Communication Efficient Aggregation for Privacy Preserving Machine Learning

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Anish Reddy Ellore
ID No. 2016A7TS0104H

Under the supervision of:

Dr. Chittaranjan Hota
&
Dr. Paresh Saxena

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, HYDERABAD CAMPUS
June 30, 2020

Declaration of Authorship

I, Anish Reddy Ellore, declare that this Undergraduate Thesis titled, 'Communication Efficient Aggregation for Privacy Preserving Machine Learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Certificate

This is to certify that the thesis entitled, "*Communication Efficient Aggregation for Privacy Preserving Machine Learning*" and submitted by Anish Reddy Ellore ID No. 2016A7TS0104H in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by him under my supervision.

Supervisor
Dr. Chittaranjan Hota
Professor,
BITS-Pilani Hyderabad Campus
Date:

Co-Supervisor
Dr. Paresh Saxena
Asst. Professor,
BITS-Pilani Hyderabad Campus
Date:

Acknowledgements

I would like to thank Anirudh Kasturi for his help and guidance in completion of my thesis.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, HYDERABAD CAMPUS

Abstract

Bachelor of Engineering (Hons.)

Communication Efficient Aggregation for Privacy Preserving Machine Learning

by Anish Reddy Ellore

One of the core principles of machine learning is to collect data and use it for learning, but the collection of data leads to privacy concerns and also network communication issues. Federated learning is one approach of distributed learning handling privacy concerns and network communication issues. In this study we propose Fusion Learning which focuses on reducing network communication overhead in distributed learning paradigm while preserving data privacy. The researcher also discusses about data aggregation issues and applicability of these distributed learning approaches on different types of datasets.

Contents

Declaration of Authorship i

Certificate ii

Acknowledgements iii

Abstract iv

Contents v

List of Figures vii

List of Tables viii

Abbreviations ix

1 Federated Learning 1
- 1.1 Introduction . . . 1
- 1.2 Data . . . 1
  - 1.2.1 IID data . . . 2
  - 1.2.2 Non-IID data . . . 2
- 1.3 Algorithm . . . 2
- 1.4 Conclusion . . . 3

2 Fusion Learning 4
- 2.1 Introduction . . . 4
- 2.2 Algorithm . . . 4
  - 2.2.1 Client's message . . . 5
  - 2.2.2 Server computation . . . 6
- 2.3 Experimetal Results . . . 7
  - 2.3.1 Feature Distributions . . . 7
  - 2.3.2 Local and Global Models . . . 7
  - 2.3.3 Training and Testing Accuracies . . . 8
  - 2.3.4 Communication Efficiency . . . 10

2.4 Issues 11

3 Hybrid Fusion Learning **12**
3.1 Introduction 12
3.2 Architecture 12
3.2.1 Client layer 12
3.2.2 Edge layer 13
3.2.3 Cloud layer 14
3.3 Experimental Results 14
3.3.1 Setup 15
3.3.2 Performance Metrics 16
3.3.3 Results 16

4 Privacy Preserving Machine Learning with GAN **18**
4.1 Introduction 18
4.2 General Adverserial Networks 18
4.2.1 GAN model construction 19
4.3 Algorithm 19
4.4 Experimental Results 20
4.5 Conclusion 22

Bibliography **23**

List of Figures

2.1 Fusion Learning architecture 5
2.2 Distribution of each feature for Credit Card, Breast Cancer, Gender Voice and Audit datasets 8
2.3 Comparison of Training accuracy between Centralized Learning, Federated Learning and Fusion Learning algorithms 9
2.4 Comparison of Testing accuracies of initial local model vs final global model at each client 10

3.1 Architectural diagram of a hybrid fusion learning system consisting of three layers, Client, Edge and Cloud. 13
3.2 Distributions of three features of Credit Card dataset. 16
3.3 Testing accuracy for hybrid fusion, fusion and federated learning for Credit Card, Bank Marketing and Adult datasets. 17

4.1 Pictorial representation of GAN model reaching convergence.Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping x = G(z) imposes the non-uniform distribution pg on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. D(x) = 0.5 .[5] 19
4.2 Model architecture 20
4.3 Generated images from client's generator 21

List of Tables

2.1 Different types of distributions used to verify the distribution of individual feature set . . . 5
2.2 Dataset description . . . 7
2.3 Comparison of training accuracies (in %) between Central Learning, Federated Learning and Fusion learning . . . 8
2.4 Network usage of Federated Learning and Fusion Learning for E epochs for a single client . . . 10
2.5 MNIST . . . 11

3.1 Dataset description . . . 15
3.2 Total time taken and accuracy with hybrid fusion, fusion, and federated learning.(*ime for Communication rounds not added) . . . 17

4.1 MNIST . . . 22

Abbreviations

GAN	**G**eneral **A**dversarial **N**etworks
FGAN	**F**usion **GAN**
PPML	**P**rivacy **P**reserving **M**achine **L**earning
ML	**M**achine **L**earning

Dedicate to my family and teachers

Chapter 1

Federated Learning

1.1 Introduction

Modern mobile devices these days have access to a plethora of data, which when used will help improve the user experience. For example, enhanced text suggestions, image recognition, and voice recognition. The rich data enabling all these tasks is often sensitive data, resulting in a privacy breach when sent to the data center for training. Federated learning [10] advocates a different approach that leaves sensitive data to mobile devices and learns a globally shared model based on the locally computed results. This federated learning approach is a distributed machine learning paradigm which will become helpful as the data size increases. In the coming sections description about algorithm, assumptions on data and network will be introduced.

1.2 Data

Federated learning can be used on any type of data such as image, text, time-series etc. This because federated learning itself does not change much from classical machine learning at core but it modifies some parts of it to make room for privacy, more about this will be discussed in the algorithm section. The working of classical distributed machine learning algorithms rely heavily on the assumption that data is IID (Independently and identically distributed) but since most of the private data is Non-IID we cannot use classical distributed machine learning algorithms for privacy preserving machine learning. In the coming sections we will discuss about two types of data IID and non-IID.

1.2.1 IID data

The term IID data stands for identically and independently distributed data. The term identically distributed means there are no trends and all the items are taken from the same probability distribution. Independent means items are not connected in any way. In distributed learning context, If the data is randomly distributed to different clients then client data should follow the same distribution as of parent.

$$X_i \in D(Distribution) \tag{1.1}$$

$$\forall X_i \neq X_j, P(X_i, X_j) = P(X_i) \cdot P(X_j) \tag{1.2}$$

1.2.2 Non-IID data

Federated learning involves two levels of sampling (1) sampling the client $i \sim C$ and (2) Sampling data X $\sim$P$_i$. Non-IID data in federated learning means the distributions P_i and P_j are different. Having IID data in distributed setup means that each micro batch of data used for client's model update is statistically similar to a micro batch from the complete training dataset. If achieving IID data is possible then we can achieve similar accuracy with fedrated and central learning.

