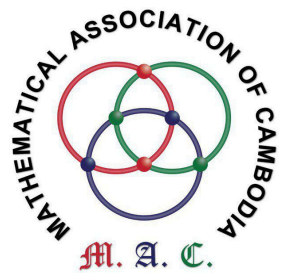
Introduction of Federated Learning

KIMSIE PHAN

M.Sc Mathematics and Computing at IIT DHANBAD, INDIA

phankimsie03@gmail.com



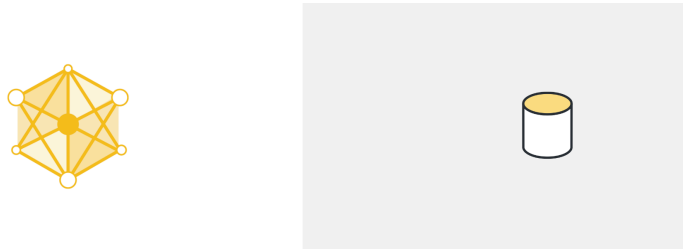
Mathematical Association of Cambodia

Computing for Secure and Intelligent Networks Lab

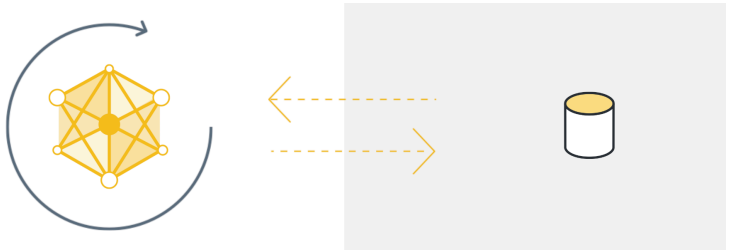
May 29, 2025

Classical (Centralized) Machine Learning

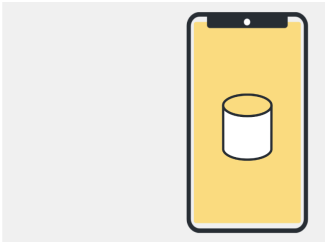
In machine learning, we have a model, and we have data. The model could be a neural network, or something else, like linear regression.



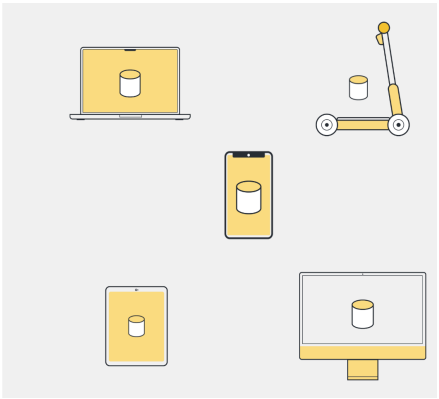
We train the model using the data to perform a useful task. A task could be to detect objects in images, transcribe an audio recording, or play a game.



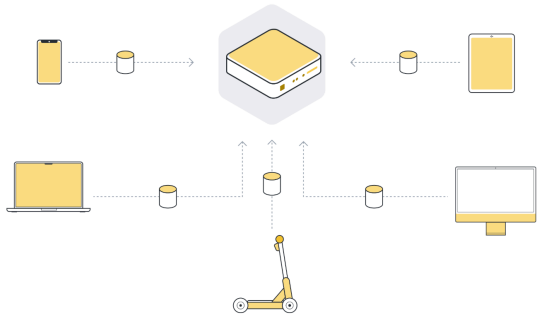
In practice, the training data we work with doesnt originate on the machine we train the model on. This data gets created somewhere else. For instance, the data can originate on a smartphone by the user interacting with an app, a car collecting sensor data, a laptop receiving input via the keyboard, or a smart speaker listening to someone trying to sing a song.



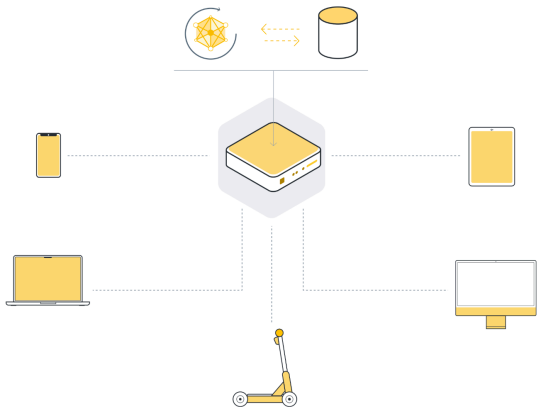
Whats also important to mention, this somewhere else is usually not just one place, its many places. It could be several devices all running the same app. But it could also be several organizations, all generating data for the same task.



So to use machine learning, the approach that has been used in the past was to collect all this data on a central server. This server can be located somewhere in a data center, or somewhere in the cloud.

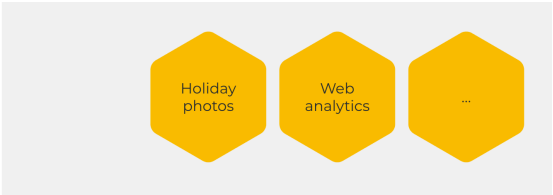


Once all the data is collected in one place, we can finally use machine learning algorithms to train our model on the data. This is the machine learning approach that we've basically always relied on.

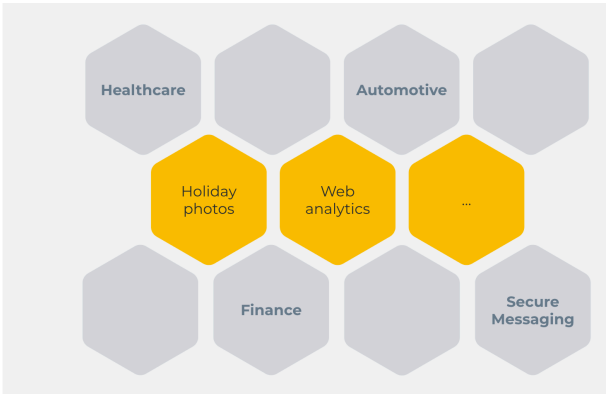


Challenges of classical machine learning

This classical machine learning approach weve just seen can be used in some cases. Great examples include categorizing holiday photos, or analyzing web traffic. Cases, where all the data is naturally available on a centralized server.



But the approach can not be used in many other cases. Cases, where the data is not available on a centralized server, or cases where the data available on one server is not enough to train a good model.



