Privacy Preserving Federated Machine Learning

Kanodia Ritwik (ritwik002@e.ntu.edu.sg)
School of Computer Science and Engineering,
Nanyang Technological University

Dr. Sourav Sen Gupta
School of Computer Science and Engineering,

Assoc. Prof. Wang Huaxion
School of Physical & Mathematical Sciences

Abstract - This project emphasizes on preserving the privacy of the training data used to train deep. Neural network algorithms. We trained a global bank model which is used to predict if a credit card transaction is fraudulent. We have a federated learning framework with three banks B1, B2 and B3. An aggregator is defined which takes weights of the three banks and the global bank, performs a weighted average and assigns this weight to the global bank. For the sake of simplicity, we trained each of the three banks with a single data point. The aggregator has access to the weights and gradients of the three banks and therefore is taken to be the attacker, trying to compromise the privacy of the three data points each of which were used for training of the three banks. In our research we found out that by implementing the underlying mathematics in neural networks, the three training points can be calculated with the weights and gradients of the three banks known. However, if we add some random gaussian noise to the gradients of the three banks, this no longer holds. maintained. This is what the TensorFlow Privacy Preserving Library does with the added benefit of trimming the gradients to reduce the effect of outliers, increasing generalization and not affecting the performance of the model. This can be extended to the entire data set instead of one point and increase the utility of neural networks to industries working with sensitive user data.

Keywords – deep learning, privacy, TensorFlow, federated learning.

1 INTRODUCTION

Machine learning is becoming popular in a data driven world like ours. It is the basis behind many widely used tools such as speech and image recognition, natural language processing etc. which are very widely accepted and used in our day to day lives. New application for machine learning are found frequently and it would be right to say that it has great potential. Deep neural networks, is a particular type of machine learning algorithm which is becoming very popular and widely used in modern day given its utility in driverless cars, image recognition, Natural language processing, etc. Machine learning algorithms in general and Deep neural networks in particular need huge chunks of data to be trained. This data is referred to as training data. The characteristics of this data is significant in deciding the characteristics of the model. One issue often faced is that a Machine learning algorithm does not generalises well over a problem it is dealing with. For example, there may be a speech to text algorithm which is accurate at converting a speech in a particular accent A to text, but not so accurate for an accent B. There is a high chance that this may be attributed to the fact that the training data set of the algorithm had most of the data points generated in accent A. Another possible issue may be that the training data set had limited number of data points for the algorithm to learn from. The traditional way of training machine learning algorithms is to train them on a single server, with the local data points of the server. This is a limitation which makes it difficult to address the two issues mentioned above. One way to solve this can be to change the architecture of training the models to a multi-server process. This is what a federated learning system is. However, in such a scenario, another concern which arises is the privacy of the data. The local data points of servers cannot be shared with each other. In this research project, we identified how such a system functions, whether or not if it is effective and if the privacy of the local data points in each of the servers in the multi-server system is persevered. In this report we assume that the reader is familiar with basic machine learning theories and how a deep neural network model functions at a low level.

2 GENERAL LAYOUT

The first half of the paper will deal with the federated learning system, discussing the general concepts and working followed by the description of the federated learning architecture set up by us for this research project and its effectiveness. The second half of the paper will discuss at a theoretical level the types of Membership inference attacks that are possible in such a design and how can they be prevented, with particular focus on the TensorFlow Privacy Library by Google.

3 FEDERATED LEARNING SYSTEM

Before jumping to the implementation, it is important we define a generalised federated learning system. As discussed before, a machine learning model generalises well over a problem when the data that was used to train it is diverse and has enough number of data points. However, data has privacy restrictions and cannot be shared between servers. For example, an organisation cannot share its data with another organisation because it would go against the privacy protection standards that exist between them and their clients. Federated learning system supposedly provides a way out of this. We should understand how.

3.1 ARCHITECTURE

Federated Learning system follows a multiple servers type of architecture. In this type of architecture, there are normal servers and a centralised server. Each server has its own properties in the form of its local data points and a local machine learning model which was trained using the local data samples. The model of the centralised server is referred to as the ‘Global model’ which is the final product after sufficient iterations of the federated learning process. No direct data sharing takes place between any of the servers.