1.3 Algorithm

Federated learning uses gradient update from each client to compute gradient update for the global model, this approach enables us to do machine learning without transferring data. Fedavg[9] algorithm uses average of all client's model parameters to update global model. The algorithm 1 summarizes the Fedavg algorithm.

Algorithm 1 Federated Learning with clients C, minibatch size B, learning rate η

Server execution:
initialize W_1
for each round t = 1, 2, . . . **do**
    for each client $k \in C$ in parallel **do**
        $W_{t+1}^k \leftarrow$ ClientUpdate(k,W_t)
        $W_{t+1} \leftarrow \sum_{k=1}^{C} W_k^{t+1}/C$
    end for
end for
ClientUpdate(k,W):
$B =$ Split data into batches of size B
for batch $b \in \{B\}$ **do**
    $W \leftarrow W - \eta \delta l(w; b)$
end for
send W to server

1.4 Conclusion

Federated Learning is the first globally accepted norm for PPM[1]L. The federated discussed here is just the first step in PPML(Privacy Preserving MAchine Learning). Results of federated learning will be discussed in coming chapter along with other approaches. Federated Learning can be used on Non-IID data but it's performance depends on how near the data is to IID assumptions. Also, there are many additions to make federated more secure as mentioned in [2]. In the coming chapters more approaches will be introduced for PPML.

Chapter 2

Fusion Learning

2.1 Introduction

Federated Learning is the new norm in Privacy Preserving Machine Learning. Federated Learning takes steps towards securing user privacy by just transmitting the model parameters instead of sensitive user data. But, when the federated networks are deployed at scale, millions and billions of devices are involved in the network, continuously communicating with central server resulting in waiting and efficiency issues when the network is slower or congested. So if we were to use federated learning at scale, we need to think of approaches reducing the communication overhead between the client and the server, and reduce dependency between clients located in different geographical locations. Considering these communication issues in federated learning we propose Fusion Learning [7] an alternate approach for privacy preserving machine learning. In Fusion Learning we reduce the communication overhead by just contacting the server once by sending the data distribution parameters and model parameters required for rebuilding the masked noisy client dataset.

2.2 Algorithm

The figure 2.1 depicts the Fusion Learning architecture. Fusion's algo mainly contains three steps

1. Clients send model parameters and distribution parameters to the server
2. Generating data at the server from the information shared by clients
3. Training a global model from the generated dataset and updating the clients with a new global model

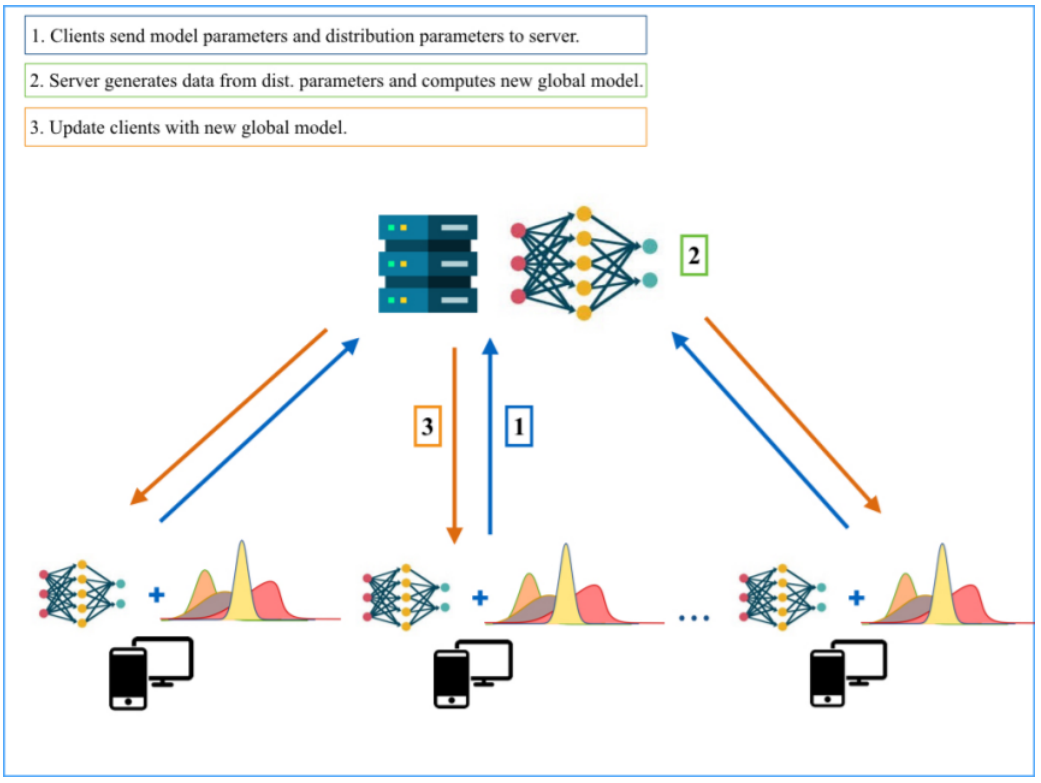


FIGURE 2.1: Fusion Learning Architecture

TABLE 2.1: Different types of distributions used to verify the distribution of individual feature set

norm	pareto	genextreme	gamma	uniform
exponweib	lognorm	expon	logistic	vonmises
weibull max	beta	cauchy	lomax	wald
weibull min	chi	cosine	maxwell	wrapcauchy
chi2	pearson3	powerlaw	rdist	erlang

2.2.1 Client's message

The client's message to the server consists of model distribution parameters and machine learning model parameters required to rebuild the dataset. Each feature in the dataset gets a distribution from the table 2.1. To determine which distribution fits a feature best, we use the P-value from *Kolmogorov-Smirnov* test and to determine best parameters for a distribution we use Maximum Likelihood Estimation. Now that we have feature distributions for each feature we need a model to classify these features for a prediction label. So we build a model from the dataset to classify data generated from feature distributions.

2.2.2 Server computation

The central server collects messages from all the clients and generates a new dataset from the shared information. At the server feature dataset is randomly generated from feature distributions of each client, and they are labelled using corresponding client's machine learning model. After generating datasets from each client, all these datasets are merged to form a combined global dataset, which is then used to build a global machine learning model. This global model is then later sent to clients for local usage. The above discussed steps are mentioned in Algorithm 2.

The steps discussed require only one round of communication between a client and the server resulting in a reduced communication overhead and easier maintenance of the system involving billion clients. Although it reduces network cost, generating a dataset at the server and finding feature distributions at the client might add computation time.

