Federated Learning

(BITS F464 Machine Learning - Project Report)

Group number- ML11

Group members:

- Jayakiran Reddy J - 2018AAPS0348H
- P Sree Vishnusai Karthik - 2018A3PS0526H
- Harsh Bohra - 2019A8PS0560H
- Nishant Kumar Rai - 2019A8PS0583H
- Harsha Vardhan Reddy - 2019A4PS0737H

Introduction:

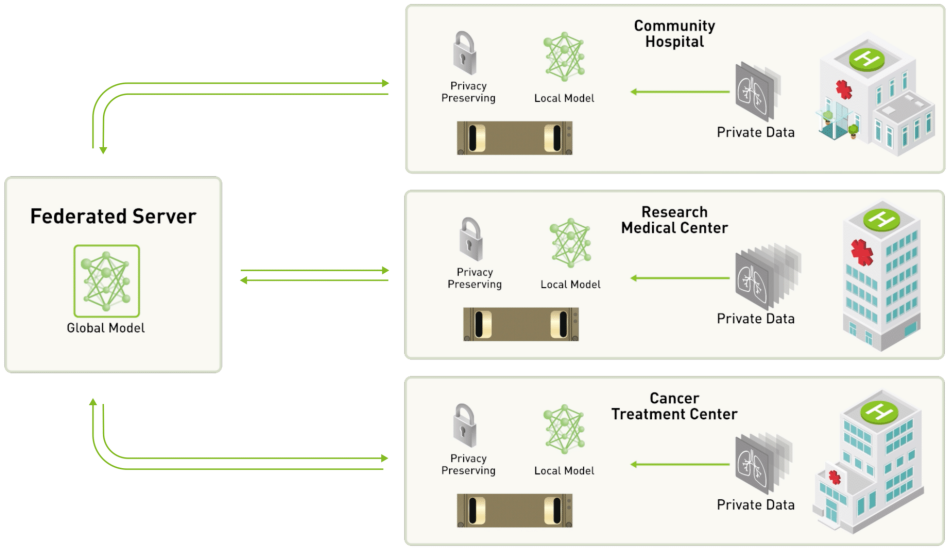
Quality data exists as islands on edge devices worldwide, including mobile phones and personal computers, and is secured by rigorous privacy rules. Federated Learning is an innovative way to connect machine learning models to these separate data sets, regardless of where they are stored, and, more crucially, without violating privacy rules. Federated Learning takes the ‘model to the data’ rather than the ‘data to the model’ for training, as is the standard practice. All that is required is for the device hosting the data to be willing to participate in the federation process.

Let us consider an example that we can observe in our own lives. Smartphones nowadays have predictive text models which allow us to autofill words from our keyboards using the suggestions provided by said models. These models are trained using our messages and other text outlets, obviously personal and private. This sensitive data must not be sent back to the server (Google Keyboard) from the edge devices (mobile phones, tablets, etc.). Furthermore, due to the enormous amount of data, having the server train the model is not practical due to the expensive data overhead and computation. Federated Learning provides us with a way to train models without data transfer, hence overcoming the privacy and computational challenges.

Federated Learning Layout. Image reference

(Source: https://towardsdatascience.com/federated-learning-a-step-by-step-implementation-in-tensorflow-aac568283399)

In its most basic version, the FL architecture comprises a server sitting in the center and coordinating training activities. Clients are primarily edge devices, which could number in the millions. Per training iteration, these devices communicate with the server at least twice. First, they all receive the weights of the current global model from the server, then train it on their own local data to generate new parameters, which are then uploaded back to the server for aggregation. This communication cycle continues until a pre-determined epoch number, or an accuracy criterion is met.



(Source: https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/)

Advantages of Federated Learning

- Data Security: data does not need to be pooled at a centralized server; instead, it is kept on the devices
- Data Diversity: companies may be unable to merge datasets from diverse sources due to challenges other than data security, such as network unavailability in edge devices. Federated learning makes it easier to access diverse data, even when data sources can only interact at particular periods.
- Continuous learning in real-time: Models are constantly upgraded utilizing client data, with no need to aggregate data
- Hardware Efficiency: Because federated learning models do not require a single complex central server to interpret data, this approach requires less complex hardware.

Disadvantages of Federated Learning

- Data Privacy:
    - In federated learning, data is not collected on a single entity/server; instead, it is collected and analysed by numerous devices. This can broaden the attack area.
    - Even though only models, not raw data, are sent to the central server, models could be reverse engineered to reveal client information. Differential privacy, secure multiparty computation, and homomorphic encryption are examples of privacy-enhancing technologies that can be utilised to improve the data privacy capabilities of federated learning.
- Performance Limitations:
    - Data Heterogeneity: In federated learning, models from several devices are combined to create a superior model. Device-specific factors may hinder the generalisation of models from some devices, lowering the accuracy of the model's next version.
    - Federated learning is a relatively new method of machine learning. To improve its performance, new studies and research are required.

