Introduction to Federated Learning at IoT Edge

Dr. Rajiv Misra, **Professor**
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Preface

After Completion of this lecture you will knowing the following:

- Current IoT scenarios
- Why there is a need to shift from centralized ML training to decentralized ML training of data?
- Concepts of Federated Learning (ie Distributed ML)
- Several challenges of federated learning

Current IoT Scenario

Explosion of IoT Market

- McKinsey reported $11.1 Trillion market value by 2025
- 14 billion connected devices - Bosch
- 5 billion connected devices - Cisco
- 309 billion IoT supplier revenue - Gartner
- 7.1 trillion IoT solutions revenue - IDC

A “deluge of data” is observed in 2020

1.5 GB of traffic per day from average internet user

3000 GB per day - Smart Hospitals

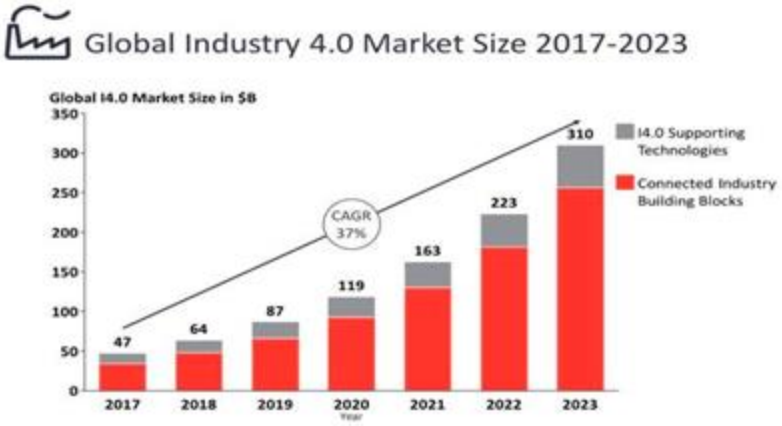
4000 GB data per day - self driving cars EACH

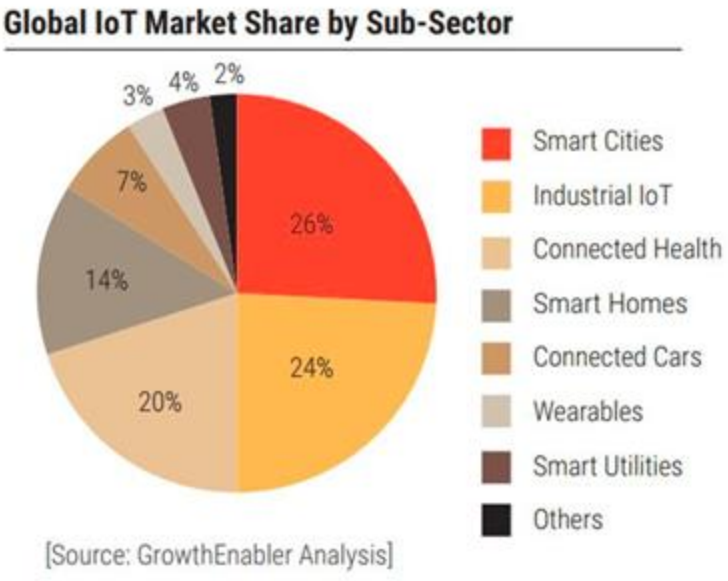
Radars ~ 10-100 kb per sec

40,000 GB per day - connected aircrafts

1,000,000 GB per day - connected factories

[Sources: GrowthEnabler Analysis/MarketsandMarkets]

