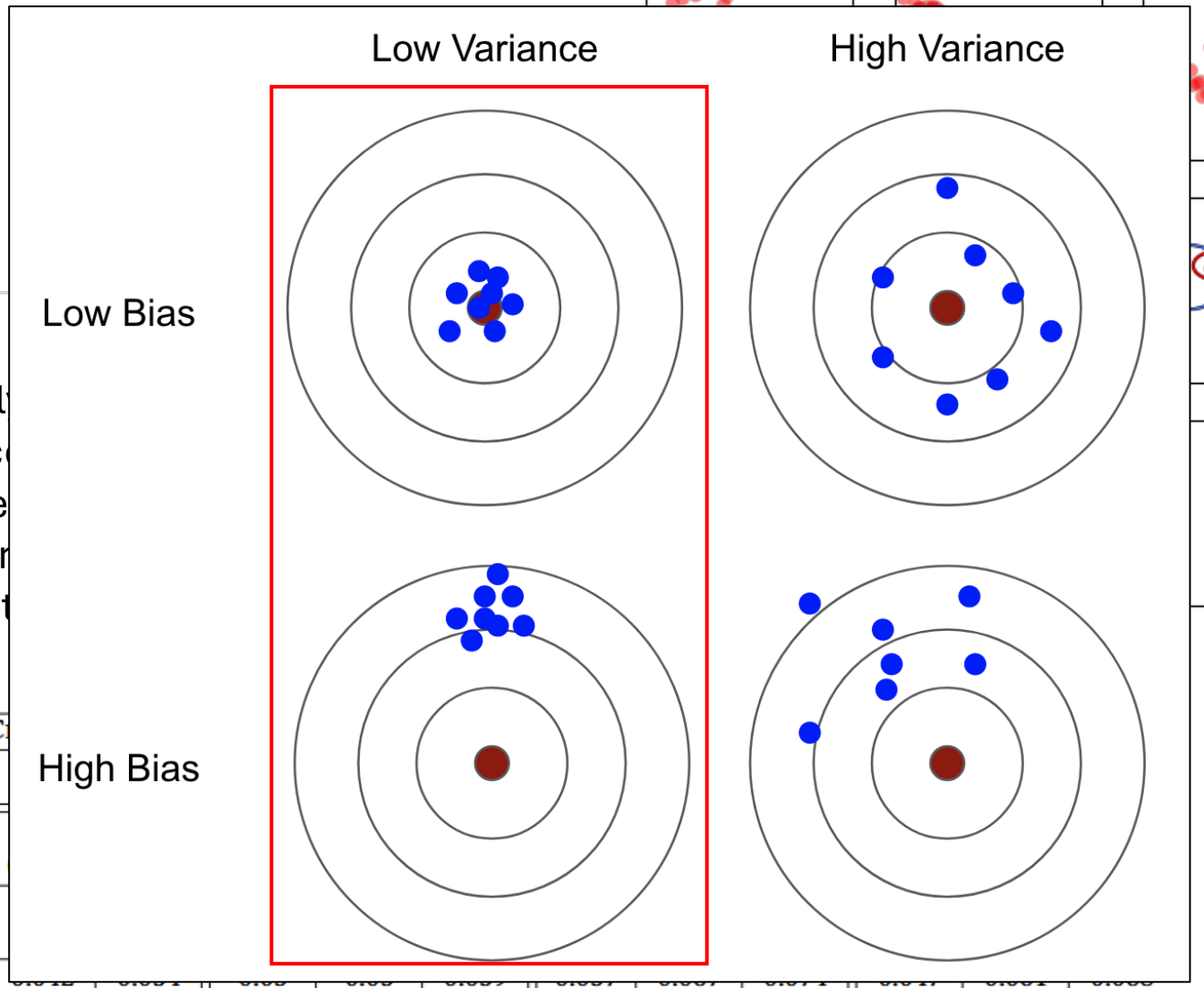



| | | CIFAR-10 | | | CIFAR-100 | | | MNIST | | | FMNIST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $e$ \ $\alpha$ | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 100 |
| MSM | 2 | 0.059 | 0.056 | 0.057 | 0.069 | 0.064 | 0.066 | 0.052 | 0.06 | 0.069 | 0.046 | 0.058 | 0.069 |
| FR | 2 | **0.006** | **0.003** | **0.005** | 0.003 | **0.003** | **0.005** | **0.01** | **0.014** | **0.014** | **0.007** | **0.013** | **0.016** |
| MSM | 5 | 0.057 | 0.051 | 0.067 | 0.054 | 0.055 | 0.059 | 0.056 | 0.061 | 0.072 | 0.045 | 0.062 | 0.072 |
| FR | 5 | **0.006** | 0.004 | 0.007 | **0.002** | **0.003** | **0.005** | 0.012 | **0.014** | 0.017 | 0.01 | 0.014 | 0.017 |
| MSM | 10 | 0.05 | 0.042 | 0.054 | 0.05 | 0.05 | 0.059 | 0.057 | 0.067 | 0.074 | 0.047 | 0.061 | 0.068 |
| FR | 10 | 0.007 | 0.006 | 0.011 | 0.003 | **0.003** | **0.005** | 0.012 | 0.015 | 0.016 | 0.011 | 0.015 | 0.017 |

**Table 1: Average standard deviation in contribution samples generated by MSM and by FEDRANDOM (FR), lower is better. Bold entries are best in column.**

# Results

- **FedRandom** drasticall
  cases) sample varianc
  contribution values we
  runs lie closer together
  baseline aggregation st