3.2 WORKING

Due to ever changing nature of the local data sets of the servers, federated learning is an iterative process. We divide a single iteration into two steps for clarity.

In the first step, in an ideal world, all the servers would have shared their data with the centralised server and then the centralised server would have compiled and trained a single global model on this data set. However, in a non-ideal world the federated learning system provides the alternative of the servers sharing their locally trained models instead, with the centralised server, which then would update its global model based on an aggregate of the parameters of all of these models, completing one iteration of the Federated learning system.

In the second step of an iteration the parameters of the global model is shared with each of the local servers. They set the parameters of their local models to this parameter. Any further training with new data points takes place on this updated local model.

Mapping to a real world scenario, the local servers maybe hosted by organisations from the same sector. As new data points come in and the local models change, , after a significant change, they have to share their local models again with the centralised server leading to the second iteration with same two steps as described above.

This process can be easily visualised with the following figure.

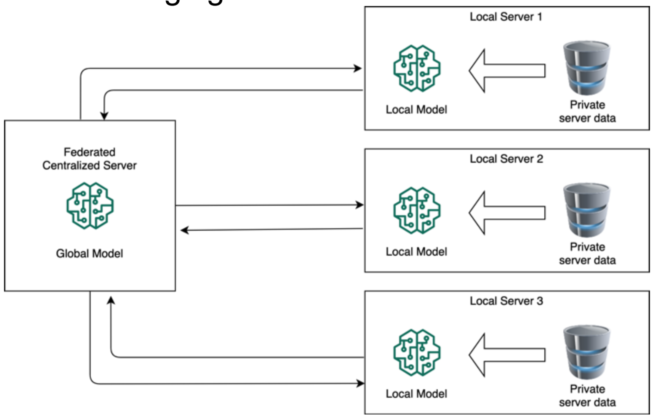

Fig 1: Illustration of the federated learning system

4 FEDERATED LEARNING SYSTEM - IMPLEMENTATION

We will now discuss our implementation of the federated learning system and its effectiveness. Note that out of the two steps in one iteration of a federated learning system, we only focussed on the first step. The scope of this report and the following implementation steps do not cover the second step where the parameters of the global model is shared with the local models in order to simplify the process, implementation and explanation.

4.1 SET-UP

For the purpose of this report, we have taken the example of a banking system. The setup is such that there are three banks, namely Bank 1, Bank 2, Bank 3, each acting as a server and a Global bank acting as the centralized server with the Global model. This setup can be defined as a scenario where three banks have collaborated and agreed to share their locally trained models with the centralized server to train a global model which is much more efficient at detecting credit card fraud across transactions, even the ones going through different banks. We have used the dataset 'creditcard.csv' from Kaggle and performed two iterations of the federated learning process.

4.1.1 Dataset

The data set 'creditcard.csv' contains credit card transactions information. It has 31 columns in total out of which, data in 28 columns namely V1 up to V28 are PCA (Principal Component Analysis) transformed as they contain sensitive information with respect to original transactions. The other three columns are Time, Amount and Class. The time for the first transaction is assumed to be zero and that for the following transactions is relative to this. Amount refers to the amount of money involved in the transaction. The column class holds binary values with the value '1' classifying a transaction as fraudulent and '0' as not.

4.1.2 Data-distribution

As mentioned before, each server has its own data samples. Therefore, the data set is divided into parts and each server is assigned some data to act as its local data. Furthermore, for each server, its local data is divided into two parts, 'A' and 'B', such that part A is used to train the local model of the server in the first iteration and part B is used to train the local model of the server in the second iteration of the federated learning process. Note that under normal circumstances, in the second iteration, a local server's model would have been trained by all data combined (A and B) but we eliminated A to produce significant differences , given the small size of the data set each server gets. The data distribution should be clear with the tree diagram given below. The tree follows a MECE structure (Mutually exclusive and Cumulatively Exhaustive) making sure that there is no data being shared between the servers and all of the data is put to use.