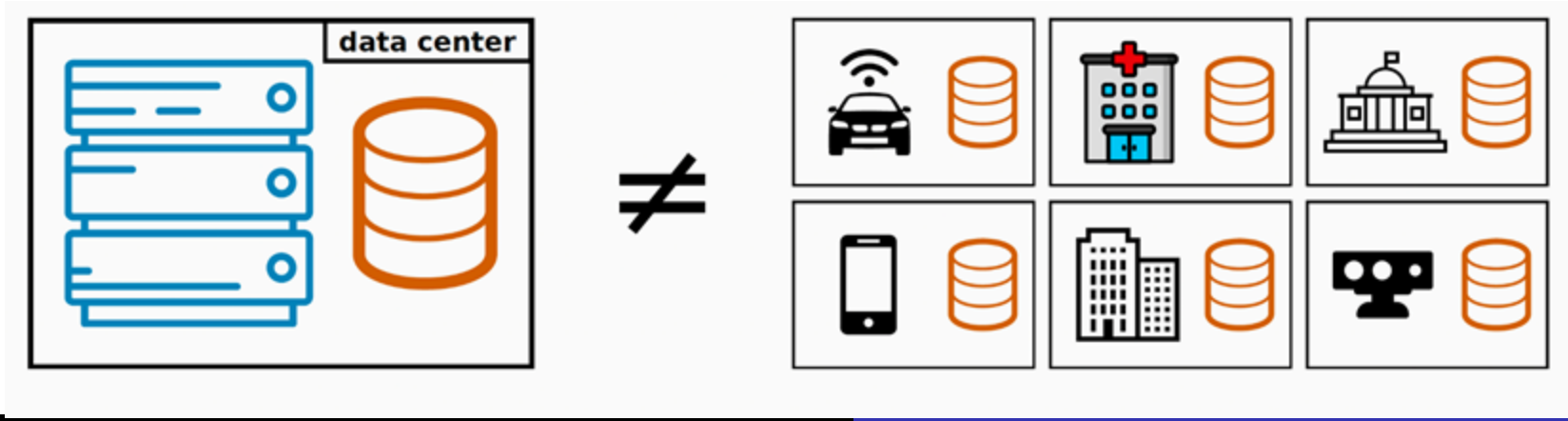




[Source: GrowthEnabler Analysis]

Shift from Centralized to Decentralized data

- The standard setting in Machine Learning (ML) considers a centralized dataset processed in a tightly integrated system
- But in the real world data is often decentralized across many IOT devices
- Sending the data tpo Cloud for centralized ML may be too costly
  - Self-driving cars are expected to generate several TBs of data a day
  - Some wireless devices have limited bandwidth/power
- Data may be considered too sensitive sometimes such as medical reports
  - We see a growing public awareness and regulations on data privacy
  - Keeping control of data can give a competitive advantage in business and research