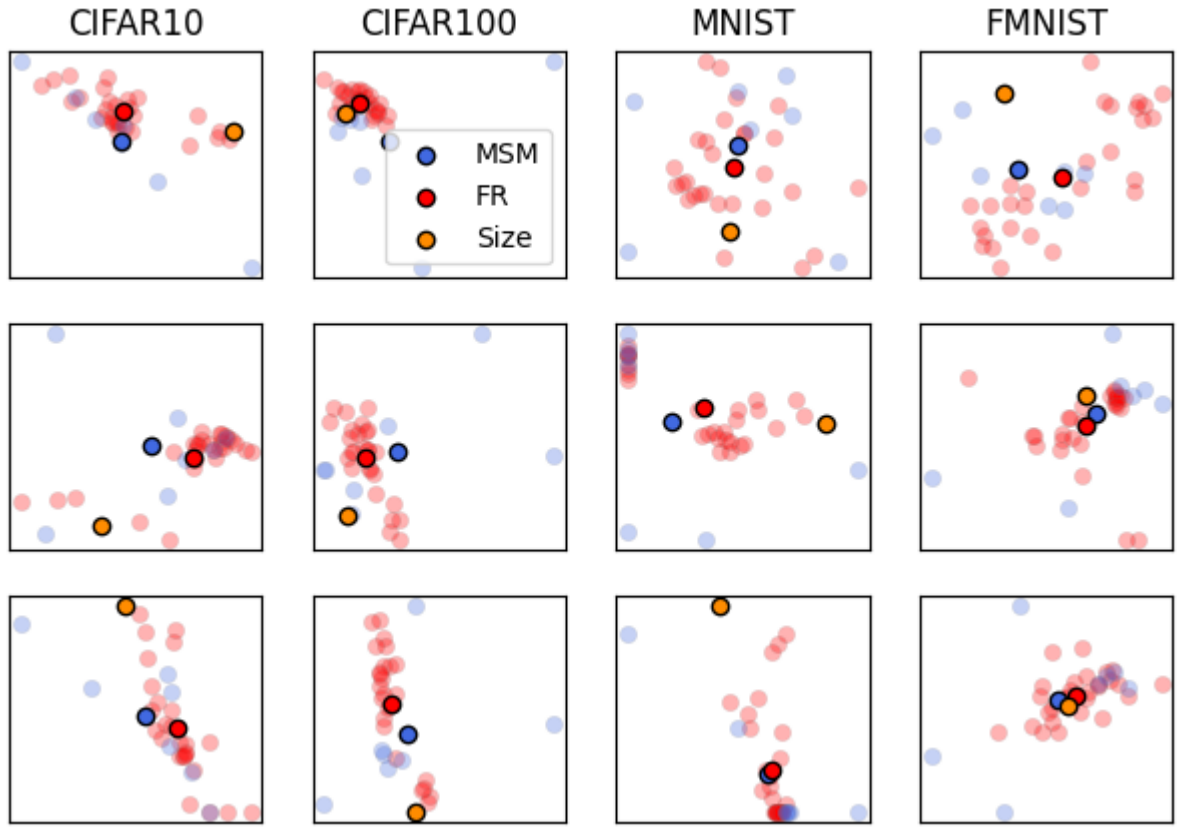


| | e \ α | 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSM | 2 | 0.059 | | | | | | | | | | | |
| FR | | **0.006** | | | | | | | | | | | |
| MSM | 5 | 0.057 | | | | | | | | | | | |
| FR | | **0.006** | | | | | | | | | | | |
| MSM | 10 | 0.05 | | | | | | | | | | | |
| FR | | 0.007 | 0.006 | 0.011 | 0.003 | **0.003** | **0.005** | 0.012 | 0.015 | 0.016 | 0.011 | 0.015 | 0.017 |