Algorithm 2 Fusion Learning

Client Update:
for $i \in \{1 \ \ to \ \ F \ \ \}\forall \ \ features$ **do**
    a. calculate the p value of each distribution using K-S test
    b. find the maximum value from the above list to indicate its feature
    c. store the distribution parameters for that feature
end for
for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ **do**
    for $x \in \{1 \ \ to \ \ X \ \ \}\forall \ \ inputs$ **do**
        Update weights given by:

$$\theta^k = \theta^k - \eta\delta L_k(\theta^k, b)$$

$$where \ \ \theta = weightvector, \eta = learningrate, L_k = Loss$$

    end for
end for
store the final weights
send distribution parameters and model parameters to server

Server Update:
for $i \in \{1 \ \ to \ \ C \ \ \}\forall \ \ clients$ **do**
    a. generate points for each distribution feature
    b. find predicted value for these points using model parameters
end for
$D_s = \bigcup_{i=1}^{C} D_i$ *//merge data points from all clients*
build a neural network model on the above dataset
transmit back the new global model parameters to the clients

TABLE 2.2: Dataset description

Dataset	Instances	Features
Credit Card	30000	24
Breast Cancer	569	9
Gender Voice	3169	20
Audit Data	777	18

2.3 Experimetal Results

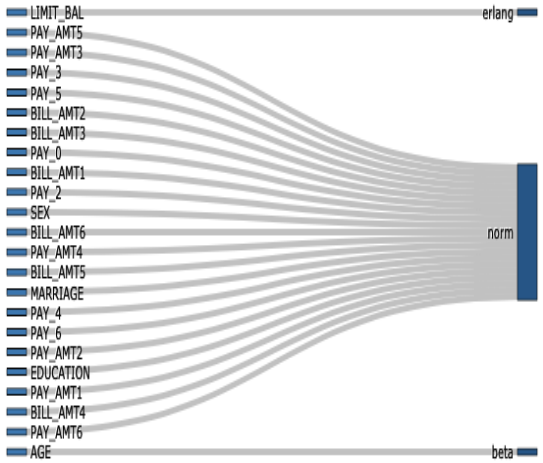
The correctness of Fusion Learning algorithm is tested on four different datasets from UCI repository [4], namely Audit, Credit Card, Gender voice and Breast Cancer dataset. It is important to notice the size of the datasets used from the Table 2.2 as it makes the testing more reliable. Also, Fusion's correctness is tested by comparing it with Federated Learning and Central Learning on each dataset. Central Learning setup is where machine learning is done by transmitting the client data to the server for learning. Although central learning requires only one communication round, it has a huge network cost because of the size of the data being transmitted. Central Learning depicts how machine learning was done before there were any privacy concerns.

2.3.1 Feature Distributions

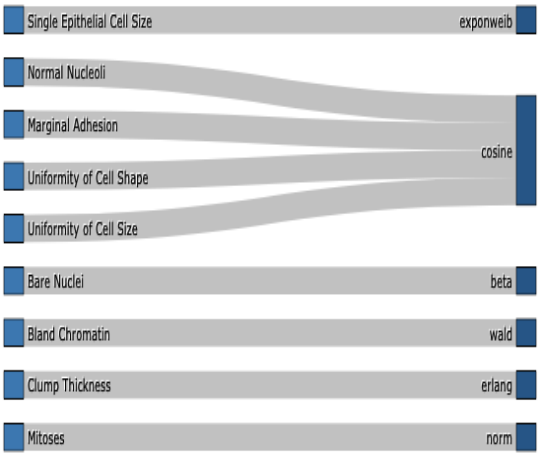
Every datset contains features and these features follow a particular distribution to some extent based on the quality of the data. To find out these feature distributions we use methods described in the section 2.2. The Figure 3.2 shows mapping between dataset features and their corresponding feature distributions. This mapping may change based on the distribution set your algorithm is using, whereas for the testing purposes we have used distributions from Table 2.1 as feature set. Also, some pre-processing or expert help in determining the distribution set will result in a reduced client data processing.

2.3.2 Local and Global Models

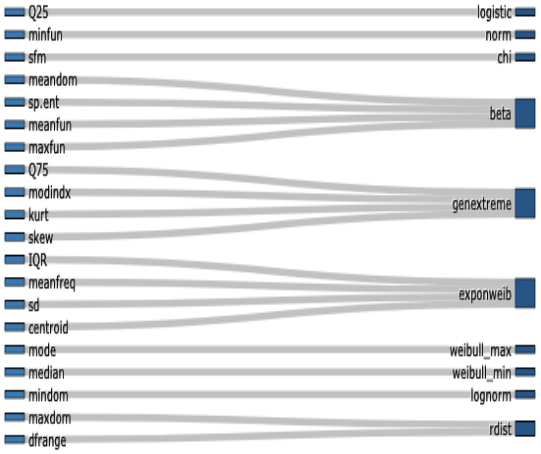
In our experiments we use a neural network with two hidden layers and hundred hidden nodes as the machine learning model. Same machine learning model is used at both client and server location. A train test split of (80,20) is used for training both local and global model. In the testing setup all the three approaches Fusion, Federated and Central use a 10 client setup.



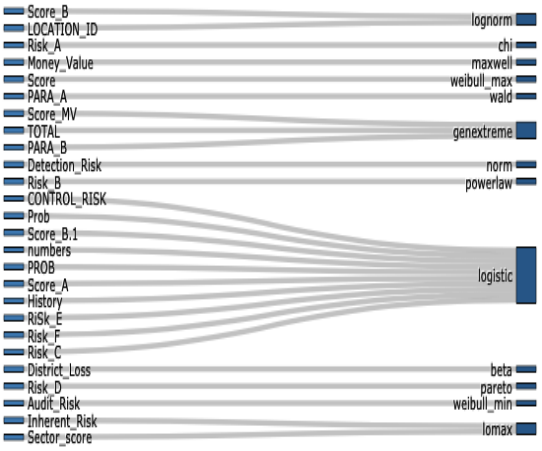
(A) Credit Card Data.



(B) Breast Cancer Data.



(C) Gender Voice Data.



(D) Audit Data.

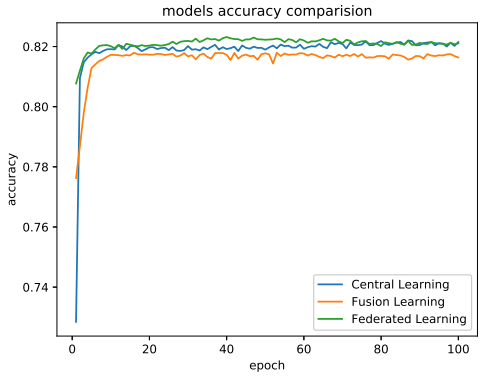
FIGURE 2.2: Distribution of each feature for Credit Card, Breast Cancer, Gender Voice and Audit datasets

2.3.3 Training and Testing Accuracies

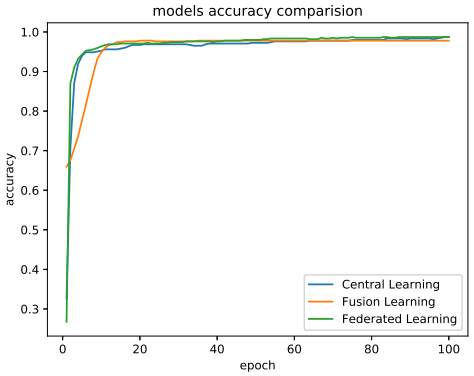
It is important to note that the training accuracy of the fusion learning approach is the testing accuracy because the model is not trained on the original data, but instead, it is trained on the data generated from the distribution of features of each client.