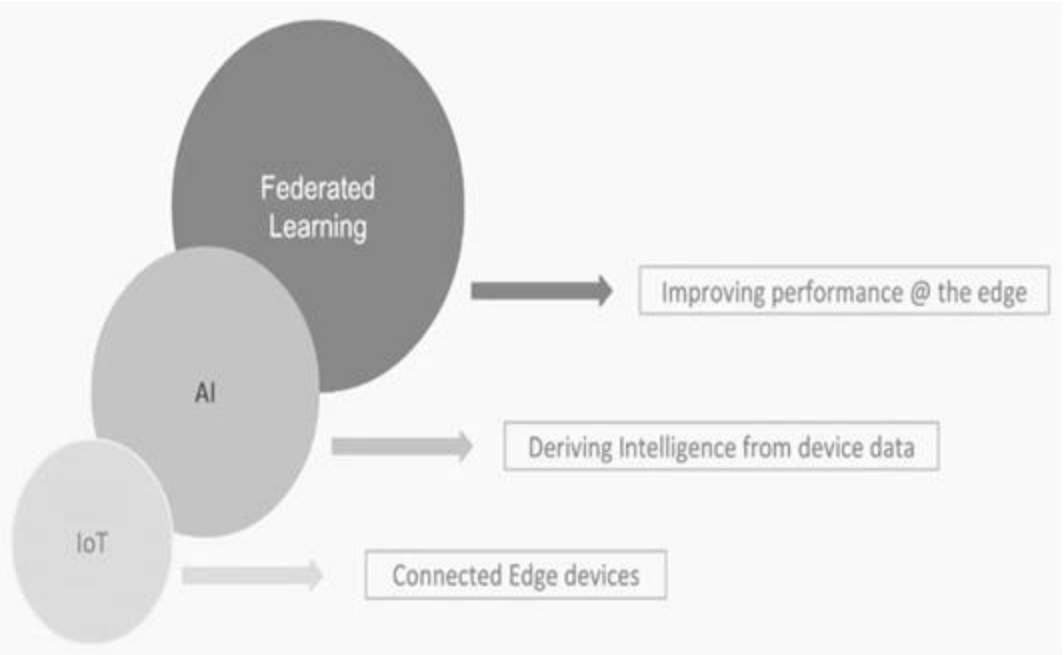


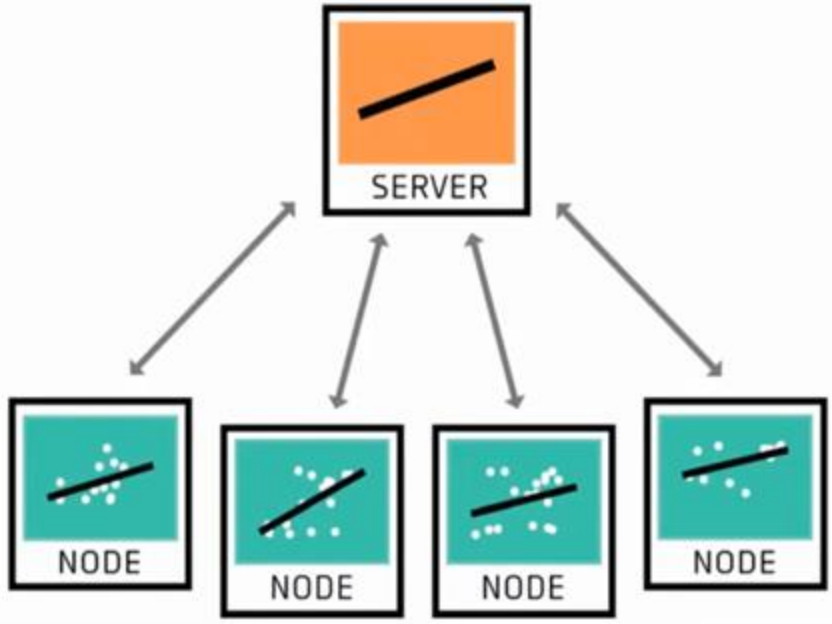
Federated Learning: Distributed ML

- 2016: the term FL is first coined by Google researchers; 2020: more than 1,000 papers on FL in the first half of the year (compared to just 180 in 2018)1
- We have already seen some real-world deployments by companies and researchers for large scale IOT devices
- Several open-source libraries are under development: PySyft, TensorFlow Federated, FATE, Flower, Substra...
- FL is highly multidisciplinary: it involves machine learning, numerical optimization, privacy & security, networks, systems, hardware…

Federated Learning: Decentralised data

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized
- Enabling devices to learn from each other (ML training is brought close
- A network of nodes and all nodes with their own central server but instead of sharing data with the central server, we share model we don't send data from node to server instead send our model to server





Gradient Descent Procedure

The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value.

coefficient = 0.0

The cost of the coefficients is evaluated by plugging them into the function and calculating the cost.

cost = f(coefficient) or cost = evaluate(f(coefficient))

We need to know the slope so that we know the direction (sign) to move the coefficient values in order to get a lower cost on the next iteration.

delta = derivative(cost)

we can now update the coefficient values.

A learning rate parameter (alpha) must be specified that controls how much the coefficients can change on each update.

coefficient = coefficient – (alpha * delta)

This process is repeated until the cost of the coefficients (cost) is 0.0 or close to 0 It does require you to know the gradient of your cost function or the function you are optimizing

Gradient Descent Algorithm

Gradient Descent

- Gradient Descent is the most basic but most used optimization algorithm. It's used heavily in linear regression and classification algorithms. Backpropagation in neural networks, Federated Learning also uses a gradient descent algorithm.
- Gradient descent is a first-order optimization algorithm which is dependent on the first order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.

$$algorithm: \vartheta = \vartheta - \alpha \cdot \nabla J(\vartheta)$$

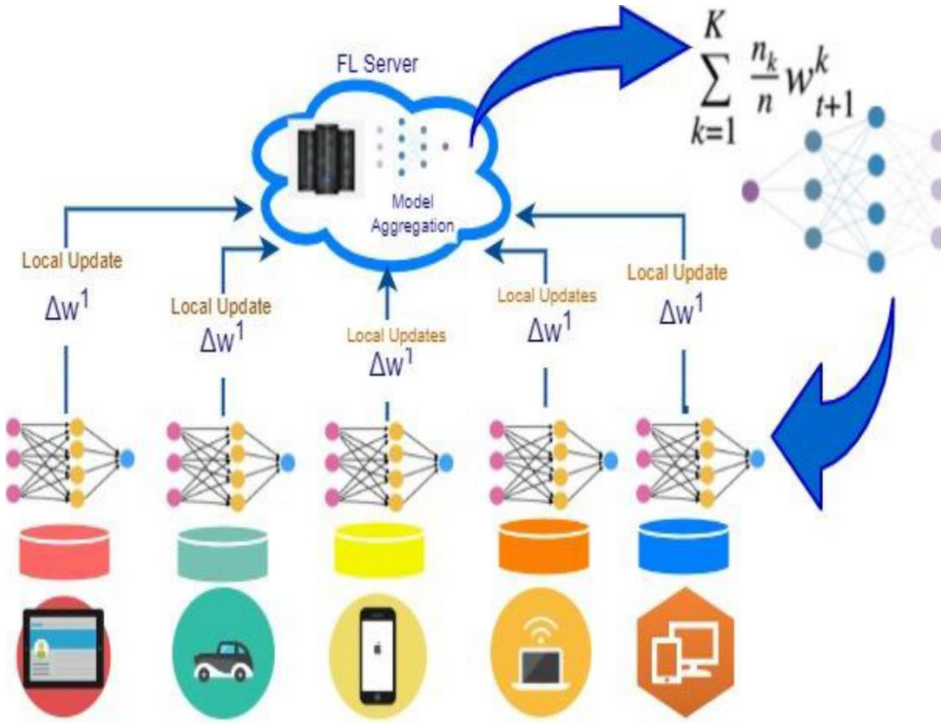
Advantages:

- Easy computation
- Easy to implement
- Easy to understand

The devices train the generic neural network model using the gradient descent algorithm, and the trained weights are sent back to the server. The server then takes the average of all such updates to return the final weights.

Edge Computing ML: FL

- FL is category of machine learning (ML) , which moves the processing over the edge nodes so that the clients' data can be maintained. This approach is not only a precise algorithm but also a design framework for edge computing.
- Federated learning is a method of ML that trains an ML algorithm with the local data samples distributed over multiple edge devices or servers without any exchange of data. This term was first introduced in 2016 by McMahan.
- Federated learning distributes deep learning by eliminating the necessity of pooling the data into a single place.
- In FL, the model is trained at different sites in numerous iterations. This method stands in contrary to other conventional techniques of ML, where the datasets are transferred to a single server and to more traditional decentralized techniques that undertake that local datasets



Edge Computing ML: FL

Finding the function: model training

- Given a training dataset containing n input-output pairs $(x_i, y_i), i \in [1, n]$, the goal of deep learning model training is to find a set of parameters w, such that the average of $p(y_i)$ is maximized given x_i.