**Table 1: Average standard deviation in contribution samples generated by MSM and by FedRandom (FR), lower is better. Bold entries are best in column.**

# Results

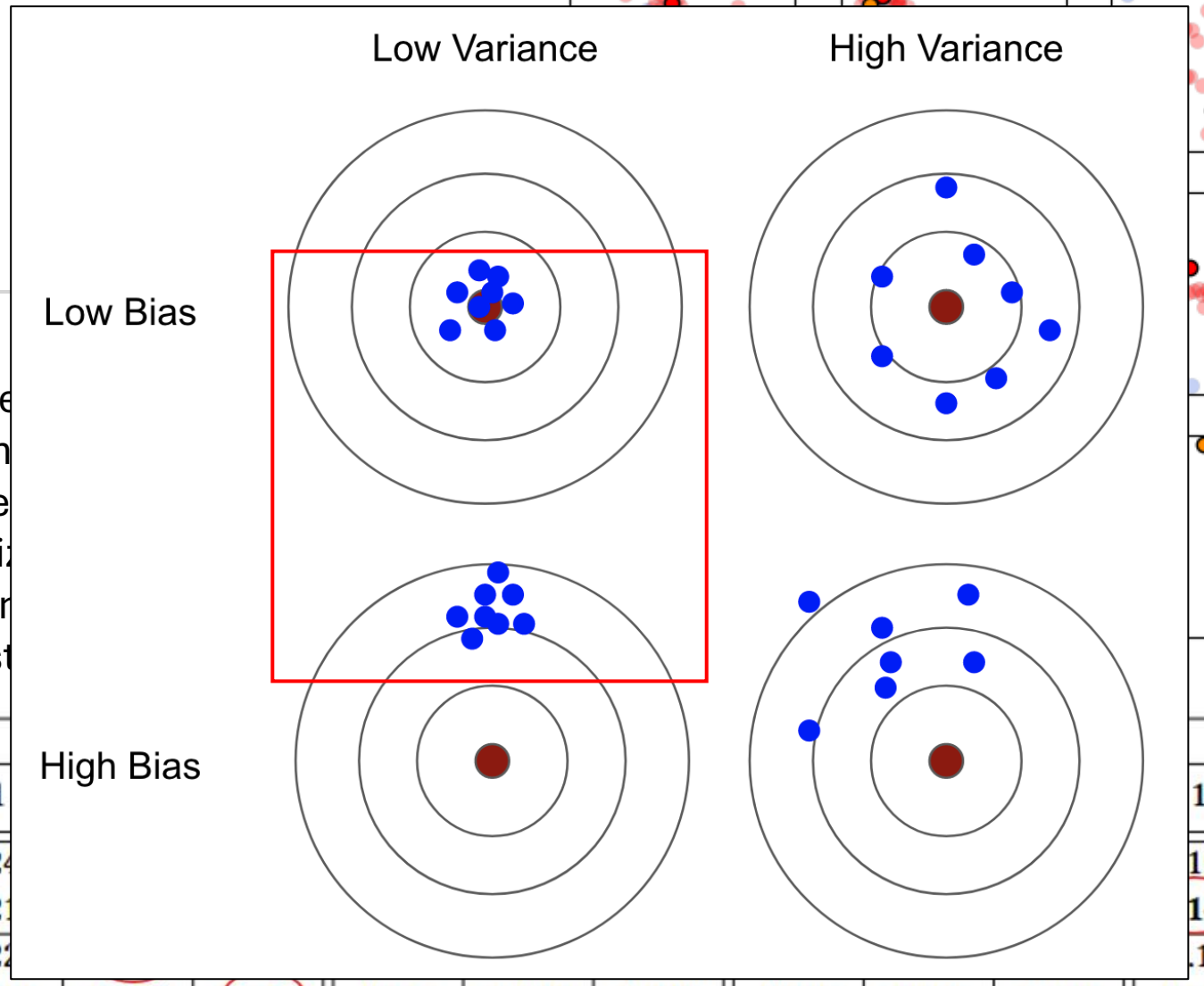
- **FedRandom** slightly reduces (83% of cases) sample bias. Thus, the contribution values we get from different runs lie closer to the size-based underlying contributions than the baseline aggregation strategies.