TABLE 2.3: Comparison of training accuracies (in %) between Central Learning, Federated Learning and Fusion learning

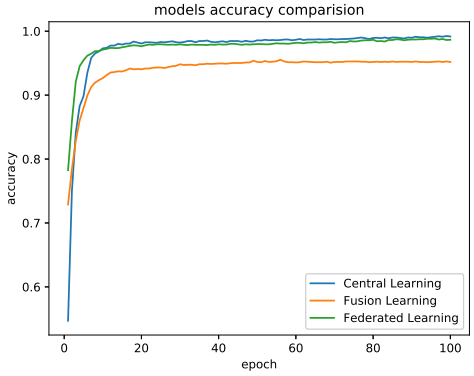
Dataset	Central Learning	Federated Learning	Fusion Learning
Credit Card	81.11	81.60	81.09
Breast Cancer	97.08	96.35	95.62
Gender Voice	96.84	97.31	94.32
Audit Data	98.06	98.71	97.42



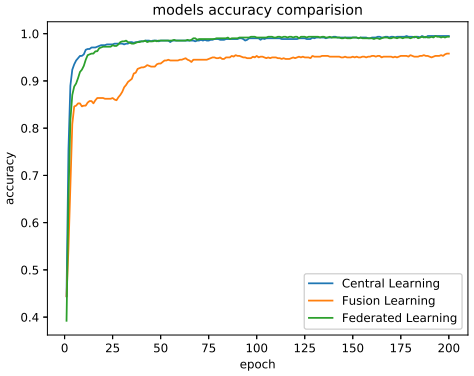
(A) Credit Card Data.



(B) Breast Cancer Data.



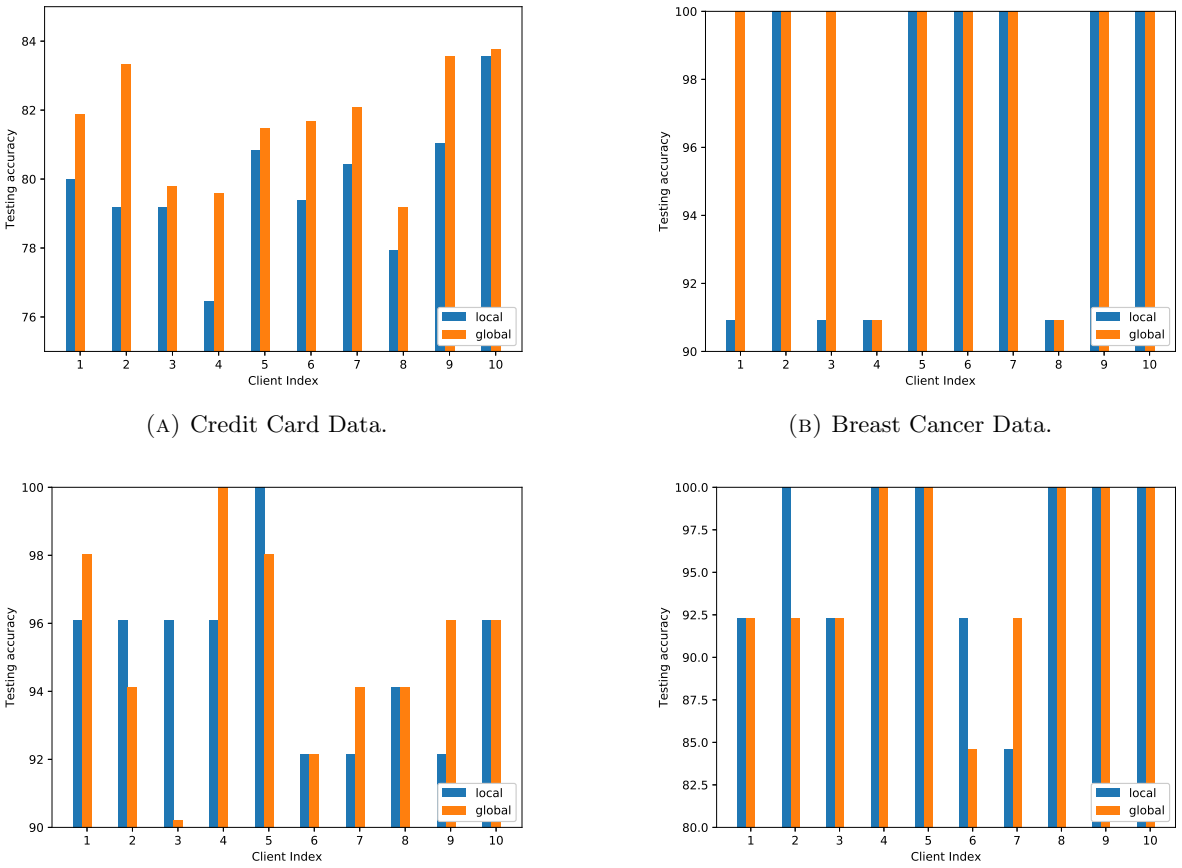
(C) Gender Voice Data.



(D) Audit Data.

FIGURE 2.3: Comparison of Training accuracy between Centralized Learning, Federated Learning and Fusion Learning algorithms

The training accuracies of all three frameworks have been illustrated in Figure 2.3 and summarized in Table 2.3.We can see from this table that the training accuracies of fusion learning framework fall slightly below those obtained from both federated and a centralized setup. This is because the quality of the data generated is not on par with the original data. We can also notice that there is a subtle difference in accuracies of Credit Card, Breast Cancer, and Audit Data sets between Federated and Fusion Learning algorithms, whereas the accuracy of the Gender Voice dataset, is slightly lesser. The accuracies of such datasets can be increased by adding more distributions because determining the right distribution plays an important role in generating artificial data. Also, more data at the client node helps in determining the corresponding feature distribution parameters with more confidence, which results in an increase in the quality of the generated data. As can be seen from Figure 2.4, we have also compared the accuracies on each client node obtained using the local model and the global model built using the fusion learning framework. We see that in all the datasets, for all clients, the global model either outperforms the local model or achieves similar accuracies, which is also the case for a federated setup.



(A) Credit Card Data.

(B) Breast Cancer Data.

(C) Gender Voice Data.

