Federated learning

Classical Machine learning

- In machine learning, we train the model using the data to detect objects, transcribe an audio recording or play a game like Go. The data we use gets generated “somewhere else”. For instance, the data can originate on a smartphone by the user interacting with an app, etc. The approach is to collect all the data available broadly on a central server. This server can be located somewhere in a data center or cloud. Once all the data is collected in one place, we use machine learning algorithms to train our model on the data.

➢ Challenges of classical machine learning

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, we are not able to use classical ml.

Examples where centralized machine learning does not work includes:

- Sensitive healthcare records from multiple hospitals to train cancer detection models.
- Financial information from different organizations to detect financial fraud.
- Location data from your electric car to make better range prediction.
- End-to-end encrypted messages to train better auto-complete models.

Federated Learning

Federated Learning enables machine learning on distributed data by moving the training to the data instead of moving the data to the training.

- *Centralized machine learning: move the data to the computation*
- *Federated (machine) Learning: move the computation to the data*

Federated learning in five steps

Step 0: Initialize global model

- ➔ Start by initializing the model on the server (Initialize model parameters either randomly or through some saved data)

Step 1: Send model to a number of connected organizations/devices (client nodes)

- ➔ Send parameters of the global model to the connected client nodes. It ensures that each participating node starts its local training using the same model parameters. We often use only a few of the connected nodes instead of all nodes.

Step 2: Train model locally on the data of each organization/device (client node)

- ➔ Now selected client nodes have the latest version of the global model parameters then they start the local training. They use their own local dataset to train their own local model. They don't train the model until full convergence, just be one epoch on the local data or mini-batches.

Step 3: Return model updates back to the server

- ➔ After local training, each client node has a different version of the model parameters. The client nodes then send those model updates back to the server. The model updates they send can either be the full model parameters or just the gradients.

Step 4: Aggregate model updates into a new global model

- ➔ The server receives model updates from the selected client nodes. In order to get one single model, we have to combine all the model updates received from the client nodes. This process is called **Aggregation**. The most basic way is called Federated Averaging, often abbreviated as **FedAvg**.

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

The global model parameters get sent to the participating client nodes (step 1), the client nodes train on their local data (step 2), they send their updated models to the server (step 3) and the server then aggregates the model updates to get a new version of the global model (step 4).

- ➔ We then have to repeat this training process over and over again to eventually arrive at a fully trained model that performs well across the data of all client nodes.

Convergence = Model fits the data well, parameters are near optimal, loss is minimized and accuracy is no longer improving significantly.

Federated Evaluation

- We can evaluate the model on that data to receive valuable metrics. This is called federated evaluation, sometimes abbreviated as FE.

Differential Privacy

- Differential privacy (DP) is often mentioned in the context of Federated Learning. It is a privacy-preserving method used when analyzing and sharing statistical data. DP adds statistical noise to the model updates.

Flower (Federated AI Framework)

Federated learning, federated evaluation and federated analytics require infrastructure to move machine learning models back and forth, train and evaluate and then aggregate the updated models. Flower presents a unified approach to federated learning, analytics and evaluation.

CIFAR-10 DATASET TRAINING

- https://colab.research.google.com/drive/1qTFnaIewrIcZ0SIxOceXYNRm02j_TW3A?usp=sharing
  - Reference : https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

CIFAR-10 DATASET TRAINING WITH NOISE

- https://colab.research.google.com/drive/1cyBjKtimAX5pBvten4STovRgCHLaAIhh?usp=sharing

CIFAR-10 DATASET TRAINING WITH FLOWER (FEDERATED LEARNING)

- https://colab.research.google.com/drive/1R9fPanS6dspKH31jfxrxEdbR2oN7RYdp?usp=sharing
  - Reference : https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html

- https://colab.research.google.com/drive/1Wl-CgqX7CU3vwvChx_zmJsC8J2I6MmNO?usp=sharing ***(Dirichlet Partition)***
  - Reference : https://flower.ai/docs/datasets/ref-api/flwr_datasets.partitioner.DirichletPartitioner.html#flwr_datasets.partitioner.DirichletPartitioner