| | | CIFAR-10 | | | CIFAR-100 | | | MNIST | | | FMNIST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $e$ \ $\alpha$ | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 100 |
| MSM | 2 | 0.245 | 0.299 | 0.349 | 0.206 | 0.359 | 0.268 | 0.147 | 0.165 | 0.176 | 0.156 | 0.178 | 0.256 |
| FR | | 0.214 | 0.26 | **0.342** | 0.148 | 0.346 | 0.255 | 0.099 | **0.12** | 0.111 | **0.143** | **0.128** | 0.221 |
| MSM | 5 | 0.223 | **0.245** | 0.37 | 0.198 | 0.344 | 0.251 | 0.133 | 0.201 | 0.137 | 0.16 | 0.19 | 0.237 |
| FR | | 0.214 | 0.253 | **0.342** | 0.146 | 0.343 | 0.242 | 0.087 | 0.126 | 0.118 | 0.153 | 0.142 | **0.213** |
| MSM | 10 | **0.212** | 0.269 | 0.344 | 0.181 | **0.321** | 0.252 | 0.112 | 0.175 | 0.15 | 0.177 | 0.161 | 0.22 |
| FR | | **0.212** | 0.247 | 0.346 | **0.142** | 0.349 | **0.241** | **0.075** | 0.13 | **0.108** | **0.143** | 0.15 | 0.218 |

# Results

- **FedRandom** slightly re
cases) sample bias. Th
contribution values we
runs lie closer to the si
underlying contributio
baseline aggregation st



