My Project:
Centralized Machine Learning by Differential Privacy using FATE-AI

Course Name: Mobile Internet

Student Number: M202161029

Student Name: Alec Mabhiza Chirawu

Chinese Name: 亚历克上

Department: Information and Engineering

Mobile Internet Course, Information And Engineering Department, USTB

Professor:Dr. Huang(Dr. Roland)

07 January 2022

Table of Content

Key Points

To run this project I recommend the use of pycharm and the installation of these specific modules versions else running into error is prone :

i. Python --version python3.8
ii. Tensorflow version 2.6.1
iii. Tensorflow _federated version 0.19.0
iv. Numpy version 1.22.0
v. Nest_asyncio version current
vi. Tensorflow-federated-nighty version current
vii. Tb-nightly version current

This module might brought some problems its safe to uninstall it :
i. Uninstall --yes tensorboard tb-nightly

1. Introduction

Multi Party Computation (MPC) is gaining traction as a technique to conduct operations in an untrustworthy ecosystem without revealing data. MPC would safeguard models parameters in machine learning while allowing numerous worker nodes to participate within training process with their very own datasets, the process is known as Federated Learning (FL). Securely trained models, on the other hand, have been demonstrated to be vulnerable to reverse-engineering attacks, which can extract critical datasets information straight from the model. Deferentially Private approaches are a collection of ways that handle this and can effectively safeguard data.

Another solution is to not release the data. Instead, it is kept centralized by a trusted party which answers queries. But how do we ensure that those queries do not leak private information? One idea might be to allow only simple queries such as counting. Moreover, answers could only be returned when there is a minimal query set size.

2. Project Target

The following are my contributions:

- First, I developed a standardized protocol for communicating across devices, which enabled federated learning.

- After that, we create a tensor chain abstraction model to efficiently encode new activities like sharing a tensor across devices and the main center.

- Finally, utilizing federated tensor flow framework present the pieces to construct newly suggested differential privacy and multiparty computing protocols.

Centralized machine learning application will have a local copy on all devices, where users can use them according to the need. The model will now gradually learn and train itself on the information inputted by the user and become smarter, time to time. The devices are then allowed to transfer the training results, from the local copy of the machine learning app, back to the central server.This same process happens across several devices, that have a local copy of the application. The results will be aggregated together in the centralized server, this time without user data this way the differential privacy will be upheld without leaking the data itself using this FATE-AI.

3. Applying differential Privacy

I introduced differential privacy as a training strategy for deep neural networks with a low privacy budget making use of tensorflow. To do this, the project presents a new estimate of the privacy loss that can be used to fine-tune the amount of noise required, as well as a novel technique for boosting the efficiency of private training.

This method ensure differential privacy by training the final model with noisy and aggregated votes from pre-trained and unpublished models.

4. Federated Learning

This system introduces the concept of of data centralization to make data ready to be trained by every device of activities easier. Different small puzzled data from different devices will be encrypted and transferred to the same server through internet. The server merely duplicate the chain of commands and offer the same interface for communication as the devices with encrypted data.

As of present, there are two implementations of network workers in the Federated Learning framework. One is based on standard network sockets, while the other is based on Web Sockets. When developing federated learning systems, this provides an additional degree of granularity before data transferring on the same workstation. Web Socket workers are also well-suited to the data science ecosystem centered on browser-based servers.

5. Code Structure

Transforming data or passing tensors to other workers may be described as a series of operations, each of which is expressed by a distinct class. To do this I have made use of tensorflow, tensorflow_federated, numpy, collections and nest_asyncio. These are data representations that may be strung together to express a state or change. The tensorflow_federated is always at the top of the structure, and the collections manipulations or phases are accessible downward by using child attribute and upward using the parent property.

```
import nest_asyncio
nest_asyncio.apply()
import collections
import numpy as np
import tensorflow as tf
import tensorflow_federated as tff
```

Nest_asyncio helps nested loops.

Collections are containers for storing data collections such as lists, dictionaries, sets, and tuples. These are collections that have already been installed. A variety of modules have been developed to provide additional data formats for data collecting storage. The Python collections module was intended to extend the capability of the built-in collection containers. This function collects the encrypted datasets to the model:

```
def preprocess(dataset):
    def batch_format_fn(element):
        return collections.OrderedDict(x=tf.reshape(element['p
        return dataset.repeat(NUM_EPOCHS).shuffle(SHUFFLE_BUFF
```

To get the deep smart models I used Keras from tensorflow, it saves well since its also distributing training of this learning model.

```
def create_keras_model():
  return tf.keras.models.Sequential([
      tf.keras.layers.InputLayer(input_shape=(784,)),
      tf.keras.layers.Dense(10, kernel_initializer='zeros'),
      tf.keras.layers.Softmax(),
  ])
```