Motivation to Federated Learning

Federated Learning simply reverses this approach. It enables machine learning on distributed data by moving the training to the data, instead of moving the data to the training. Heres a one-liner explanation:

Centralized Machine Learning: move the data to the computation.
Centralized Federated Learning: move the computation to the data.

By doing so, Federated Learning enables us to use machine learning in areas where it wasn't possible before. We can now train excellent medical AI models by enabling different hospitals to work together. We can solve financial fraud by training AI models on the data of different financial institutions. We can build novel privacy-enhancing applications (such as secure messaging) that have better built-in AI than their non-privacy-enhancing alternatives. And those are just a few of the examples that come to mind.

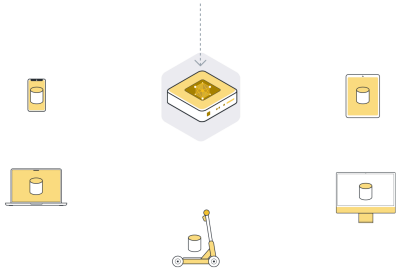
Federated Learning

Definition

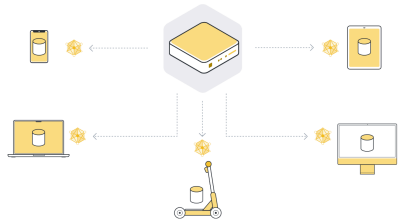
Federated Learning (FL) is a machine learning technique that enables multiple **clients** (such as devices or organizations) to collaboratively train a machine learning model while keeping their data local and private. Instead of sending raw data to a central **server**, clients train the model on their own data and only share the **parameters** to a central server. The server **aggregates** these updates to improve the global model, which is then sent back to the clients for further training. This process continues iteratively until the model converges.

Workflow of Federated Learning

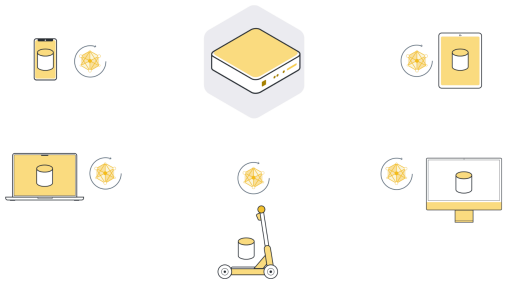
Step 0: A global model is initialized on the central server.



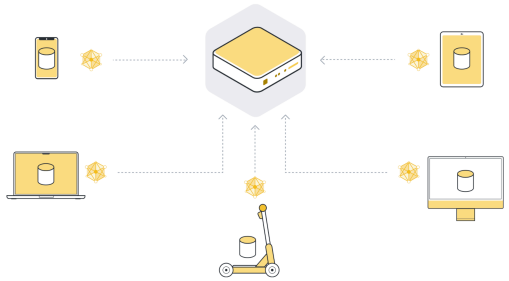
Step 1: Server send to a number of connected client nodes



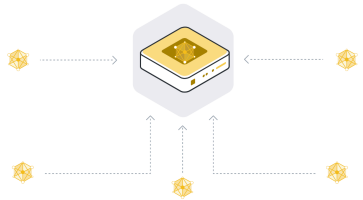
Step 2: Train model locally on the data of each client node



Step 3: Return updated parameters back to the server.



Step 4: Aggregate updated parameters into a new global model.



Step 5: Repeat steps 1 to 4 until the model converges.

Definition (Parameters)

In Machine Learning, **parameters** are the internal values that a model learns from the data during training. They control how the model makes predictions, and they are updated again and again to make the model better.

Examples of Models with Parameters

1. Linear Regression
   - Formula: $y = w \cdot x + b$
   - Parameters: w is weight (slope) and b is bias (intercept)
   - Case: Predicting a continuous value.
2. Logistic Regression
   - Formula: $P(y = 1|x) = \frac{1}{1+e^{-(w \cdot x + b)}}$
   - Parameters: w is weight vector and b is bias
   - Case: Predicting the probability.
3. Convolutional Neural Network (CNN) :
   - Parameters: Filter weights (kernel values) are values inside the filter that are learned, and biases are one for each filter.
   - Case: Image classification

Optimization Methods in Machine Learning

Optimizers in machine learning are algorithms that adjust a models parameters (usually weights and biases) to minimize the loss function during training. The loss function measures how far the models predictions are from the actual (true) values. The optimizer works to reduce this difference by updating the parameters step-by-step.

Definition (Gradient Descent (GD))

Gradient Descent (GD) is an optimization algorithm used to minimize a loss function by adjusting the models parameters (like weights and biases) step by step in the direction that reduces the error the most. The updateed parameters is defined by

$$\theta := \theta - \eta \cdot \nabla_\theta L(\theta)$$

where

- θ : Model parameters (weight, biases)
- $\nabla_\theta L(\theta)$: Gradient of the loss function L w.r.t parameters θ
- η : Learning rate (step size).

Definition (Stochastic Gradient Descent (SGD))

Stochastic Gradient Descent (SGD) is a type of gradient descent that updates the models parameters using only one data sample (or a small batch) at a time rather than the entire dataset. Same as GD, the updated parameters is defined by

$$\theta := \theta - \eta \cdot \nabla_\theta L(\theta)$$

But here, $L(\theta)$ is computed on just one training sample or batch, not the whole dataset.

Definition (SGD with Momentum)

SGD with Momentum is an improvement over regular Stochastic Gradient Descent (SGD). It helps accelerate learning and smooth out noisy updates by taking into account the past gradients when updating the parameters.
Let v be the velocity or momentum term and μ be the momentum coefficent. Then the updates are:

- Update the velocity:
$$v := \mu v - \eta \cdot \nabla L(\theta)$$
- Update the parameter:
$$\theta := \theta + v$$

Definition (RMSProp)

RMSProp (Root Mean Square Propagation) is an adaptive learning rate optimization algorithm designed to overcome the limitations of plain Gradient Descent, especially in cases where the gradients vary greatly in different directions (e.g., deep neural networks).

- Update the squared gradient average:

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma) g_t^2$$

- Update the parameter:

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} \cdot g_t$$

where g_t is the gradient at time step, $E\left[g^2\right]$ is moving average of squared gradients, γ is decay rate (usually around 0.9), η is learning rate, ϵ is small value to prevent division by zero (e.g., 10^{-8}), and θ is the parameter.

Definition (Adam (Adaptive Moment Estimation))

Adam (Adaptive Moment Estimation) is one of the most popular and powerful optimization algorithms used in training deep learning models. It combines the benefits of Momentum and RMSProp to provide adaptive learning rates for each parameter and faster convergence.