Applications of Federated learning

Federated learning has various applications across several fields and industries. Some of them have been listed below.

- Detection of device failures in IIOT and IOT - A platform architecture of FL systems based on blockchain is designed, which supports the integrity of client data. In industrial working environment monitoring, it is very important yet difficult to follow the changing trend of the time series monitoring data when they come from different types of sensors and are collected by different companies. FL structure can not only keep the data privacy but also extract and fuse the trend features of time-series monitoring data of multi-sensors.
- Protecting highly sensitive information is really important and is the shared responsibility of hospitals and AI companies. To ensure this, Fedhealth (a federated transfer learning framework) was proposed, especially for wearable healthcare.
- FedCoin is a blockchain based payment system, It can mobilize free computing resources in the community to perform the expensive computing tasks required by the FL incentive plan.
- FedVision is an end-to-end ML engineering platform that supports the easy development of FL powered computer vision applications. The challenge of using image data owned by different organizations to establish an effective visual target detection model is solved with FL.
- The amount of labeled data collected in smart cities is small, and there is a lot of unlabeled data. A semi-supervised federated edge learning method, called FedSem, utilizes unlabeled data in smart cities. FedSem can use unlabeled data to improve learning performance, even if the ratio of labeled data is low.
- We can implement a federated energy demand learning method that allows charging stations to share their information without exposing the real dataset. The cluster based energy demand learning method is applied in charging stations to further improve the accuracy of energy demand prediction.
- Federated learning can be used to build models on user behavior from data pools of smart phones without leaking personal data, such as for next-word prediction, face detection, voice recognition, etc.
- Manufacturing companies can use federated learning models to develop predictive maintenance models for equipment.

Industries/Industrial applications that can benefit from FL:

FL could be applied to the entire product life cycle. FL also gives small data users (such as SMEs) an opportunity to make full use of intelligence.

- **Product recommendation systems**: In the non-FL setting, manufacturers can only make product recommendations that rely on their own sales. Companies should obtain more accurate recommendation services if they utilize FL mechanisms to train the recommendation model.
- **Industrial equipment health monitoring**: Modern industrial equipment is being connected to the Internet via IoT, and their health status can be monitored by big data intelligence. Industrial companies can apply FL mechanisms to harvest federated intelligence for monitoring equipment's health more accurately.
- **AR/VR-guided operations** : AR/VR has been widely used in industries, such as remote operation guidance, virtual assembly and machine operation training. Industrial companies can use FL strategies to train optimal models to improve the accuracy of detecting objects.
- **Product defect detection :** DL has a broad application prospect in the field of automatic detection. Enterprises that produce similar products can be attracted to join FL to realize sample expansion as one of the main challenges of applying DL is the lack of data samples for classification tasks.
- **Optimal supply chain scheduling :** Traditionally, the data on sales forecast across-regional distributors/industry associations is private. To realize efficient supply chain scheduling, manufacturers can encourage suppliers to participate in FL to extract the optimal model for predicting demand orders, supply quantity, inventory, and supply schedule.
- **Generative product design :** The design data from different companies are only available to themselves for privacy reasons. To shorten the design cycle and reduce design iterations, FL is expected for companies to optimize the generative product design process across enterprises based on the modeling of the human/machine/material resources in each enterprise.
- **Precise robotics collaboration :** Traditional RFID-based positioning accuracy is not high. RSSF positioning based on FL can achieve higher accuracy. This FL-enhanced precise positioning can be applied to robotics collaboration.

The Federated Averaging Algorithm

FedAvg is a generalization of FedSGD that allows local nodes to execute multiple batch updates on local data and exchanges updated weights rather than gradients. The reasoning behind this generalisation is that in FedSGD, averaging the gradients is strictly identical to averaging the weights themselves if all local nodes start from the same initialisation. Furthermore, averaging tuned weights from the same starting does not always degrade the performance of the averaged model.

1 - The coordinator executes:
2 - Initializes model weight w_0, and broadcasts the initial model weight w_0 to all participants.