- That is,

$$maximize \quad \frac{1}{n}\sum_{i=1}^{n} p(y_i|x_i, w)$$

Which is equivalent to

$$minimize \quad \frac{1}{n}\sum_{i=1}^{n} -\log(p(y_i|x_i, w))$$

A basic component for loss function $l(x_i, y_i, w)$ given sample (x_i, y_i):

Let $f_i(w) = l(x_i, y_i, w)$ denote the loss function.

Deep Learning model training

- Given one input sample pair (x_0, y_0), the goal of deep learning model training is to find a set of parameters w, to maximize the probability of outputting y_0 given x_0.

For a training dataset containing n samples $(x_i, y_i), 1 \leq i \leq n$, the training objective is:

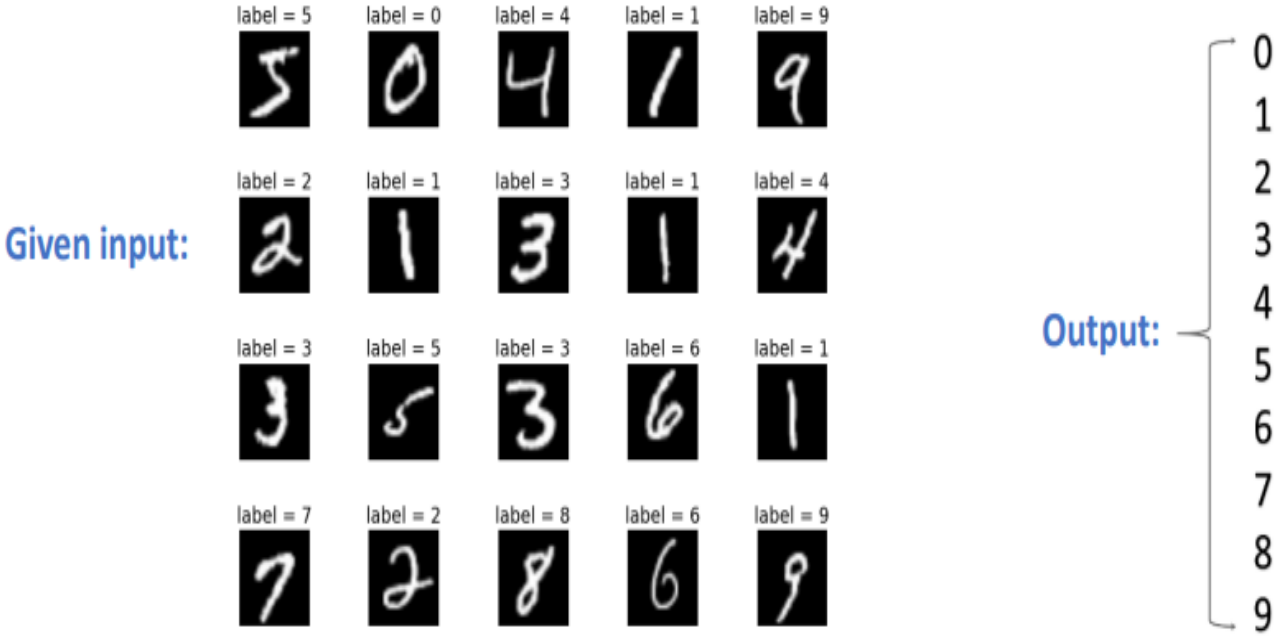
$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where } f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

$f_i(w) = l(x_i, y_i, w)$ is the loss of the prediction on example (x_i, y_i)

Edge Computing ML: FL

Finding the function: model training

- Given one input sample pair (x_0, y_0), the goal of deep learning model training is to find a set of parameters w, to maximize the probability of outputting y_0 given x_0.



How is this aggregation applied? FedAvg Algo

Federated learning – *FederatedAveraging (FedAvg)*

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

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

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

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

1. At first, a model is randomly initialized on the central server.
2. For each round *t:*
   - *i. A random set of clients are chosen;*
   - *ii. Each client performs local gradient descent steps;*
   - *iii. The server aggregates model parameters submitted by the clients.*