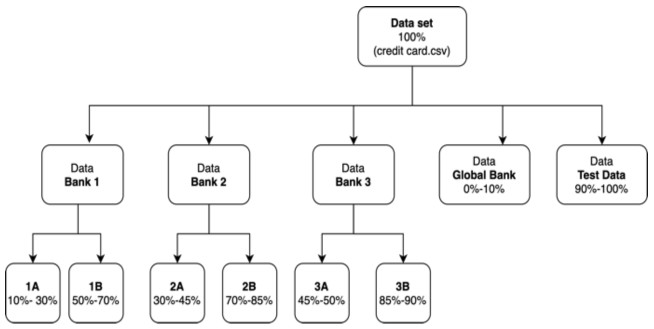

Fig. 2: **Data distribution tree for servers**

In the figure above, the representation 10% - 30% refers to the 20% portion of the data that is after the first 10% of the data set up to 30% of the data set. Other figures can also be read accordingly. The Global bank data refers to the portion of the data used to train an initial global model at the centralized server. Test data is used to evaluate the efficiency of the global model after two iteration of the federated learning framework.

4.2 TRAINING

4.2.1 Global model

We start with training an initial Global bank model at the central server with the data samples which was assigned to it. (10% of the data set)

4.2.2 Individual Banks

Each of the individual banks, Bank 1, Bank 2 & Bank 3 are trained with the data samples which was assigned to their servers respectively.

4.2.3 Aggregation

The aggregation step of an iteration constitutes of two steps. The first step is to aggregate the weights of the models of the three banks, namely, Bank 1, Bank 2, Bank 3. The weights are aggregated by taking a weighted average of the weights of the models. The weight assigned to each model’s weight is based on the amount of data that was used to train the model. More the data that was used for training, higher the weight. Based on the data distribution illustrated above, Bank 1 has the highest percentage of data (20%), followed by Bank 2 (15%) followed by Bank 3 (5%) in both iterations. Accordingly the weights assigned to the parameters of each model is,

Bank 1: 0.55 (w1)

Bank 2: 0.35 (w2)

Bank 3: 0.1 (w3)

The equation for calculating the aggregate can be given by,

```
Aggregate = (Weights of bank 1
model)*(w1) +  (Weights of Bank
2 model)*(w2) + (Weights of
Bank 3 model)*(w3)
```

The second step comprises of updating the Global Bank model based on the aggregate calculated. The aggregate acts as a gradient for the Global model and is simply added to its initial weight.

A second iteration of the federated learning system is carried out in a similar way with the three bank models being trained with their respective second datasets, i.e. part B.

We get the final model which we proceed to test on the test data which is not part of the datasets of any of the servers.

4.3 TESTING

For the testing the efficiency of a federated learning framework, we trained the same machine learning algorithm assuming only one server but with the entire dataset. Given that this server had access to much more data points, as the data set was not getting distributed, the accuracy was higher. However, this will not be the case in a real-world scenario where data sharing between servers is not allowed. Also, in a federated learning framework, the model at the centralised server is exposed to a variety data points indirectly, coming from a number of different organisations. This makes the model much more generalised and will prove to be more efficient for solving the overall problem in a general sense instead of for a given organisation.

5 PRIVACY PRESERVING FEDERATED LEARNING

It is important to understand that the entire basis for a system like federated learning, is to protect the privacy of local data in servers. However, in a federated learning framework, one of the servers may come under Membership inference attacks, aiming to compromise the data of a particular individual or a particular data element in the local data samples of that server.

5.1 MEMBERSHIP INFERENCE ATTACKS

The servers share their models with the aggregator in the centralized server which aggregates the weights of the model. Given that the aggregator has access to the models, it may be used to launch an attack on any of the servers with the aim of compromising the server's local data points. In this context, a data point can be said to be compromised if any inference with respect to its contents or its presence itself in the server can be made. Therefore, mainly two types of membership attack on a server are possible. An attack may be capable of finding the contents of a data point, or it may be able to infer a particular data element is present, on a server's local data set by mapping an outsized gradient to an outsized point in the dataset. In this paper, the scope will be limited to the first type of attack.

