A study of privacy-preserving mechanisms in Tier-Based Federated Learning

Sundaresan G
Dept. of CDS
IISc

Ajeya B S
Dept. of CDS
IISc

https://github.com/ajeyab22/fedml-privacy-mechanisms.

***Abstract*—Federated learning is a distributed machine learning approach that enables multiple parties to collaboratively build a model without sharing their private data with a central server. Each party trains a local model on its own data, and only the model parameters are communicated to the central server for aggregation. Federated learning has potential to address scalability challenges in machine learning.**

However, simply maintaining data locality during training processes does not provide sufficient privacy guarantees. We need a system capable of preventing inference over both the messages exchanged during training and the final trained model while ensuring the resulting model has acceptable accuracy. Other challenges that need to be addressed in order to make Federated learning practical for real-world scenarios includes handling of stragglers (slow clients). These stragglers lead to increase in overall training time.

In this paper, we aim to address the mentioned challenges by using a synchronous approach that separates the model training and aggregation processes. Tier based approach (TiFL [1]) is used for tackling the stragglers. The effect of Homomorphic Encryption (Pyfhel [2]) and Differential Privacy on the overall training time and final accuracy is experimentally evaluated.

I. Introduction

In traditional centralized machine learning, data is collected and stored in a centralized location, and the model is trained on this data. However, this approach may not be feasible or desirable in many situations, such as when the data is too large to be transferred or when the data is distributed across multiple organizations or individuals who cannot share their data due to privacy or legal concerns.

Federated learning addresses this challenge by allowing multiple devices or clients to collaboratively train a model without sharing their raw data. The basic idea is to distribute the model training process among the devices or nodes that have local data, and only exchange a small portion of the model parameters or updates instead of the raw data. This enables privacy-preserving training of models on decentralized data, which is a necessity in applications such as healthcare, finance, and so on. The model is trained locally on each device using its own data exclusively, and the updates are aggregated in a secure and privacy-preserving manner to update the global model.

This presents a scalability problem because federated learning requires coordination (synchronization) among a large number of devices or clients, which can result in communication overheads. Moreover, the time for each round is dictated by the slowest client (straggler) in a synchronous approach. Additionally, privacy concerns require that the data remains local to each device and that the model updates are securely aggregated without revealing any sensitive information. Therefore, developing efficient and scalable algorithms and architectures for federated learning is necessary.

Tier-Based Federated Learning (TiFL [1]) tackles the issues of stragglers, data, and resource heterogeneity. Its adaptive tier selection policy helps handle these issues by classifying the clients into logical tiers. This logical tier classification is done by a tiering module at the start by measuring each client's latency response.

To address the privacy aspects, we assume the model of a Curious server with no adversaries. This assumption is justified in the sense that the data transferred between clients and servers are encrypted by existing protocols such as HTTPS. Hence the possibility of an external adversary decrypting the data by eavesdropping is minimal. However, we assume honest clients and servers, i.e. no internal adversary. In this paper, efforts are taken to ensure that the curious server cannot estimate the data (accurately) by the weights received from clients. In the available plethora of existing privacy-preserving algorithms, this study is focused on Homomorphic encryption (Pyfhel) and Differential Privacy.

Homomorphic encryption is the conversion of data into ciphertext that can be analyzed and worked with as if it were still in its original form. Homomorphic encryption enables complex mathematical operations to be performed on encrypted data without compromising the encryption. This is useful in the sense that each client could send the encrypted model weights to the server. The centralized server could perform aggregation without ever knowing the model weights of individual clients. Moreover, it would not know the global weights too. After the aggregation, the clients receive, carry out decryption to obtain the global weights. This is a powerful technique ensuring privacy to a great extent. However, this method suffers from encryption, and decryption overheads (both time and communication bandwidth).

Differential Privacy, unlike homomorphic encryption, adds noise to model weights before sending to the server. Hence, server could see only distorted data. This is based on the principle that when server carries out aggregation on several (sufficiently high) clients, the noise cancels out. Though, this

approach does not suffer from the aforementioned drawbacks, this too has challenges. One challenge is the amplitude of allowable noise; too little reveals the data and too high leads to severe distortion and further affecting the global convergence of weights.