(D) Audit Data.

FIGURE 2.4: Comparison of Testing accuracies of initial local model vs final global model at each client

2.3.4 Communication Efficiency

The main aim of Fusion Learning approach is to introduce a privacy preserving machine learning approach reducing the communication rounds involved in the federated learning setup. Federated Learning takes E communication rounds per client to reach convergence whereas Fusion Learning takes only one communication round to complete the setup. The size of the message shared between the client and the server is similar for both federated and fusion. These metrics are summarized in Table 2.4.

TABLE 2.4: Network usage of Federated Learning and Fusion Learning for E epochs for a single client

Approach	Network Calls	Data exchanged
Federated Learning	2*E	Model parameters
Fusion Learning	2	Model params + feature distr parameters

2.4 Issues

Fusion Learning might not work on every kind of dataset, the algorithm works well when the feature distributions parameters capture the dataset accurately. If the feature distribution parameters fail to capture the dataset, the quality of the generated data at the server is sub par resulting in poor performance. To show this drawback we use MNIST image dataset which is difficult to represent using feature disributions. Table 2.5 show the testing accuracy of the three approaches on MNSIT, as suspected Fusion performs poorly. This poor performance of fusion is mainly because we use each pixel as a feature, the pixel feature is a binary variable only taking 0 or 1. The pixel feature follows a discontinuous distribution (bernouli) kind of like a coin toss. As a result the generated data at the server does not represent MNIST image datset leading to poor model performance.

Table 2.5: MNIST

Dataset	Central Learning	Federated Learning	Fusion Learning
MNIST	98.56	98.05	71.4

Chapter 3

Hybrid Fusion Learning

3.1 Introduction

In the previous chapters two privacy preserving machine learning approaches were introduced, both these approaches have their issues with client's geographical location, client's network connection and single a point of failure at the server. To mitigate these issues, Hybrid Fusion Learning, a new PPML model using client-edge-server architecture is introduced. Hybrid Fusion Learning incorporates fusion Learning at the client-edge level and federated learning at the edge-server level. This approach tries to reduce the load on server by adding edge servers, making it more accessible to clients with different geographical background.

3.2 Architecture

Figure 3.1 shows the architecture of hybrid fusion learning containing three layers (1) client layer, (2) edge layer, (3) cloud layer. It is divided into these layers based on the network speeds (1) low latency (edge-cloud) (2) high latency (client-edge). This division is very convenient because we can cluster clients based on their device's sleep time, geographical location and privacy policy. Fusion learning is performed between edge server and it's corresponding clients to reduce communication cost whereas federated is performed between the edge servers and the cloud owing to federated learning's better performance in low latency networks.

3.2.1 Client layer

A group of client nodes communicates to the edge layer using fusion learning, where each client computes its local model parameters along with the distribution parameters Ψ. The

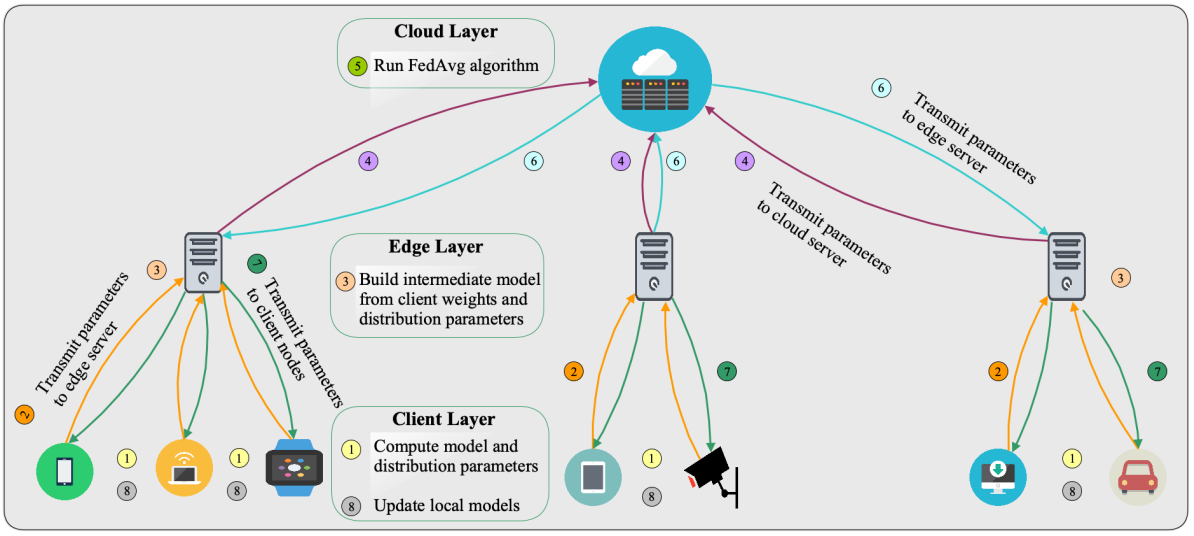


FIGURE 3.1: Architectural diagram of a hybrid fusion learning system consisting of three layers, Client, Edge and Cloud.

first step in this process is to find out what distribution each feature follows. In order to do so, Kolmogorov-Smirnov ($K-S$) test [3] is used to compare the sample data with reference probability distributions and computes the probability (p) value for each distribution. The value of p indicates the similarity of sample data with a given distribution. The probability distribution with the highest p value is selected. All the clients build a local model and generate the model vector. The model parameters along with the distribution parameters are transmitted to their corresponding edge server as described in Algorithm 3.

3.2.2 Edge layer

The edge server receives the distribution and the model parameters from its clients. It uses the distribution parameters to generate sample data points and the weights to compute the predicted outcomes of the generated data. Once the edge server generates the points from each client, it merges all these points to create a large dataset. This large dataset is now used to create an intermediate machine learning model. Such models are created across all edge servers. These edge servers participate in federated learning by transmitting their intermediate model parameters to the cloud server. This process is repeated for every epoch as detailed in Algorithm 4. Since the communication latency from the clients to the edge server is smaller than that to a cloud server, the transmission cost is significantly reduced. Besides, the computation load on a single cloud server is also reduced by dividing the computation among different edge servers.

3.2.3 Cloud layer

The cloud server's primary purpose is to aggregate the model weights received from the edge server and transmit them back, as shown in Algorithm 5. It uses FedAvg algorithm to aggregate the model parameters. After aggregation, the final values are passed back to the edge nodes. This process is repeated until the desired accuracy is achieved.