5.2 ATTACK (BACKTRACKING)

When we say that all the servers share their models with the centralized server, it means the centralized server has access to the gradients of each of the servers. Given that they have access to this gradient, we found out the privacy of the local data set of a server can be compromised. Let us see how,

Let the server being attacked be Bank 1. For simplicity of explanation, let us assume that the local data set of bank 1 has only one data point and the machine learning algorithm being used is logistic regression, i.e., there are no hidden layers, just one input and one output layers.

Terminology:

dw[1] -> Gradient for the weight parameter of the output layer of the network. dw[0] is not defined.

db[1] -> Gradient for the bias parameter of the output layer of the network. db[0] is not defined.

dz[1] -> Differentiation of the Loss function with respect to z[1] where z[1] can be defined as

z[1] = w[1][T] . a[0] + b[1] ;

w[1]->Weight parameter for the output layer.

b[1] -> Bias parameter for the output layer

a[0] -> The data in the input layer. In this case a single data point.

[T] -> Refers to the transpose of a matrix.

The relations used are the following,

Note: The explanations and derivations of the following equations are out of the scope of the report.

dw[1] = dz[1] **.** a[0]T ....[1]

dz[1] = db[1] ....[2]

a[0]T = dw[1] / dz[1]...[3, From 1]

a[0]T = dw[1]/db[1] .....[From 2 & 3]

Therefore the final equation we get is,

a[0]T = dw[1] / db[1]

a[0]T = dw[1] * db[1]$^{-1}$

As mentioned before, the centralized server has access to the model of the server and thus the gradients. Therefore the centralized server has the dw[1] and db[1] matrices. If the inverse of db[1] matric exists, the original data point of server 1, represented by a[0] can be calculated, compromising the privacy of the server.

5.3 PREVENTION

TensorFlow Privacy Library, a library provided by google can prove to be useful in such a scenario to prevent such attacks and maintain the privacy of the local data samples of a server

6 TENSORFLOW PRIVACY

It is a library provided by Google in the TensorFlow environment which can be used to protect the privacy of data in federated learning framework like the one described above. The attack involves mathematics (for backtracking) and the reason this library will prove to be effective is because it provides strong mathematical guarantees of privacy.

6.1 METHODOLOGY

The library works mainly by altering the optimisation process during the training of a machine learning algorithm. The working can be broken down to five simple steps. For simplicity, lets assume the optimisation algorithm being used is SGD (Stochastic Gradient Descent). Applying a gradient descent algorithm on the entire dataset is a tedious task therefore to speed up the process, a batch of data is taken from the data set at random. This is what differentiates SGD from normal Gradient Descent. Let’s jump into the steps assuming we have a batch of data,

Step 1: The batch of data is further split into a number of mini batches.

Step 2: The library then calculates gradient over each of these mini batches.

Step 3: If a particular mini batch has data points which is an outlier, the gradient will be high thus making it easy for an attacker to map the gradient to that mini batch. Therefore, the gradients of all the mini batches are clipped.

Step 4: The library then averages the clipped gradients across the mini batches to get the overall gradient for the batch of data.

Step 5: The process ends with adding some random gaussian noise on top of this averaged gradient.

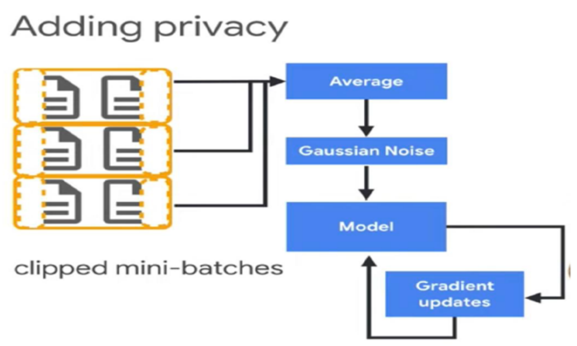

Fig 3: Illustration of TensorFlow Privacy library implementation.

The use of this library introduces three new hyper parameters, number of micro batches, extent of gradient clipping and amount of noise multiplied. These can be tuned to increase the privacy guarantees and performance. It is important to note that this library will work in a similar fashion with any other optimisation algorithm including normal gradient descent. For normal gradient descent, the entire data set will be treated as a batch, followed by the same steps from one through five.