- Update biased first moment estimate (momentum):
  $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
- Update biased second moment estimate (RMS):
  $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
- Bias correction: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
- Update parameter: $\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$

where g_t : gradient at step t, m_t : first moment (mean of gradients), v_t : second moment (uncentered variance of gradients), θ_t : model parameter, β_1, β_2 : decay rates for the moving averages (commonly 0.9 and 0.999), η : learning rate, ϵ : small number to prevent division by zero (e.g., 10^{-8}).

Comparision of Optimizers / When to use which ?

Optimizer	Uses Momentum	Fast Convergence	Good Noisy gradients	Generalization
GD	No	No	No	Yes
SGD	No	No	No	Yes
SGD+Momentum	Yes	Yes	Yes	Yes
RMSProp	No	Yes	Yes	Sometimes
Adam	Yes	YesYes	YesYes	May overfit

Task Type	**Recommended Optimizer**
Small datasets	GD or SGD
Deep learning (CNNs, NLP, etc.)	Adam (default choice)
Noisy data or unstable training	RMSProp or Adam
Want best generalization	SGD + Momentum
Recurrent Neural Networks (RNNs)	RMSProp

Strategy of Federated Learning

In Federated Learning (FL), a strategy is the method used to coordinate client participation and combine their model updates to improve the global model. Now we observe three popular FL strategies such as FedAvg, FedProx, and FedNova.

Definition (FedAvg (Federated Averaging))

A simple strategy where clients train locally and the server averages their model updates to create a new global model is called FedAvg. The server aggregates the new global model as

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^{(k)}$$

- K : number of clients, n_k : number of samples on client k
- w_t : global model at round t, $w_{t+1}^{(k)}$: updated model on client k
- $n = \sum_{k=1}^{K} n_k$: total number of smaples across all participating clients.

Definition (FedProx (Federated Proximal))

An improved FedAvg that adds a term to limit how much local models can deviate from the global model is called FedProx. It modifies the local training objective by adding a proximal term that penalizes the local model from drifting too far from the global model.
The server aggregates the new global model as

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^{(k)}$$

- $F_k(w)$: local loss function on client k, K : number of clients,
- w_t : global model at round t, $w_{t+1}^{(k)}$: updated model on client k
- n_k : number of samples on client k
- $n = \sum_{k=1}^{K} n_k$: total number of smaples across all participating clients.

Definition (FedNova (Federated Normalized Averaging))

Normalizes model updates based on local training steps to ensure fair contribution from all clients, even if they train for different durations is called FedNova. Server aggregates normalized updates:

$$w_{t+1} = w_t - \sum_{k=1}^{K} \frac{n_k}{n} \cdot \frac{\Delta_k}{\tau_k}$$

- K : number of participating clients,
- w_t : global model at round t
- w_{t+1} : updated global model after round $t+1$
- n_k : number of samples on client k
- $n = \sum_{k=1}^{K} n_k$: total number of smaples across all participating clients.
- $\Delta_k = w_t - w_k$: The difference between the global model and the updated local model from client k.
- τ_k : The total number of local gradient steps performed by client k in that round.

General Loss function per client and all clients

1. General Loss Function Formula (per client):

$$F_k(w) = \frac{1}{n_k} \sum_{i=1}^{n_k} l\left(f(w; x_i), y_i\right)$$

- $F_k(w)$: Local loss function on client k
- n_k: Number of data points on client k
- (x_i, y_i): Data sample from client k's dataset
- $f(w, x_i)$: Model prediction using parameters w
- l: Loss function (e.g., MSE)

In FedProx, General Loss Function :

$$F_k^{prox}(w) = F_k(w) + \frac{\mu}{2}||w - w_t||^2, \quad \mu : \text{ proximal term coefficient}$$

2. Global Loss Function (all clients together):

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

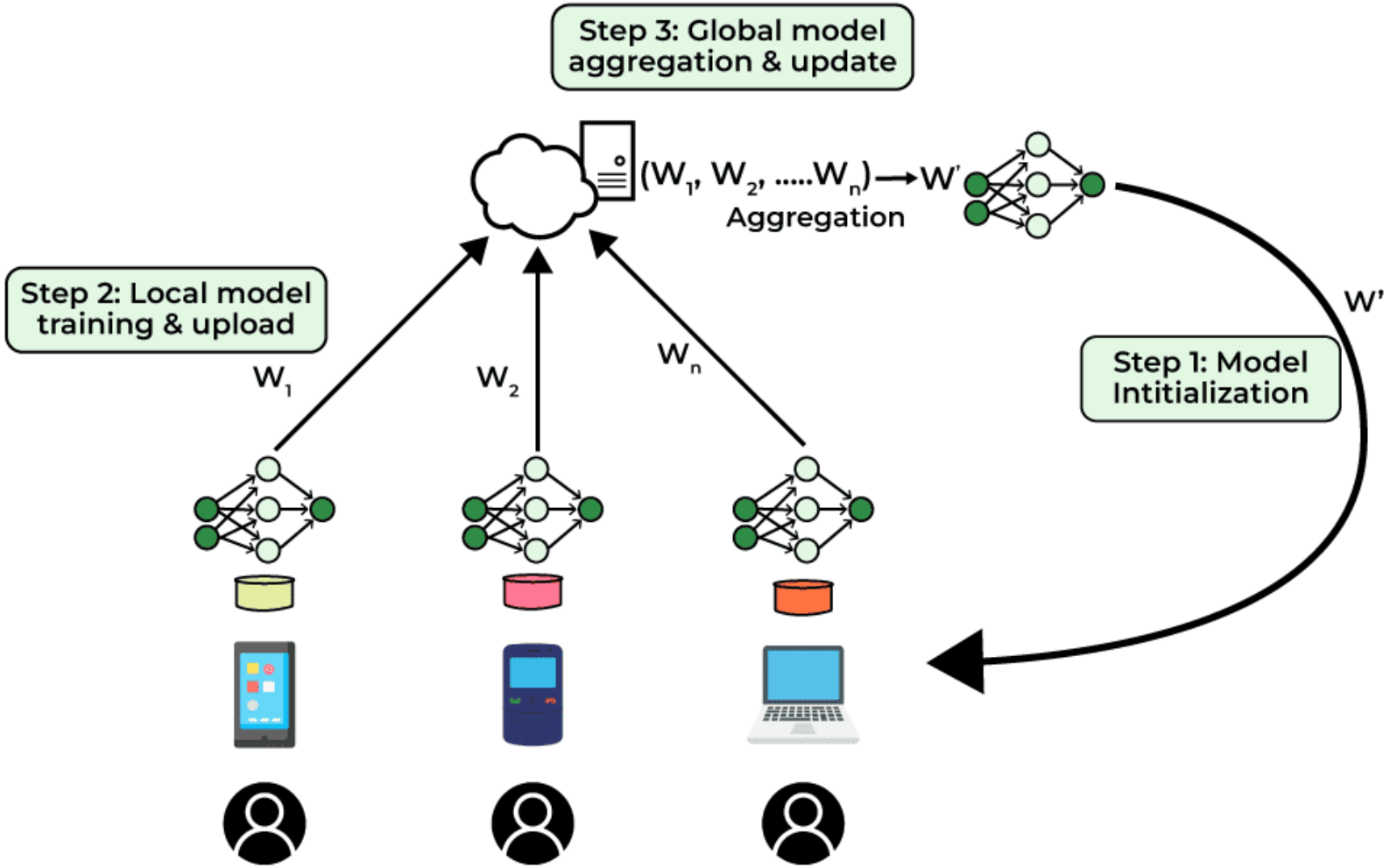
- $n = \sum_{k=1}^{K}$: Total data across all clients