Algorithm 3 Hybrid fusion learning: Client Update

Client Update:
for $i \in \{1 \ \ to \ \ F \ \ \}\forall \ \ features$ **do**
    a. calculate the p value for each distribution using K-S test
    b. find the distribution ψ_i, with max 'p' value
end for
for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ **do**
    for $x \in \{1 \ \ to \ \ X \ \ \}\forall \ \ inputs$ **do**
        Update weights given by:
        $\theta^{e+1} = \theta^e - \eta\nabla L(\theta^e)$
        where θ = weight vector, η = learning rate, and
        L = Emperical loss function
    end for
end for
send $[\Psi, \theta^{e+1}]$ to server

Algorithm 4 Hybrid fusion learning: Edge Update

for $i \in \{1 \ \ to \ \ C \ \ \}\forall \ \ clients$ **do**
    a. generate points from feature distribution
    b. find predicted value for the generated points
    using θ_i
end for
$D_s = \bigcup_{i=1}^{C} D_i$ *//merge data points from all clients*
for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ **do**
    for $d \in \{1 \ \ to \ \ s \ \ \}\forall \ \ D_s$ **do**
        Update weights given by:
        $\theta^{e+1} = \theta^e - \eta\nabla L(\theta^e)$
        where θ = weight vector, η = learning rate, and
        L = Emperical loss function
    end for
    transmit θ^{e+1} to cloud
end for

3.3 Experimental Results

In this section information about architecture parameters, metrics and results are presented

Algorithm 5 Hybrid fusion learning: Cloud Update

for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ **do**
  for $k \in \{1 \ \ to \ \ K \ \ \}\forall \ \ EdgeServers$ **do**

$$\theta^{avg} = \frac{1}{K}\sum_{k=1}^{K}\theta_k$$

    //avg weights from all edge servers
  end for
  transmit θ^{avg} to edge server
end for

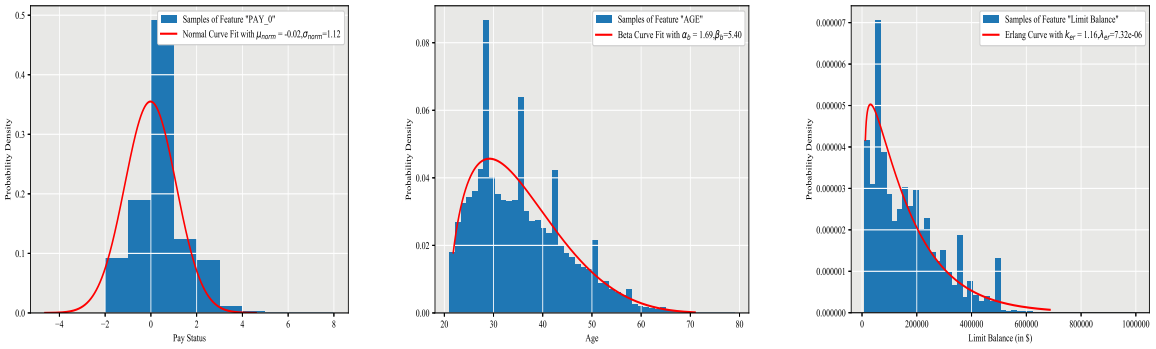
TABLE 3.1: Dataset description

Dataset	Instances	Features	Distributions
Credit Card	30000	24	22 - Normal, 1 - Beta, 1 - Erlang
Adult	48842	14	12 - Normal, 1 - Beta, 1 - Pearson3
Bank Marketing	45211	17	15 - Normal, 1 - Lognormal, 1 - Exponential Weibull

3.3.1 Setup

To test HFL, 100 clients were used varying the edge servers count. This was done mainly to see how different edge servers setup affects the accuracy and computation time. Three HFL setups were used with edge servers count set to 2, 5, 10. Different HFL setups contain different no of clients under each server because client count was not changed. More edge servers leads to less clients per edge server, this will decrease the time taken but increases the server cost. So determining the right amount of edge servers is important to make HFL work in scale.

Three datasets were used to check the performance of HFL namely: Credit Card, Bank Marketing, and Adult Datasets from UCI Repository [4]. The Credit Card dataset consists of 30,000 instances and 24 features. The distribution of the individual features of the data set is calculated using the Fusion algorithm. Figure 3.2 shows an example with the distribution for three features of Credit Card data set. Overall, 22 features follow a normal distribution, whereas the remaining two features follow Erlang and beta distributions [3]. The Adult dataset consists of 48,842 instances with 14 features, 12 of which follow normal distribution while the other two follow beta and Pearson3. The final Bank Marketing dataset consists of 45,211 instances with 17 features, with 15 features following normal distribution while the remaining two follow lognormal and exponential weibull distribution. This information has been consolidated in Table 3.1.



(A) The normal distribution of feature (Pay_0) from Credit Card dataset with $\mu_{norm} = -0.02$ and $\sigma_{norm} = 1.12$. Overall 22 features from Credit Card dataset follow the normal distribution.

(B) The beta distribution of feature (AGE) from Credit Card dataset with $\alpha_b = 1.69$ and $\beta_b = 5.4$. Overall 1 feature from Credit Card dataset follows the beta distribution.

(C) The Erlang distribution of feature (Limit Balance) from Credit Card dataset with $k_{er} = 1.16$ and $\lambda_{er} = 7.32e-06$. Overall 1 feature from Credit Card dataset follows the Erlang distribution.

FIGURE 3.2: Distributions of three features of Credit Card dataset.

3.3.2 Performance Metrics

The two metrics taken for this analysis are (1) time taken and (2) testing accuracy. To calculate the time taken we add two variables L1(latency between edge and cloud) and L2 (latency between client and edge). Fusion and federated learning are used to compare these results.

$$Timetaken = L1 * (FederatedCommunicationrounds) + L2 * (FusionCommunicationrounds) + (Fusiontime) + (Federatedtime) \tag{3.1}$$

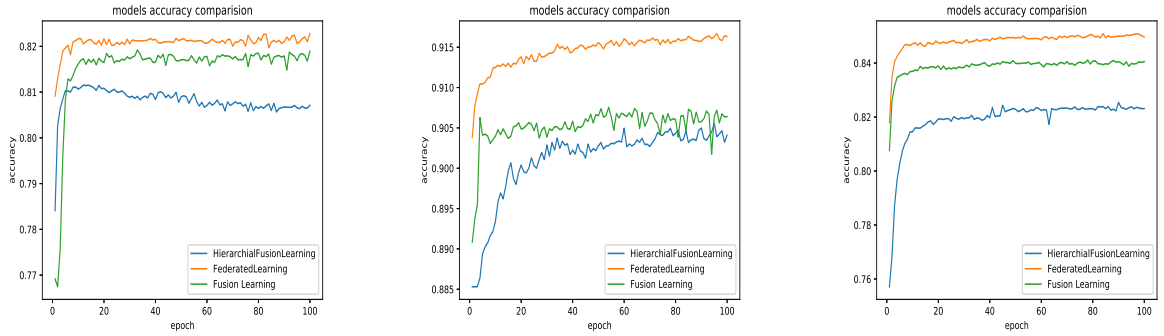
3.3.3 Results

Table 3.2 summarizes the total time taken and the accuracies of the proposed hybrid fusion learning with different edge servers. To calculate the time taken in the table 3.2, communication rounds and latency were not involved, this was done mainly to allow user to set latency values (L1, L2) based on their setup. The results are compared with federated and fusion learning techniques. In this table, K indicates the number of edge servers and C indicates the number of clients per edge server. In both federated and fusion learning, the number of edge servers is zero as the parameters are transmitted directly to the cloud and hence $C = 100$. The results in table 3.2 are not looking good because those results involve raw time without communictaion rounds, if we assume that the the communication latency between the client and the cloud server is