6.2 WORKING WITH A FEDERATED LEARNING SYSTEM

The TensorFlow privacy can prove to be particularly effective in a Federated learning system as the servers can use the library to train their local models before sharing it with the aggregator or the centralised server. The use of the library will ensure that the shared model cannot be used to leak any kind of information with respect to the local data samples of the server. The process can be visualised with the help of the following figure.

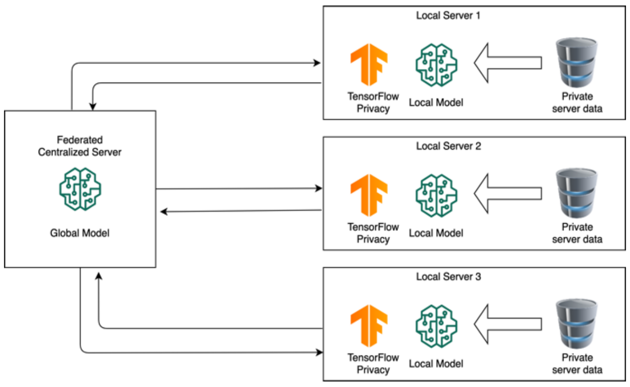

Fig 4: Illustration of TensorFlow Privacy Library in a federated learning system.

6.3 EFFECT ON PERFORMANCE

The effect of using TensorFlow Privacy Library while training a deep learning model is minimum. This is mainly credited to the fact that the library only minimizes the effects of outliers in the training process which have the highest risk of being exposed. The random gaussian noise added is also a minimum. There is only a small perplexity loss and the performance of the model is hardly reduced for predicting the labels for data points which are not outliers or have an outsize effect. To summarize, if we train a model with some data and another model with the some data along with an outlier and the TensorFlow privacy library, there will not be any difference between the two models.

7 CONCLUSION

To conclude the report, the federated learning system is an effective method of training a machine learning model which is more generalised and has a widespread use than one which is trained using data points coming from a single source. The traditional method makes the model more personalised for the use of that organisation which is providing the data than the general use case. Although, we did observe a minor loss in performance for the federated learning system when compared to the traditional method, this can be attributed to the fact that the testing data used is also from the same data set and is not diversified. However, some privacy concerns do arise in such a method in the case that if an attack is launched by the aggregator, the privacy of the data stored locally by the servers may be compromised. This concern can be addressed by using the TensorFlow Privacy library while training the machine learning models at each of the servers. Using this library protects the server from any attack that maybe launched by the aggregator.

ACKNOWLEDGMENT

I would like to acknowledge my research project supervisors, Dr. Sourav Sen Gupta and Assoc. Prof. Wang Huaxion who provided their constant support throughout the course of this project, in all aspects.

I would like to thank Numair Fazili, my fellow research partner who helped and guided me through various concepts.

I would like to acknowledge Andrew Ng whose online courses helped me become familiar with machine learning concepts.

I would like to thank my family and friends for their constant encouragement.

I would like to acknowledge the funding support from Nanyang Technological University – URECA Undergraduate Research Programme for this research project.

REFERENCES

[1] Rieke, N., 2020. *What Is Federated Learning?* | *NVIDIA Blog*. [online] The Official NVIDIA Blog. Available at:

<https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/>

[2] TensorFlow. 2019. *Federated Learning | Tensorflow Federated*. [online] Available at: <https://www.tensorflow.org/federated/federated_learning>

[3] Radebaugh, C. and Erlingsson, U., 2019. *Introducing Tensorflow Privacy: Learning With Differential Privacy For Training Data*. [online] Blog.tensorflow.org. Available at: <https://blog.tensorflow.org/2019/03/introducing-tensorflow-privacy-learning.html>

[4] Shokri, R., Stronati, M., Song, C. and Shmatikov, V., 2020. *Membership Inference Attacks Against Machine Learning Models*. [online] Cs.cornell.edu. Available at: <https://www.cs.cornell.edu/~shmat/shmat_oak17.pdf>

[5] Papernot, N., 2020. *Machine Learning With Differential Privacy In Tensorflow*. [online] cleverhans-blog. Available at: <http://www.cleverhans.io/privacy/2019/03/26/machine-learning-with-differential-privacy-in-tensorflow.html>