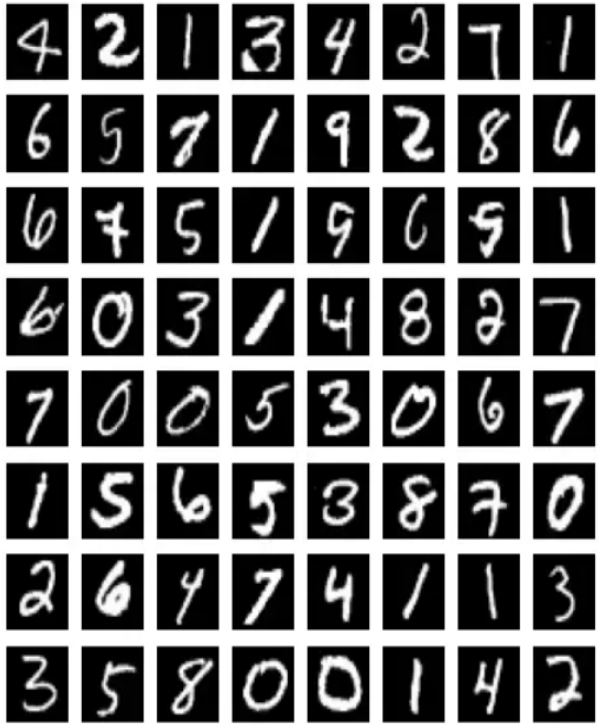
Example: FL with i.i.d.

In FL, each client trains its model decentral. In other words, the model training process is carried out separately for each client.

Only learned model parameters are sent to a trusted center to combine and feed the aggregated main model. Then the trusted center sent back the aggregated main model back to these clients, and this process is circulated.

A simple implementation with IID (independent and identically distributed) data to show how the parameters of hundreds of different models that are running on different nodes can be combined with the FedAvg method and whether this model will give a reasonable result.

This implementation was carried out on the MNIST Data set. The MNIST data set contains 28 * 28 pixel grayscale images of numbers from 0 to 9.



Handwritten Digits from the MNIST datas

Image Classifier using FedAvg

The MNIST data set does not contain each label equally. Therefore, to fulfill the IID requirement, the dataset was grouped, shuffled, and then distributed so that each node contains an equal number of each label.

A simple 2-layer model can be used for the classification process used FedAvg. Since the parameters of the main model and parameters of all local models in the nodes are randomly initialized, all these parameters will be different from each other, so the main model sends its parameters to the nodes before the training of local models in the nodes begins.

Nodes start to train their local models over their own data by using these parameters. Each node updates its parameters while training its own model. After the training process is completed, each node sends its parameters to the main model.
The main model takes the average of these parameters and sets them as its new weight parameters and passes them back to the nodes for the next iteration.
The above flow is for one iteration. This iteration can be repeated over and over to improve the performance of the main model.
The accuracy of the centralized model was calculated as approximately 98%. The accuracy of the main model obtained by FedAvg method started from 85% and improved to 94%.

Apple personalizes Siri without hoovering up data

The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

It relies primarily on a technique called federated learning.

It allows Apple to train different copies of a speaker recognition model across all its users' devices, using only the audio data available locally.

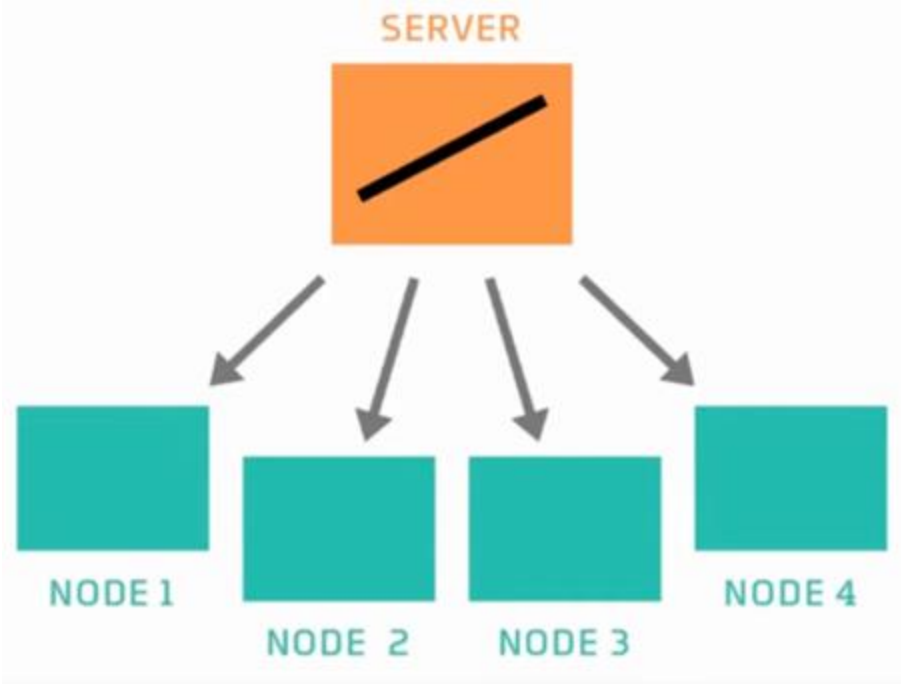
It then sends just the updated models back to a central server to be combined into a master model.

In this way, raw audio of users' Siri requests never leaves their iPhones and iPads, but the assistant continuously gets better at identifying the right speaker. In addition to federated learning, Apple also uses something called differential privacy to add a further layer of protection. The technique injects a small amount of noise into any raw data before it is fed into a local machine-learning model. The additional step makes it exceedingly difficult for malicious actors to reverse-engineer the original audio files from the trained model.