FL Strategy Comparison

Criteria / Strategy	FedAvg	FedProx	FedNova
Simplicity	Very simple	Slighlt complex	More complex
Best for	Simple, IID	NonIID	NonIID
Use Cases	Image classification	Healthcare	Edge devices
Additional Parameters	No	Yes	Yes
Model Convergence	Fast	Moderate	Good
Handles Non-IID	Poor	Good	Good
Handles Unbalanced data	Moderate	Moderate	Excellent

Note: IID = Independent and Identically Distributed

Overview of Federated Learning



Algorithm of Federated Learning (FedAvg with SGD)

Inputs:

- K: Number of clients
- T: Number of communication rounds
- E: Number of local epochs
- η: Local learning rate
- $w^{(0)}$: Initial global model parameters

Algorithm:

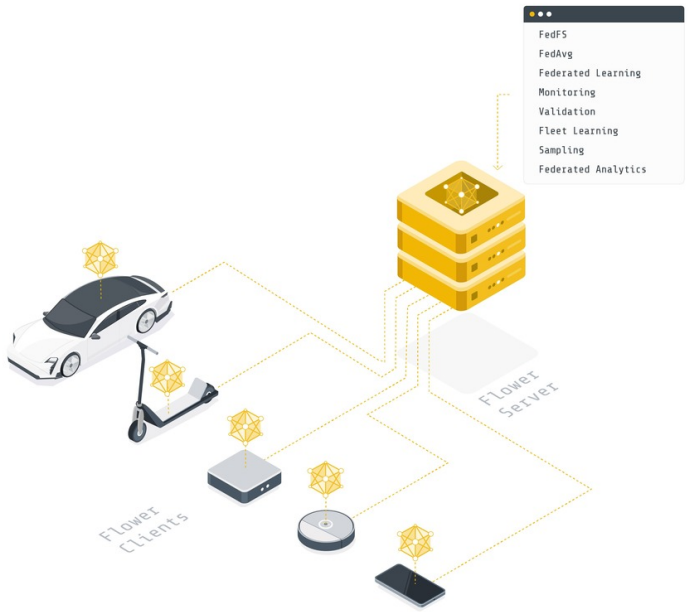
1. Initialize global model parameters $w^{(0)}$
2. **For** each round $t = 1$ to T **do**:
   1. Server selects a subset S_t of clients
   2. Server sends current global model $w^{(t-1)}$ to selected clients in S_t
   3. Each client $k \in S_t$ performs local training:
      - Initialize $w_k^{(t,0)} = w^{(t-1)}$
      - **For** local epoch $e = 1$ to E **do**:
        Update $w_k^{(t,e)}$ using local data and gradient descent:
        $$w_k^{(t,e)} \leftarrow w_k^{(t,e-1)} - \eta \nabla F_k(w_k^{(t,e-1)})$$
      - After E epochs, client obtains $w_k^{(t)} = w_k^{(t,E)}$
   4. Clients send $w_k^{(t)}$ back to the server
   5. Server aggregates updates to form new global model:
      $$w^{(t)} = \sum_{k \in S_t} \frac{n_k}{n_S} w_k^{(t)}$$
      where n_k is the number of data samples on client k, and $n_S = \sum_{k \in S_t} n_k$
3. **Return** final model $w^{(T)}$

Top Challenges in Federated Learning

1. Challenge: Non-IID data
   Description: Clients have different data distibutions
   Solutions: FedProx, FedNova
2. Challenge: Unbalanced Data
   Description: Varying amounts of data across clients
   Solutions: Adaptive aggregation
3. Challenge: Data leakage via updates
   Description: Gradient leak private info
   Solutions: Differential privacy, secure aggregation
4. Challenge: Cost
   Description: Expensive and slow
   Solutions: Model compressionm, fewer rounds
5. Challenge: Client Drift
   Description: Local models diverge too much
   Solutions: FedProx, SCAFFOLD

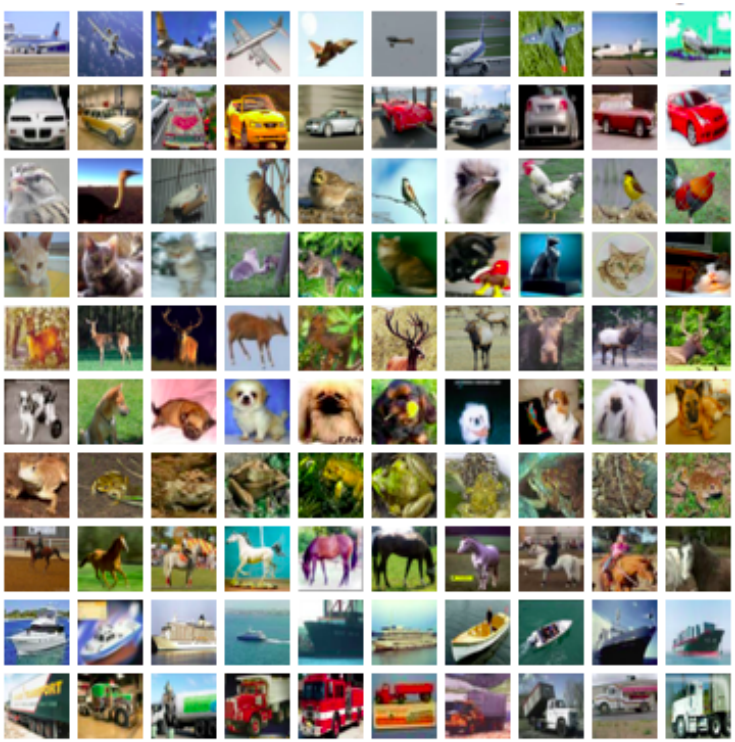
Flower

Federated learning require infrastructure to move machine learning models back and forth, train and evaluate them on local data, and then aggregate the updated models. Flower provides the infrastructure to do exactly that in an easy, scalable, and secure way. In short, Flower presents a unified approach to federated learning, analytics, and evaluation. It allows the user to federate any workload, any ML framework, and any programming language.