TABLE 3.2: Total time taken and accuracy with hybrid fusion, fusion, and federated learning.(*ime for Communication rounds not added)

Datasets	Credit Card		Adult		Bank Marketing	
	Total Time (s)	*Accuracy* (%)	*Total Time* (s)	*Accuracy* (%)	*Total Time* (s)	*Accuracy* (%)
C=50, K=2	88.36	80.9	439.8	90.2	211.32	82.2
C=20, K=5	69.26	80.6	250.8	90.1	120.12	81.9
C=10, K=10	63.66	76.9	186	86.1	91.02	76.6
C=100, K=0 (*Fusion*)	156.54	81.8	394.11	90.4	235.90	83.8
C=100, K=0 (*Fed*)	5.45	82.4	11.66	91.5	7.88	84.6

approximately ten times larger than that to the edge server [8] then HFL results will look much better than federated and fusion.



(A) Credit Card Data. (B) Bank Marketing Data. (C) Adult Data.

FIGURE 3.3: Testing accuracy for hybrid fusion, fusion and federated learning for Credit Card, Bank Marketing and Adult datasets.

Chapter 4

Privacy Preserving Machine Learning with GAN

4.1 Introduction

In fusion learning we try to address some of the issues from federated learning but fusion learning cannot be applied everywhere. Fusion Learning performs poorly if we cannot recreate important characteristics of dataset from it's feature distribution. For example, MNIST image dataset where each pixel is considered as a feature, where finding distribution parameters for features will not represent characteristics of the whole image. This disability to reproduce certain type of data results in poor model performance. To mitigate the issues in Fusion, a framework using General Adversarial Networks [6] is introduced to reproduce data. GAN's are known for their ability to reproduce realistic fake data, which is exactly what we want in privacy preserving machine learning

4.2 General Adverserial Networks

GAN is a new framework for creating generative models using adverserial training. It consists of a generator G which tries to generate data similar to the data distribution, and a discriminator D that tries to discriminate generated data G from the original data distribution.The goal of G is to maximize the probability of D making a mistake. The framework resembles a mini-max game between generator and discriminator. The GAN model reaches convergence when the discriminator cannot distinguish the data distribution from the generated distribution G. This is pictographically demonstrated in Fig 4.1[6]. When the model has achieved convergence the generator distribution resembles the data distribution and the discriminator display 0.5

everywhere. If both generator and discriminator are multilayer perceptron, then training can be done using backpropogation.

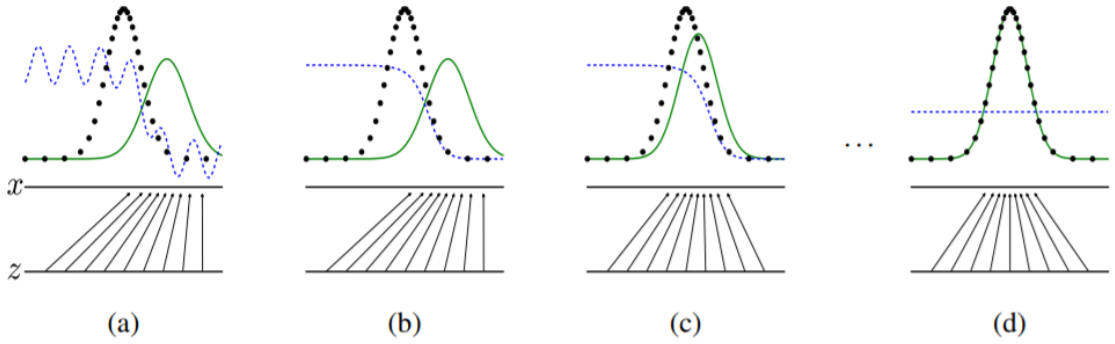


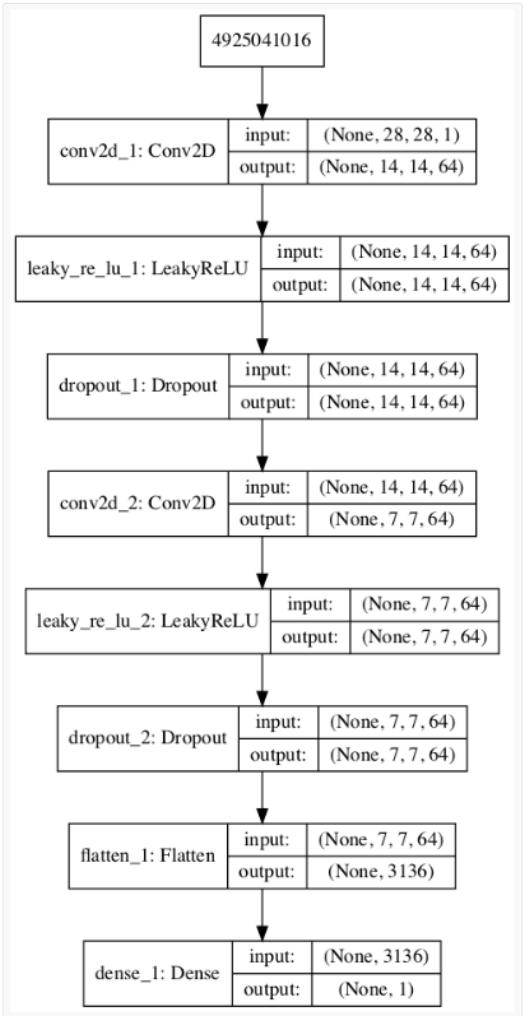
FIGURE 4.1: Pictorial representation of GAN model reaching convergence.Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping x = G(z) imposes the non-uniform distribution pg on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. D(x) = 0.5 .[5]