Siri is a perfect example of how Apple runs on AI. The voice-powered assistant is designed for continual, at-the-edge improvement

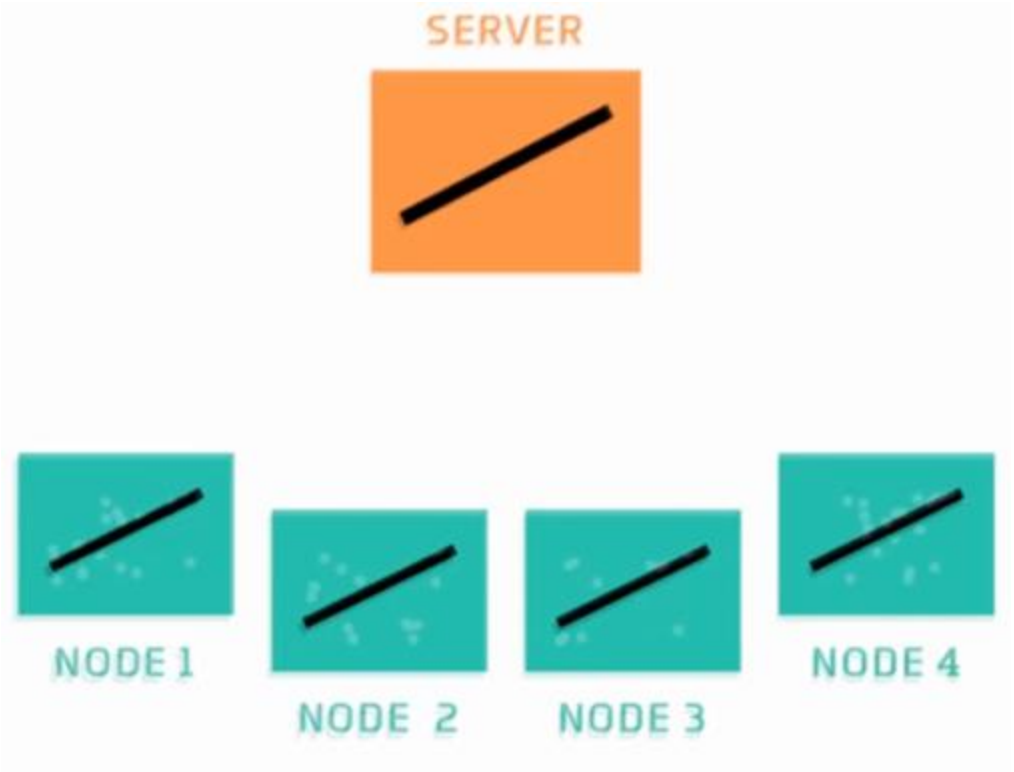
Federated Learning: Training

- There are connected devices let's say we have cluster of four IOT Devices from four of the IOT devices and there is one central server that has an untrained model.
- We will send a copy of the model to each of the node.
- Each node would receive a copy of that model.



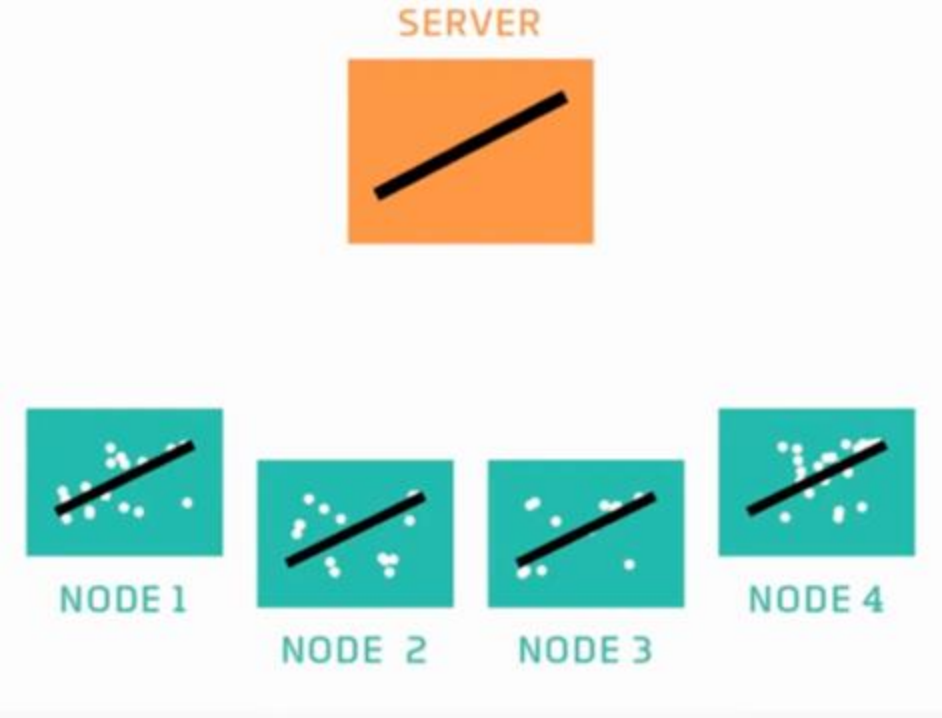
Federated Learning: Training

- Now all the nodes in the network has that untrained model that is received from the server.