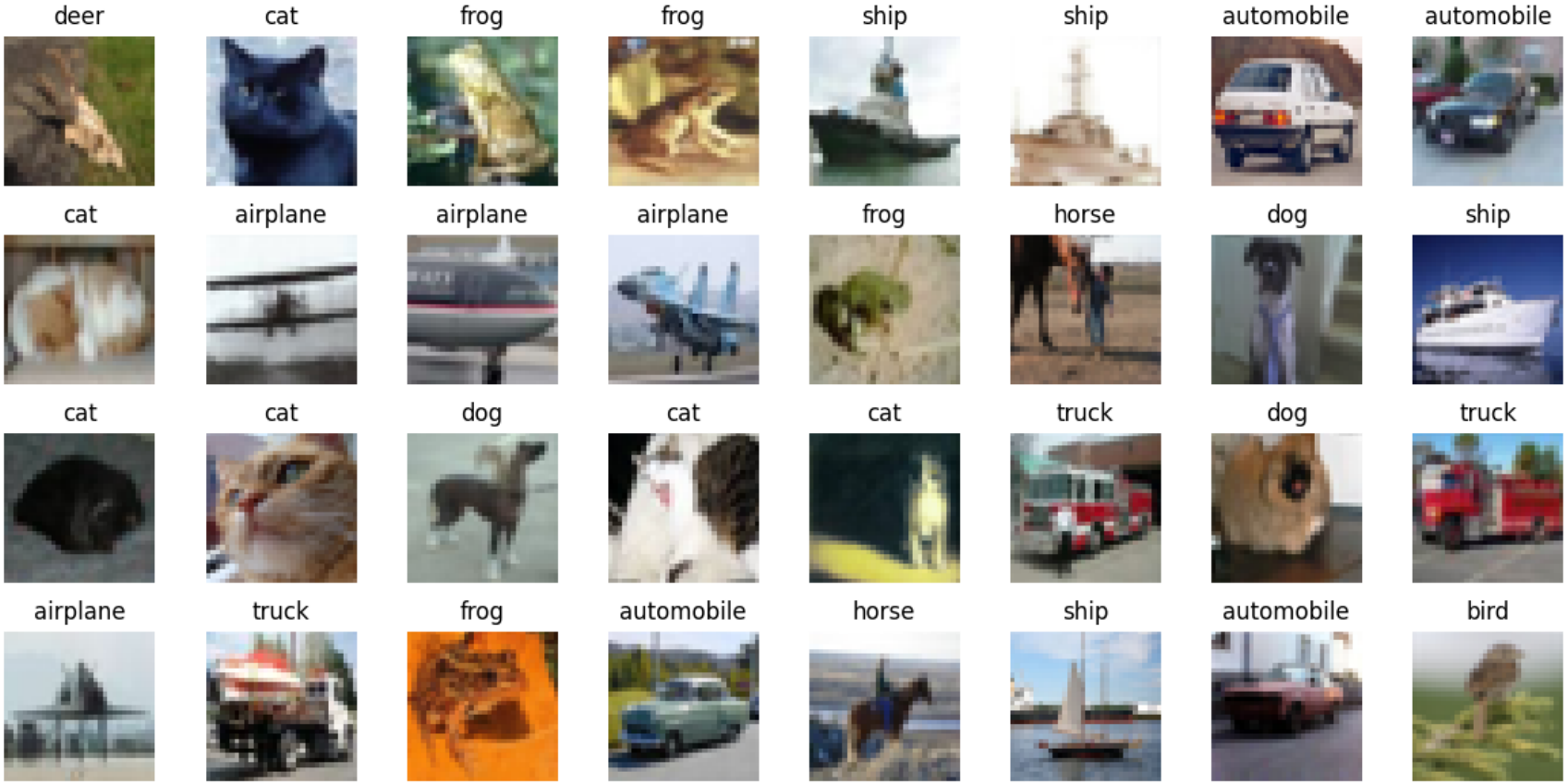
Application of Federated Learning

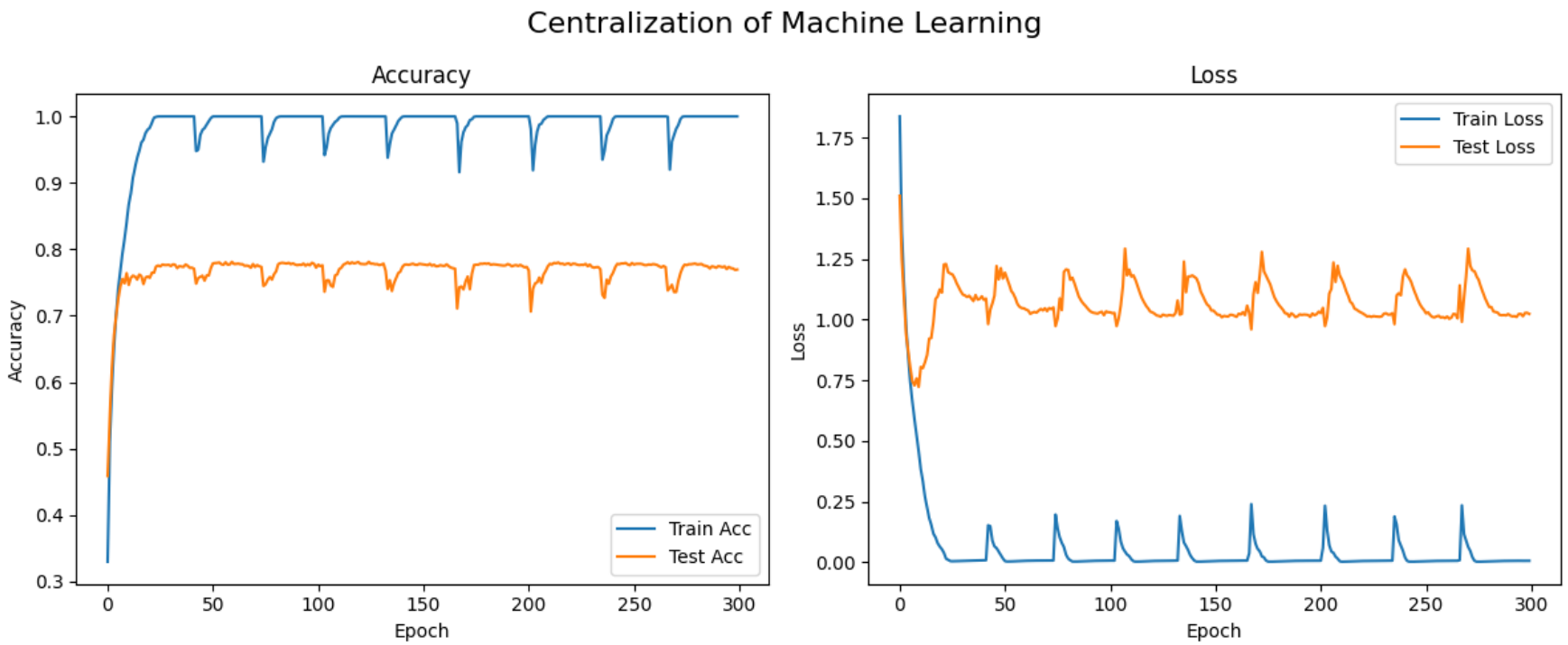
CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000, 32x32 color images containing one of 10 object classes, with 6000 images per class, i.e., 50,000 training images and 10,000 testing images.