4.2.1 GAN model construction

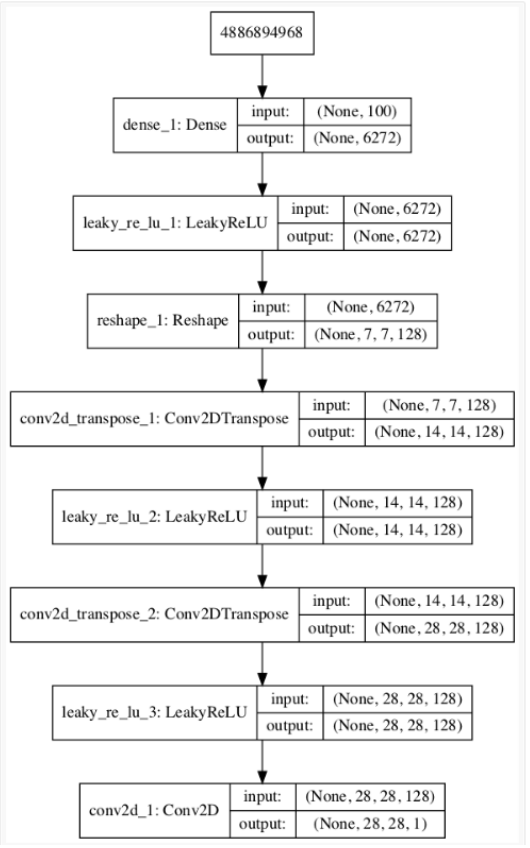
For generator we take input from a random point in latent space of dimension 100 (Eg. 100 feature input vector) and output a gray scale image of 28*28 pixels. Whereas for the discriminator we take input from a gray scale image of 28*28 pixels and output a value between 0 and 1 (probability that input belongs to data distribution). In the GAN model first layer contains generator models and the second layer contains discriminator model, back propagation can be used to update the generator based on the discriminator's feedback. The Figure 4.2 summarizes the model architecture for the generator, discriminator and the GAN model.

4.3 Algorithm

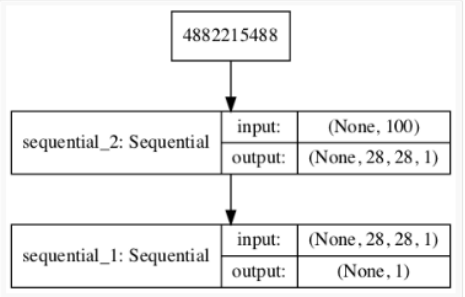
The algorithm is similar to that of fusion, but instead of finding distributions to generate data, GAN's generator is used to do the work for corresponding client. The Algorithm 6 summarizes fusion GAN algorithm.



(A) Discriminator architecture.



(B) Generator architecture.



(C) GAN model architecture.

FIGURE 4.2: Model architecture

4.4 Experimental Results

For the proof of correctness we have tested Fusion GAN using MNIST digits dataset with ten clients. In the future work we plan to extend this to normal classification datasets like (Credit Card). The Figure 4.3 shows the evolution of a clients' generator with epochs, it can be observed

Algorithm 6 Fusion GAN

Client Update:
Train the generator g using GAN
for $e \in \{1 \ \ to \ \ E \ \ \}\forall \ \ epochs$ **do**
  for $x \in \{1 \ \ to \ \ X \ \ \}\forall \ \ inputs$ **do**
    Update weights given by:

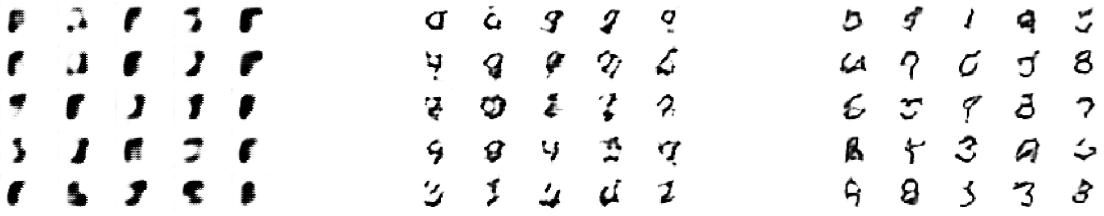
$$\theta^k = \theta^k - \eta\delta L_k(\theta^k, b)$$

$$where \ \ \theta = weightvector, \eta = learningrate, L_k = Loss$$

  end for
end for
store the final weights
send the generator parameters and model parameters to server

Server Update:
for $i \in \{1 \ \ to \ \ C \ \ \}\forall \ \ clients$ **do**
  a. generate points from the client's generator function
  b. find predicted value for these points using client's model parameters
end for
$D_s = \bigcup_{i=1}^{C} D_i$ *//merge data points from all clients*
build a neural network model on the above dataset
transmit back the new global model parameters to the clients

that after training for 200 epochs the generated images were quite clear, this results in high quality data for the server. Although images were more clear at 200 epochs, the testing accuracy does not change much between blurry(100 epochs) data and clear(200 epochs) data. Considering this observation we can send blurry images to the server not compromising on the privacy and still get accurate models. The table 4.1 compares the performance of fusion GAN with the existing approaches.



(A) 10 epochs. (B) 100 epochs. (C) 200 epochs.

FIGURE 4.3: Generated images from client's generator

Table 4.1: MNIST

Dataset	Central Learning	Federated Learning	Fusion Learning	FGAN
MNIST	98.56	98.05	71.4	96.9

4.5 Conclusion

Fusion GAN overcomes some of the short comings in the fusion and proposes more generic framework to generate data. Although there are some privacy concerns in Fusion and FGAN, we can always overcome them with more secure message passing protocols and adding noise to mask the original data.All the approaches mentioned so far have their issues but it is a good first step towards achieving the final goal, Privacy Preserving Machine Learning (PPML).

Bibliography

[1] Mohammad Al-Rubaie and J. Morris Chang. "Privacy-Preserving Machine Learning: Threats and Solutions". In: *IEEE Secur. Priv.* 17.2 (2019), pp. 49–58. DOI: 10.1109/MSEC.2018.2888775. URL: https://doi.org/10.1109/MSEC.2018.2888775.

[2] Keith Bonawitz et al. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 1175–1191. DOI: 10.1145/3133956.3133982. URL: https://doi.org/10.1145/3133956.3133982.

[3] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley, 2012. ISBN: 9780321500465. URL: https://books.google.co.in/books?id=4TlEPgAACAAJ.

[4] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[5] Ian Goodfellow et al. *Deep learning. vol. 1*. 2016.

[6] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[7] Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota. "Fusion Learning: A One Shot Federated Learning". In: *Computational Science - ICCS 2020 - 20th International Conference, Amsterdam, The Netherlands, June 3-5, 2020, Proceedings, Part III*. Ed. by Valeria V. Krzhizhanovskaya et al. Vol. 12139. Lecture Notes in Computer Science. Springer, 2020, pp. 424–436. DOI: 10.1007/978-3-030-50420-5_31. URL: https://doi.org/10.1007/978-3-030-50420-5_31.

[8] L Liu et al. "Client-edge-cloud hierarchical federated learning". In: *arXiv preprint arXiv:1905.06641* (2019).

[9] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Artificial Intelligence and Statistics*. 2017, pp. 1273–1282.

[10] H. Brendan McMahan et al. “Federated Learning of Deep Networks using Model Averaging”. In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629.