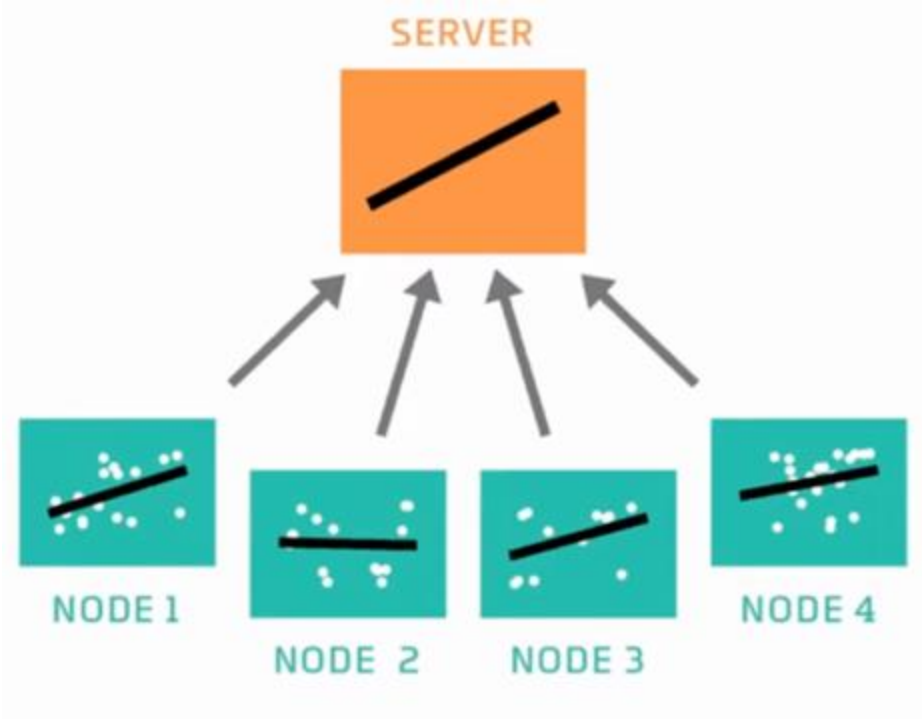
Federated Learning: Training

- In the next step, we are taking data from each node by taking data it doesn't mean that we are sharing data.
- Every node has its own data based on which it is going to train a model.



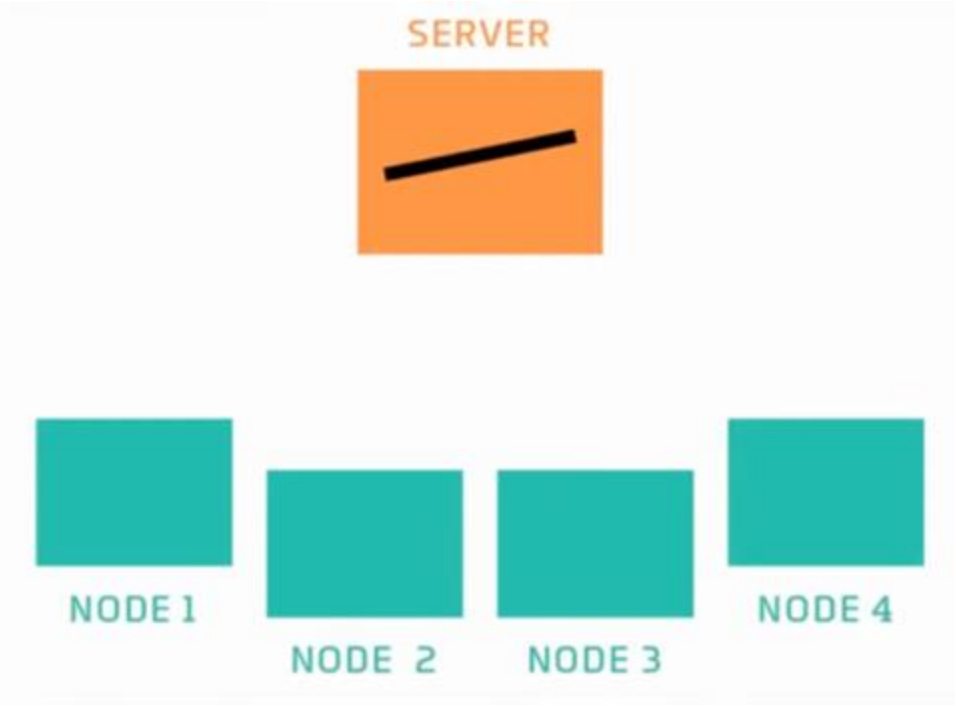
Federated Learning: Training

- Each node is training the model to fit the data that they have and it will train the model accordingly to its data.