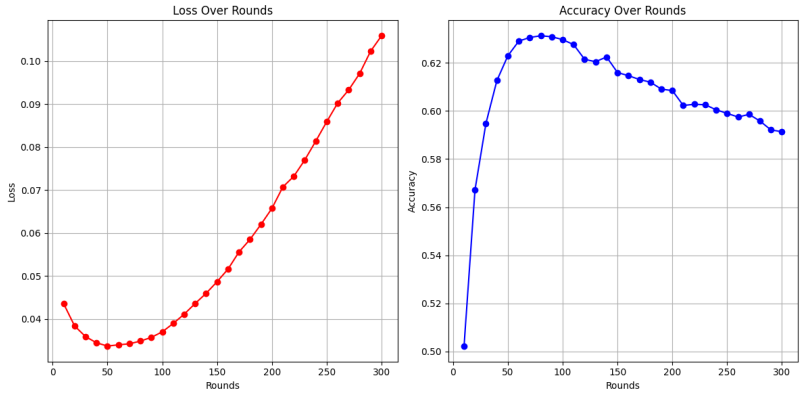
Classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck

Output

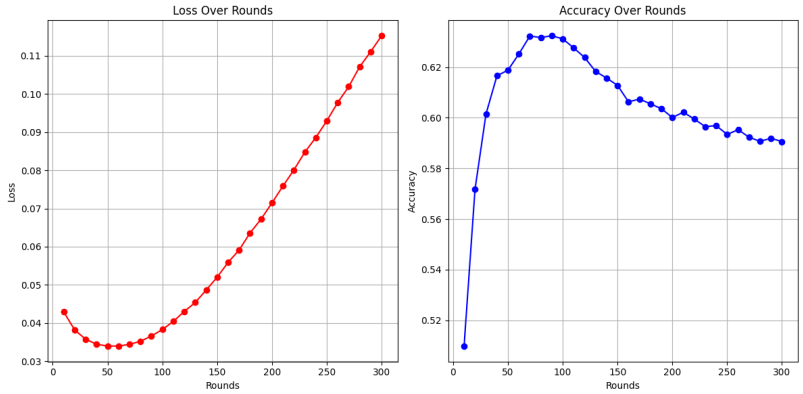

deer
cat
frog
frog
ship
ship
automobile
automobile
cat
airplane
airplane
airplane
frog
horse
dog
ship
cat
cat
dog
cat
cat
truck
dog
truck
airplane
truck
frog
automobile
horse
ship
automobile
bird


Centralization of Machine Learning
Accuracy
Accuracy
Epoch
Train Acc
Test Acc
Loss
Loss
Epoch
Train Loss
Test Loss