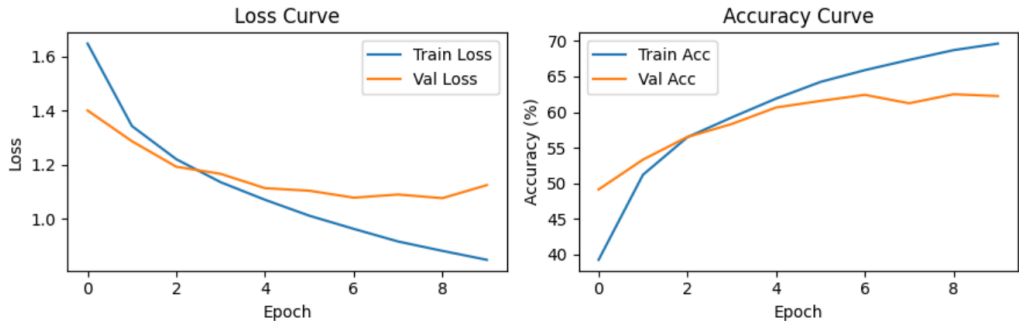
- https://colab.research.google.com/drive/1KH9o74DNOKOvUQFFjpiS5gZL3HInsYUt?usp=sharing ***(Pathological Partition)***
  - Reference : https://flower.ai/docs/datasets/ref-api/flwr_datasets.partitioner.PathologicalPartitioner.html#flwr_datasets.partitioner.PathologicalPartitioner

CIFAR-10 DATASET TRAINING KEY POINTS :

★ *CNN Network used for model training :*

Step	Operation	Shape of x
Input	Raw images	(32, 3, 32, 32)
Conv1	self.conv1 = nn.Conv2d(3, 6, 5)	(32, 6, 28, 28)
ReLU	Activation	(32, 6, 28, 28)
Pool1	self.pool(x)	(32, 6, 14, 14)
Conv2	self.conv2 = nn.Conv2d(6, 16, 5)	(32, 16, 10, 10)
ReLU	Activation	(32, 16, 10, 10)
Pool2	self.pool(x)	(32, 16, 5, 5)
Flatten	.view(-1, 16*5*5)	(32, 400)
FC1	Fully connected (Linear)	(32, 120)
ReLU	Activation	(32, 120)
FC2	Fully connected (Linear)	(32, 84)
ReLU	Activation	(32, 84)
FC3	Output layer	(32, 10)

★ *Plots :*



CIFAR-10 DATASET TRAINING WITH NOISE KEY POINTS :

★ *Difference between Gaussian & Activation Noise :*

Difference Between Gaussian Noise and Activation Noise

Type of Noise	Where is it Added?	Purpose	Example Use Case
Gaussian Noise (Input Noise)	Directly to the **input data (images)**	To simulate data corruption, improve robustness, and regularization	Like making the model resilient to blurry/noisy inputs in real-world scenarios
Activation Noise (Internal Noise)	Added to **intermediate layers (activations)** inside the model	Acts like internal regularization (similar to Dropout), helps prevent overfitting, encourages learning more general features	Used to stabilize learning, improve generalization, especially in noisy environments

❖ So, the accuracies we are getting from these partitions is very low as compared to iid fashion but in general when using federated learning we will get this kind of dataset only which is into non iid fashion, so what are the different ways to improve the accuracies?

(1) Advanced Aggregation Techniques

- **FedProx** → Adds a proximal term to local loss to reduce divergence.
- **FedNova**, **FedAdam**, **SCAFFOLD** → Improve convergence in non-IID settings.

(2) Increase Communication Rounds

- Needs more communication rounds for the global model to stabilize.

(3) Data Augmentation / Client-level Regularization

(4) Clustering-based FL (Group Similar Clients)

(5) Transfer Learning etc.,

❖ FL offers some level of privacy by design but not complete privacy. It provides:

1. Data remains on client device.
2. Only model updates are sent to server and raw data is with client only.

But, Federated Learning ≠ Fully Private, because in non iid settings, model updates can leak private information. To resolve all this, differential privacy, secure aggregation etc. are used.

➔ ***Differential Privacy (DP) in Federated Learning***

Federated Learning keeps the raw data on-device, it still doesn't guarantee that someone can't infer private information from the model updates. That's where Differential Privacy is used as powerful additional layer of protection.