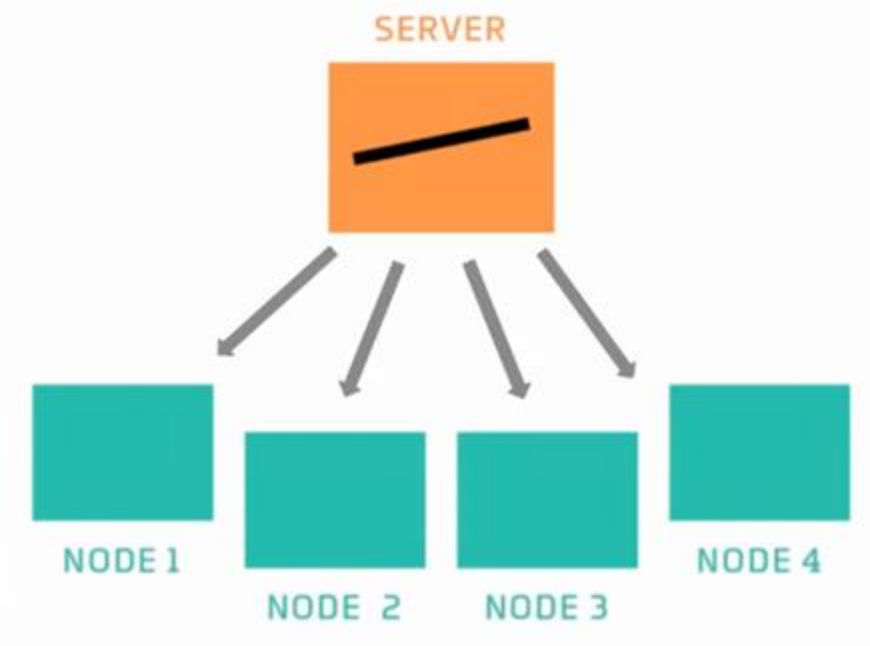
Federated Learning: Training

- Now the server would combine all these model received from each node by taking an average or it will aggregate all the models received from the nodes.
- Then the server will train that a central model, this model which is now trained by aggregating the models from each node. It captures the pattern in the training data on all the nodes it is an aggregated one



Federated Learning: Training

- Once the model is aggregated, the server will send the copy of the updated model back to the nodes.
- Everything is being achieved at the edge so no data sharing is done which means there is privacy preservation and also very less communication overhead.



Federated Learning: Challenges

Systems heterogeneity

- Size of data
- Computational power
- Network stability
- Local solver
- Learning rate

Expensive Communication

- Communication in the network can be slower than local computation by many order of magnitude.

Statistical Heterogeneity:

Fig. 1: Federated learning with non-iid data - The data has different distributions among clients.

Federated Learning: Challenges

Dealing with Non-I.I.D. data i.i.d (independent and identical distributed)

- Learning from non-i.i.d. data is difficult/slow because each IOT device needs the model to go in a particular direction
- If data distributions are very different, learning a single model which performs well for all IOT devices may require a very large number of parameters
- Another direction to deal with non-i.i.d. data is thus to lift the requirement that the learned model should be the same for all IOT devices ("one size fits all")
- Instead, we can allow each IOT k to learn a (potentially simpler) personalized model θk but design the objective so as to enforce some kind of collaboration
- When local datasets are non-i.i.d., FedAvg suffers from client drift
- To avoid this drift, one must use fewer local updates and/or smaller learning rates, which hurts convergence

Federated Learning: Challenges

Preserving Privacy

- ML models are susceptible to various attacks on data privacy
- Membership inference attacks try to infer the presence of a known individual in the training set, e.g., by exploiting the confidence in model predictions
- Reconstruction attacks try to infer some of the points used to train the model, e.g., by differencing attacks
- Federated Learning offers an additional attack surface because the server and/or other clients observe model updates (not only the final model)

Key differences with Distributed Learning

Data distribution

- In distributed learning, data is centrally stored (e.g., in a data center)
  - The main goal is just to train faster
  - We control how data is distributed across workers: usually, it is distributed uniformly at random across workers
- In FL, data is naturally distributed and generated locally
  - Data is not independent and identically distributed (non-i.i.d.), and it is imbalanced

Additional challenges that arise in FL

- Enforcing privacy constraints
- Dealing with the possibly limited reliability/availability of participants
- Achieving robustness against malicious parties

Federated Learning: Concerns

When to apply Federated Learning

- Data privacy needed
- Bandwidth and power consumptions are concerns
- High cost of data transfer

When NOT to apply Federated Learning

- When more data won't improve your model (construct a learning cure)
- When additional data is uncorrelated
- Performance is already at ceiling

Federated Learning: Applications

- Predictive maintenance/industrial IOT
- Smartphones
- Healthcare (wearables, drug discovery, prognostics, etc.)
- Enterprise/corporate IT (chat, issue trackers, emails, etc.)

Lecture Summary

- Market trend of IoT platform
- Why decentralized training is important?
- Understanding of Federated Learning
- Different issues with federated learning


THANK YOU!