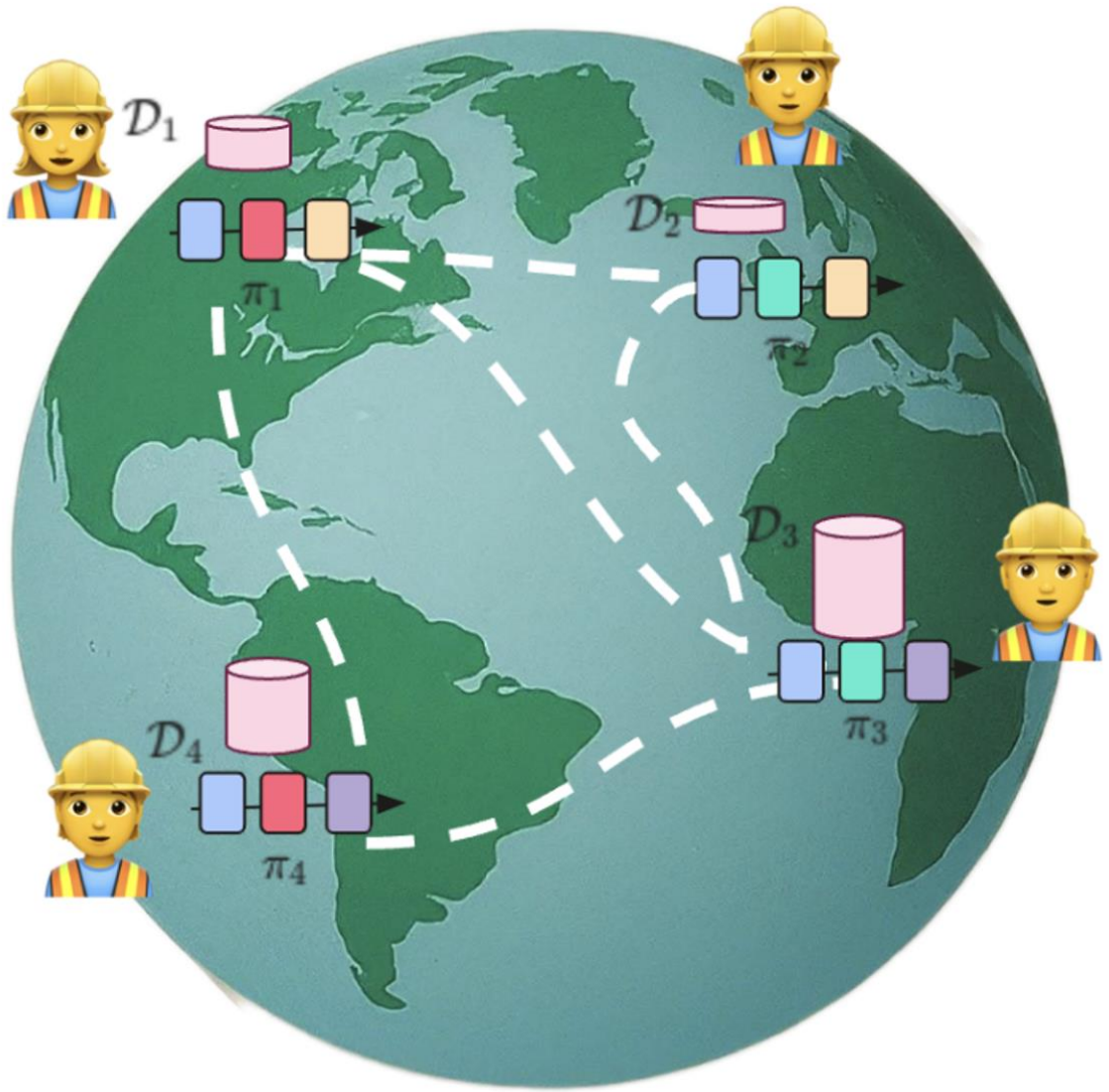




| | $e$ \ $\alpha$ | 1 | | | | | | | | | FMnist 1 | FMnist 10 | FMnist 100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSM | 2 | 0.2 | | | | | | | | | 156 | 0.178 | 0.256 |
| FR | | 0.2 | | | | | | | | | 143 | **0.128** | 0.221 |
| MSM | 5 | 0.2 | | | | | | | | | .16 | 0.19 | 0.237 |
| FR | | 0.214 | 0.253 | **0.342** | 0.146 | 0.343 | 0.242 | 0.087 | 0.126 | 0.118 | 0.153 | 0.142 | **0.213** |
| MSM | 10 | **0.212** | 0.269 | 0.344 | 0.181 | **0.321** | 0.252 | 0.112 | 0.175 | 0.15 | 0.177 | 0.161 | 0.22 |
| FR | | **0.212** | 0.247 | 0.346 | **0.142** | 0.349 | **0.241** | **0.075** | 0.13 | **0.108** | **0.143** | 0.15 | 0.218 |

# Future work.

- A **more extensive study** of contribution value instability. More use cases, bigger datasets, more up-to-date models.
- Try to **gain an understanding** of why contributions vary.
- Introduction of a better **stability mechanism** which accurately approximates the underlying data values.



# Any questions?

Federated Learning use cases?

Technical details?

Doing a PhD?

Future of Federated Learning?

Please repeat?