➔ **Why use Differential Privacy (DP) in FL?**

DP adds random noise to the model updates/gradients so that no-one can trace them.

Explanation:

1. Client trains locally on private data.
2. Adds **differential privacy noise** (Gaussian or Laplace) to the gradients/weights.
3. Sends **noisy updates** to the central server.
4. Server aggregates updates from many clients — privacy is preserved!

- **More Privacy (Stronger DP) → More Noise → Slightly lower accuracy**
- **Less Noise → Higher accuracy but weak priacy**

Differential Privacy

https://flower.ai/docs/framework/explanation-differential-privacy.html

(Traditional DP) :
A randomized mechanism M provides (ϵ,δ)-differential privacy if for any two neighboring databases, D1 and D2

$$P[M(D1 \in A)] \leq e^{\epsilon} P[M(D2 \in A)] + \delta$$

The ϵ parameter, also known as the privacy budget, is a metric of privacy loss. Lower ϵ values indicate higher levels of privacy. The amount of noise needed to achieve differential privacy is proportional to the sensitivity of the output.

- **Central Differential Privacy (User-level DP):**
  - DP is applied by the server and the goal is to prevent the aggregated model from leaking information about each client's data.
- **Local Differential Privacy**:
  - DP is applied on the client side before sending any information to the server and the goal is to prevent the updates that are sent to the server from leaking any information about the client's data.

Central Differential Privacy (User-level DP):

➔ The approach is to clip the model updates sent by the clients and add some amount of noise to the aggregated model. In each iteration, a random set of clients is chosen with a specific probability for training. Each client performs local training on its own data. The update of each client is then clipped by some value S (sensitivity S).
➔ Afterwards, the Gaussian mechanism is used to add noise in order to distort the sum of all clients' updates.

The Gaussian mechanism is used with a noise sampled from $N(0,\sigma^2)$ where;
σ=(noise_scale*S)/(number of sampled clients).

Clipping

There are two forms of clipping commonly used in Central DP

- Fixed Clipping : A predefined fix threshold is set for the magnitude of clients' updates. Any update exceeding this threshold is clipped back to the threshold value.
- Adaptive Clipping : The clipping value is tuned during the rounds with respect to the quantile of the update norm distribution.

Local Differential Privacy:

Local DP leads to a decrease in accuracy but better privacy in comparison to central DP.

- Each client adds noise to the local updates before sending them to the server. To achieve (ϵ,δ)-DP, considering the sensitivity of the local model to be Δ, Gaussian noise; $\sigma=\Delta\times2\times\log(1.25\delta)\epsilon$

Federated Learning with Gaussian Differential Privacy

Differential privacy is used in privacy protection but differential privacy will reduce the accuracy of the machine learning model. Federated learning algorithm based on Gaussian differential privacy, **Noisy-FL** can more accurately track the changes in privacy loss during model training.

➢ Noisy-FL can achieve user-level privacy protection while increasing the number of communication rounds compared to the previous algorithm. Experimental results show that the new algorithm can increase the number of communication rounds by 3 times and when the total number of clients is small, the model accuracy of the Noisy-FL algorithm is increased by 10%.

- **Differential Privacy (DP)** is widely used due to its **low computational cost** and **strong mathematical guarantees**, but traditional DP adds noise that may hurt model accuracy, especially when the number of clients is small.
- Uses **gradient clipping + Gaussian noise** to hide individual client contributions more effectively.

Noisy-FL Algorithm

Two Steps in Model Aggregation:

1. **Gradient Clipping**:

   Gradient clipping can set the gradient within the effective limit, which is a necessary condition for obtaining a limited sensitivity. Therefore, setting an appropriate threshold for cropping is very important for the accuracy of the final intelligent model of federated learning. If the threshold is too low, it will cause information loss, while setting the threshold too high will increase a lot of noise.

2. **Adding Gaussian Noise**:

   Adding Gaussian noise to the cropped gradient is equivalent to continuously applying the Gaussian mechanism to each round of the communication global model.

- ➔ Differential Privacy
  https://colab.research.google.com/drive/1MiEWvcu8wzZu_Dy99tJQN0E_pp-G7Aqa?usp=sharing