Tensorflow Federated is a decentralized data machine learning and computations open-source platform. It encourage open research and experimentation with Federated Learning, a machine learning approach that involves training a single global model across several clients that save their training data locally. FL has been used to train prediction algorithms for mobile keyboards without providing sensitive typing data to servers.
In this project tensorflow federated makes use of this functions where it arrange the data from the clients(clients refers to the device) for processing.

```
def make_federated_data(client_data, client_ids):
  return [
      preprocess(client_data.create_tf_dataset_for_client(x))
      for x in client_ids
  ]
```

This function a hybrid of many perceptron models. Perceptron are inspired by the human brain and attempt to address issues by simulating its functioning. These perceptron are extensively linked and parallel data evaluation happens in this function:

```
for round_num in range(2, NUM_ROUNDS):
  state, metrics = iterative_process.next(state, federated_train_data)
  print('round {:2d}, metrics={}'.format(round_num, metrics))
  evaluation = tff.learning.build_federated_evaluation(MnistModel)
  train_metrics = evaluation(state.model, federated_train_data)
  federated_test_data = make_federated_data(emnist_test, sample_clients)
  test_metrics = evaluation(state.model, federated_test_data)
```

Outputs commands:

Firstly the federated train data checks the number of sample clients and print them out and the it also print all the datasets applied to it , these are the commands in charge of these 2 print outs:

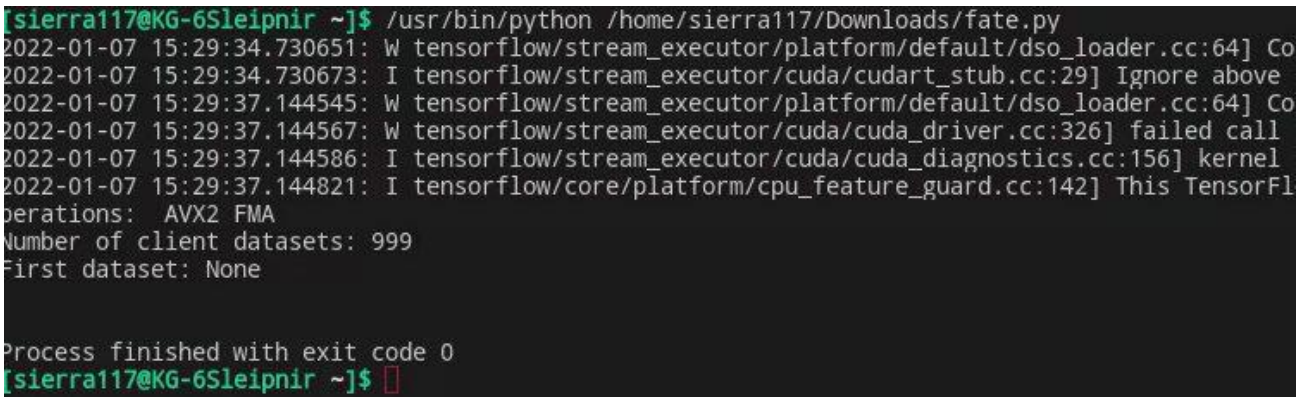
```
sample_clients = emnist_train.client_ids[0:NUM_CLIENTS]
federated_train_data = make_federated_data(emnist_train, sample_clients)
print('Number of client datasets: {l}'.format(l=len(federated_train_data)))
print('First dataset: {d}'.format(d=federated_train_data[0]))
```

All the metrics data will be displayed in the form of encrypted functions by this command:

```
state, metrics = iterative_process.next(state, federated_train_data)
print('round  1, metrics={}'.format(metrics))
```

Results

The FATE managed to access all the encrypted datasets outputs in less time and the operation was done by AVX2 and FMA which are tensorflow reliance's. This proves that the task was successful with 999 datasets output and threat was none.



Conclusion

Fate with Differential privacy in machine learning is based on a simple premise to run the encrypted data then make decision and that decision does not reveal information about any single training. In this project we have seen good results , but there is a huge room for improvement and upgrading of the system since these system wasn't fully tested due to limited time.

References

[1] P.C. Mahawaga Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe and M. Atiquzzaman, "Local Differential Privacy for Deep Learning," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 5827-5842, July 2020, doi: 10.1109/JIOT.2019.2952146.

[2] J. Zhao, Y. Chen and W. Zhang, "Differential Privacy Preservation in Deep Learning: Challenges, Opportunities and Solutions," in IEEE Access, vol. 7, pp. 48901-48911, 2019, doi: 10.1109/ACCESS.2019.2909559

[3] T. Ha, T. K. Dang, T. T. Dang, T. A. Truong and M. T. Nguyen, "Differential Privacy in Deep Learning: An Overview," 2019 International Conference on Advanced Computing and Applications (ACOMP), 2019, pp. 97-102, doi: 10.1109/ACOMP.2019.00022.

[4] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, May 2020, doi: 10.1109/MSP.2020.2975749.