3 - **for** each global model update round t = 1, 2, ... **do**
4 - The coordinator determines C_t, which is the set of randomly selected max (Kp, 1) participants.
5 - **for** each participant k ∈ C_t **in parallel do**
6 - Updates model weight locally: w_{t+1}^{k} ← *Participant Update* (k, $\overline{w}_t$)
7 - Sends the updated model weight w_{t+1}^{k} to the coordinator.
8 - **end for**
9 - The coordinator aggregates the received model weights, i.e. taking the weighted average of the received model weights: $\overline{w}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \times w_{t+1}^{k}$
10 - The coordinator checks whether the model weights converge. If yes, then the coordinator signals the participants to stop.
11 - The coordinator broadcasts the aggregated model weights $\overline{w}_{t+1}$ to all the participants.
12 - **end for**

13 - **Participant Update** (k,$\overline{w}_t$)
(this is executed by participant k, ∀k = 1,2,3…K
14 - Obtain the latest model weight from the server, i.e. set $w_{1,1}^{k} = \overline{w}_t$
15 - **for** each local epoch i from 1 to number of epochs S **do**
16 - batches ← randomly divides dataset D_k into batches of size M.
17 - Obtain the local model weight from the last epoch, i.e. $w_{1,i}^{k} = w_{B,i-1}^{k}$
18 - **for** batch index b from 1 to number of batches B = $\frac{n_k}{M}$ **do**
19 - Computes the batch gradient g_k^b
20 - Updates model weights locally: $w_{b+1,i}^{k} \leftarrow w_{b,i}^{k} - \eta g_k^b$
21 - **end for**
22 - **end for**
23 - Obtains the local model weight update $w_{t+1}^{k} = w_{B,S}^{k}$, and sends it to the coordinator.

(Source: Federated Learning; Qiang Yang, Yang Liu, Yong Cheng, Yan Keng, Tianjian Chen, Han Yu; Morgan and Claypool Publishers)

Improvement Made to Algorithm (as used in code):

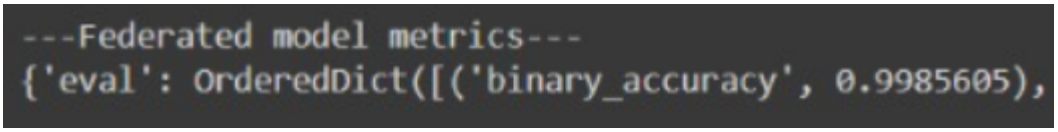
We see that in the data that we are presented with, the target attribute “Class” takes values 0 and 1. The distribution is overwhelmingly dominated by the negative class, i.e. value 0.

```
1 df['Class'].value_counts()

0    284315
1       492
Name: Class, dtype: int64
```

Due to this, our model may overfit the data with negative class since most of our records are of that. Hence there is a need for class balancing to tackle this. We will undersample the negative class in order to keep the ratio for both classes equal. From the resulting dataset, we train our model.

```
def make_tf_dataset(dataframe, negative_ratio=None, batch_size=None):
    dataset = dataframe.drop(['Time'], axis=1)

    # Class balancing
    pos_df = dataset[dataset['Class'] == 1]
    neg_df = dataset[dataset['Class'] == 0]
    # if negative_ratio:
    #     neg_df = neg_df.iloc[random.sample(range(0, len(neg_df)), len(pos_df)*negative_ratio), :]
    balanced_df = pd.concat([pos_df, neg_df], ignore_index=True, sort=False)

    y = balanced_df.pop('Class')

    # Dataset creation
    dataset = tf.data.Dataset.from_tensor_slices((balanced_df.values, y.to_frame().values))
    dataset = dataset.shuffle(2048, seed=SEED)
    if batch_size:
        dataset = dataset.batch(batch_size)

    return dataset
```

As a result, we notice increased accuracy and a significant decrease in runtime (by a little over 30 minutes). More importantly, our model will not be overfitting a certain class, hence improving the quality of the algorithm.

- 

(Without class balancing)

- 

(With class balancing)

Work distribution:

Group member	Task
Nishant Kumar Rai	Code + Improvement, Report (Improvement Made to Algorithm)
Harsh Bohra	Code, Report (Introduction, Advantages, Disadvantages, Federated Avg Algorithm)
Jayakiran Reddy J	Research on Federated learning, Report(Applications of Federated Learning)
P Sree Vishnusai Karthik	Report (Applications of FL, Industrial Scope)
Harsha Vardhan Reddy C	Research on FL (3,4) + report(introduction)

References:

1.https://ieeexplore.ieee.org/document/9475501

2.https://ieeexplore.ieee.org/document/9233457

3.https://ieeexplore.ieee.org/document/9187874

4.https://towardsdatascience.com/federated-learning

5.https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8950073

Link to the Kaggle dataset:

https://www.kaggle.com/mlg-ulb/creditcardfraud

Link to the github repository:

https://github.com/Felix080501/Machine_Learning_Project_ML11.git