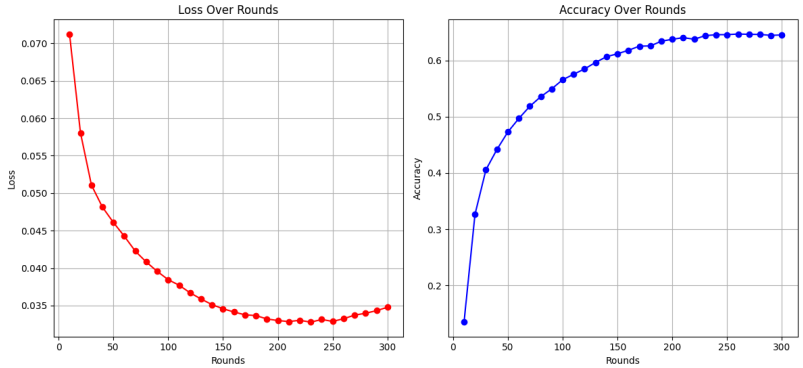
CNN-FedAvg-Adam-IID



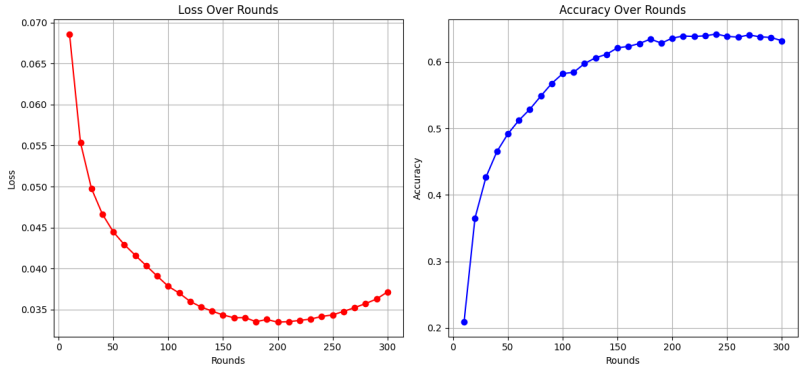
CNN-FedProx-Adam-IID



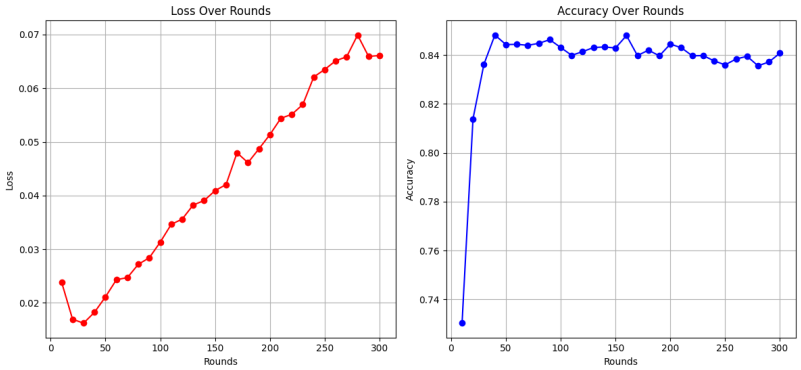
Net_FedProx_ProxSGD_IID



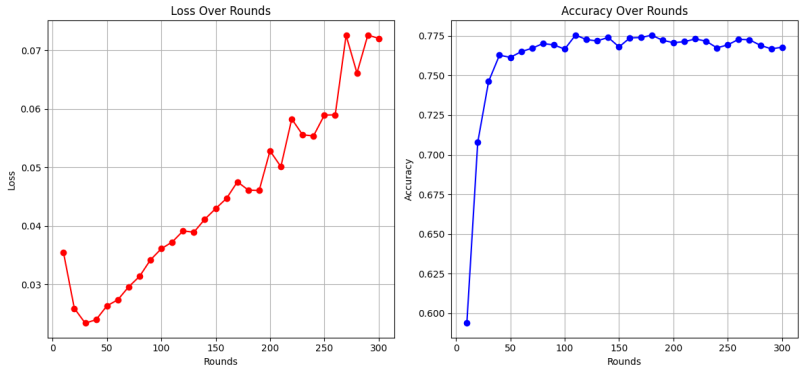
Net_FedAvg_ProxSGD_IID



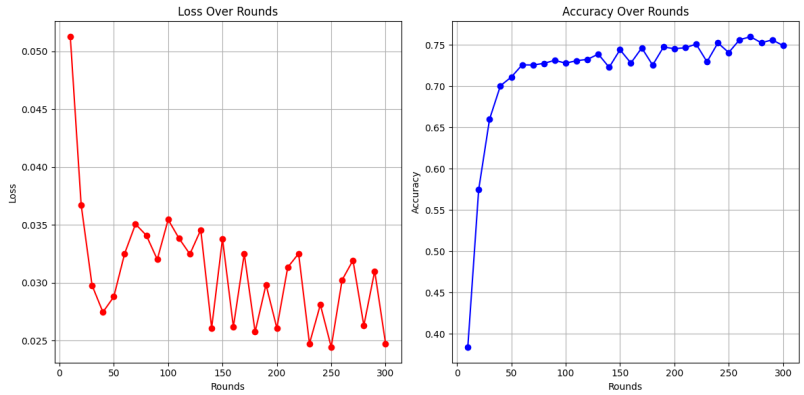
ResNet18_FedProx_Adam_IID



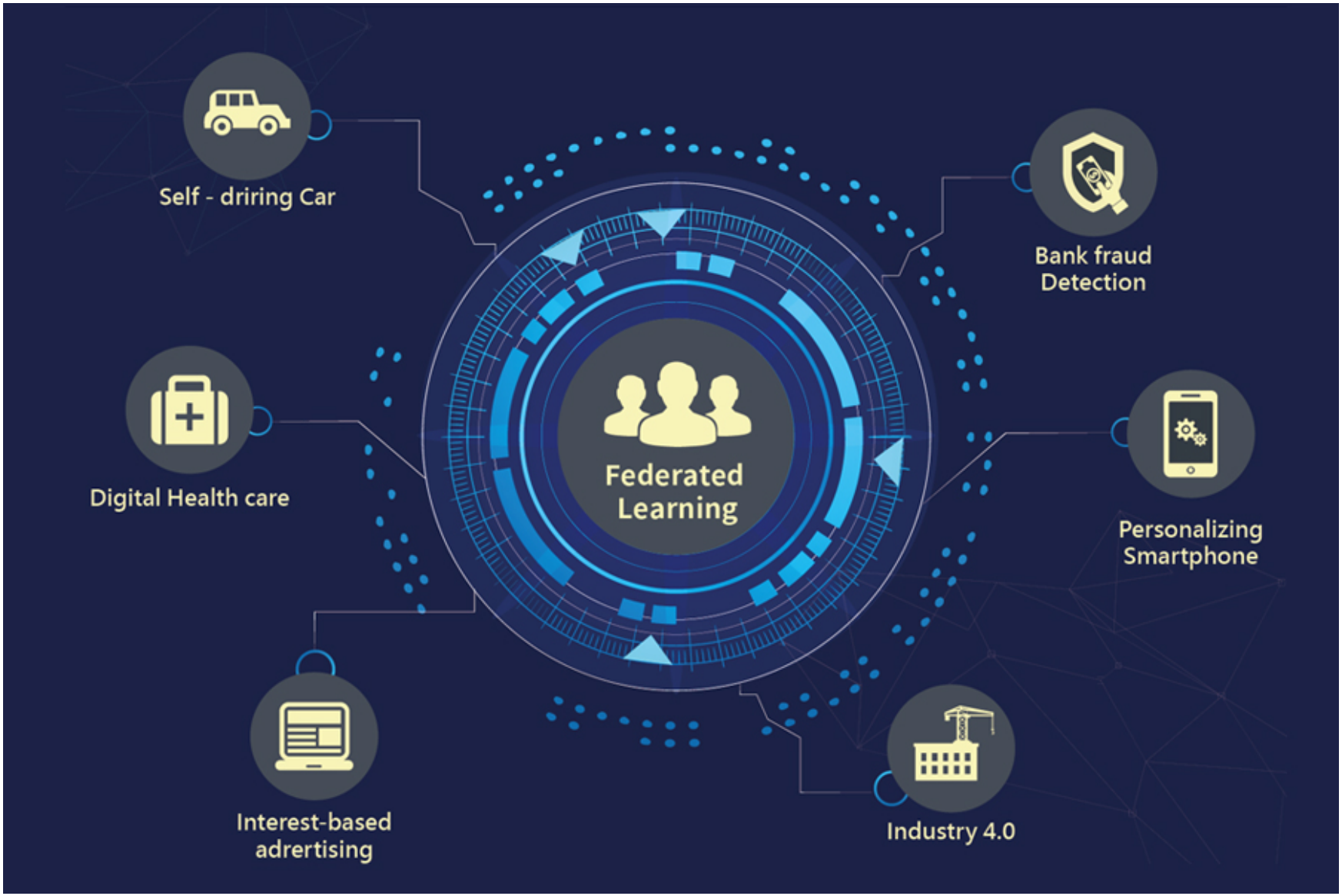
WideResNet_FedProx_Adam_IID



WideResNet-FedAvg-SGD-IID

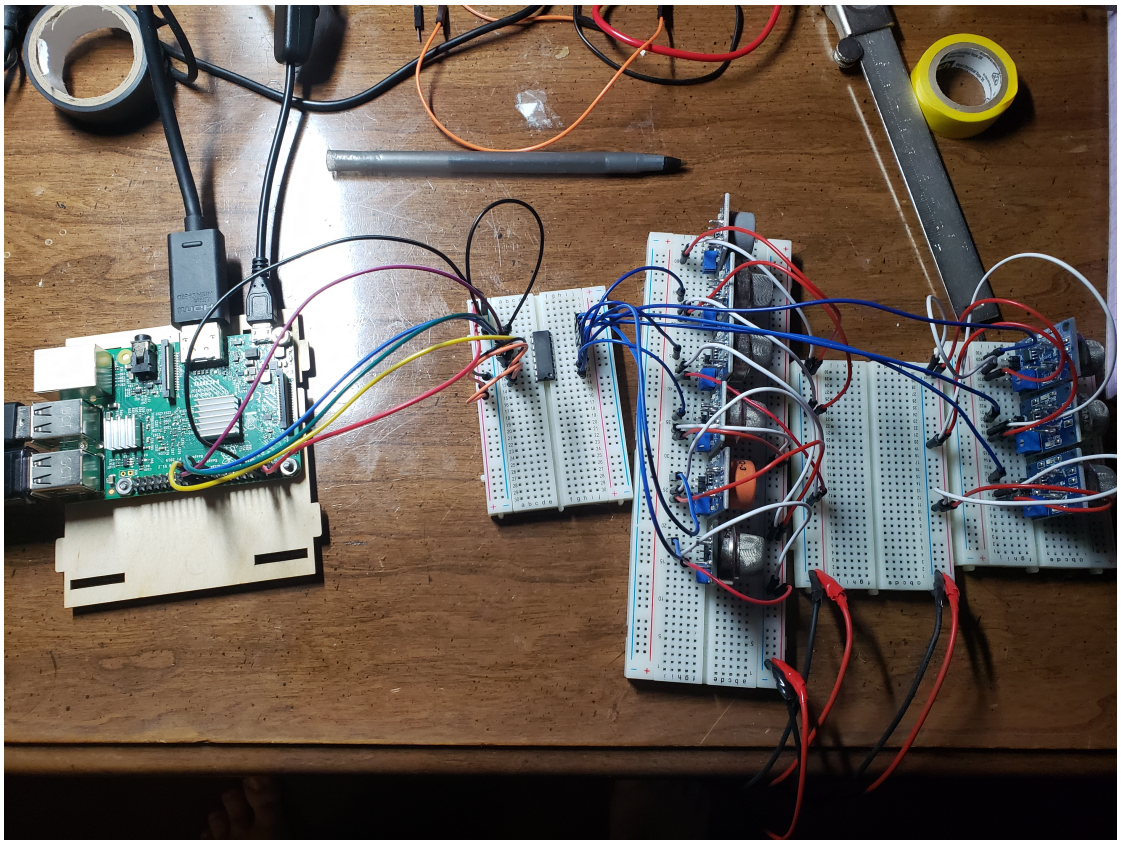


Some Applications



Future Plan

We focus on 'Application of Federated Learning in mining industry'. We will use several gas sensors to collect the data from the mining. After that, we will store dataset from sensors to Raberry Pi 5 as a client by connecting to computer as a server. In this connection, we will use federated learning to simulate the output to enhance safety, efficiency, and privacy-preserving intelligence in the mining industry.



References

[1]. Heiko Ludwig, Nathalie Baracaldo, *Federated Learning A Comprehensive Overview of Methods and Applications*, Springer, 2022.
[2]. Yaochu Jin ů Hangyu Zhu ů Jinjin Xu ů Yang Chen, *Federated Learning Fundamentals and Advances*, Springer, 2023.
[3]. Gethzi Ahila Poornima, Research Scholar , B. Paramasivan, Professor, *Anomaly detection in wireless sensor network using machine learning algorithm*, Computer Communications 151 (2020) 331337.
[4]. J. JITHISH , (Member, IEEE), BITHIN ALANGOT, NAGARAJAN MAHALINGAM , (Senior Member, IEEE), AND KIAT SENG YEO , (Fellow, IEEE), *Distributed Anomaly Detection in Smart Grids: A Federated Learning-Based Approach*, IEEE Access, 2023.
[5]. M. Victoria Luzón , Nuria Rodríguez-Barroso , Alberto Argente-Garrido , Daniel Jiménez-López , Jose M. Moyano , Javier Del Ser , Senior Member, IEEE, Weiping Ding , Senior Member, IEEE, and Francisco Herrera , Senior Member, IEEE, *A Tutorial on Federated Learning from Theory to Practice: Foundations, Software Frameworks, Exemplary Use Cases, and Selected Trends*, IEEE/CAA JOURNAL OF AUTOMATICA SINICA, VOL. 11, NO. 4, APRIL 2024.

[6]. Mirko Nardia, Lorenzo Valeriob, Andrea Passarella, *Federated Clustering: An Unsupervised Cluster-Wise Training for Decentralized Data Distributions*, arXiv:2408.10664v1 [cs.LG] 20 Aug 2024.
[7]. Luigi Palmieria, Chiara Boldrinia, Lorenzo Valerioa, Andrea Passarellaa, Marco Contia, J anos Kert eszb, *Robustness of Decentralised Learning to Nodes and Data Disruption*, arXiv:2405.02377v1 [cs.LG] 3 May 2024.
[8]. Samuele Sabella, Chiara Boldrini, Lorenzo Valerioa, Andrea Passarella, Marco Conti, *The Built-In Robustness of Decentralized Federated Averaging to Bad Data*, arXiv:2502.18097v1 [cs.LG] 25 Feb 2025.
[9]. Luigi Palmieri, Chiara Boldrini1, Lorenzo Valerio1, Andrea Passarella, Marco Conti, *Impact of network topology on the performance of Decentralized Federated Learning* , Computer Networks 253 (2024) 110681.


END OF PRESENTATION
THANKYOU FOR LISTENING
makeameme.org