Section II provides the findings of the recent works in Federated Learning. Section III discusses the Methodology of our project. Section IV covers the experiments we would be doing to evaluate our methodology.

II. Related Work

TIFL, a Tier-based Federated Learning System [1], divides clients into tiers based on some measure metrics that help profile the device, then selects one tier per round and trains data using an adaptive algorithm that helps tackle both data and device-heterogeneity. Stacey Treux et.al [3] present an approach to federated learning that utilizes differential privacy and secure multiparty computation to balance the trade-offs between privacy and accuracy. But the implementation is not on Tier-based FL, which we approach. Alberto Ibarrondo et.al describe the implementation of Pyfhel [2], a Python library for fully homomorphic encryption. Pyfhel is designed to be user-friendly and provides an interface for various fully homomorphic encryption schemes. The library includes support for basic arithmetic operations on encrypted data, as well as comparison, input/output, and serialization functionalities. The paper also provides a detailed explanation of the underlying concepts of fully homomorphic encryption and the different schemes used by Pyfhel, making it a useful resource for those looking to understand the theoretical and practical aspects of FHE.

III. Methodology

In this study, the Federated Learning environment is simulated on a single computer with GPU. As mentioned earlier, privacy-preserving schemes are implemented on top of TiFL. The experiments are performed on datasets MNIST with models Logistic Regression (LR) and CNN.

A. TiFL

Since the experiments are carried out on a single computer, we need to simulate tiers with different latency responses. Since the computational capabilities are similar, we fix the threshold range to divide clients into tiers based on their data set size. Hence, a client with low data points shall be classified as a Fast tier. We use 3 tiers - fast, medium, and slow. Then for each round, exactly one of the tiers is selected for training. The overview is shown in figure 1.

B. Homomorphic Encryption

Pyfhel encryption library is used for this purpose. Since the weights are of floating type, we use CKKS (Cheon-Kim-Kim-Song) scheme. This works for floating objects array of length (size) 8192. Pyfhel uses FHE. We choose this over PHE

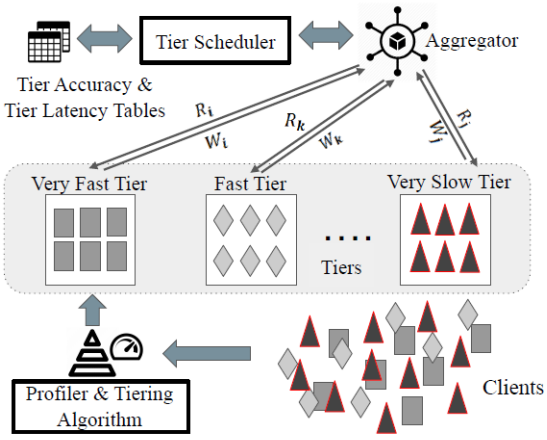


Fig. 1: Overview of TiFL

because other than better security, it allows for both addition and multiplication operations to be performed on encrypted data. As FHE is a complete encryption scheme, meaning that it can be used to perform any computation on the encrypted data, regardless of the complexity of the computation.

CNN model consists of 1,206,590 parameters distributed among layers. LR model consists of 7850 parameters. Since the weights are placed in several tensors of varied dimensions, each tensor is unrolled into a linear array. Then the encryption is applied. This is done to reduce the overall time for encryption. However, encryption puts quite a load on the computational resources of the client. Moreover, the overall training time is tremendously increased. Hence, for CNN, the encryption is limited to the first layer. This is under the hypothetical justification that the data is more prone to be revealed from the first layer weights than subsequent layer weights. However, for smaller model LR, all the parameters are encrypted.

The number of encrypted parameters for CNN = 8,510.
The number of encrypted parameters for LR = 7,850.

Model	N before encrypt.	N after encrypt.
CNN	1,206,590	8192*6 + 1,198,080 = 1,247,232
LR	7850	8192*2 = 16,384

TABLE I: Total size of weight matrix in multiples of 8 bytes (double).

Table I shows the number of parameters before and after encryption. Since each layer weight is encrypted individually, each weight length is increased from its own size to multiples of 8192 (encrypted text length).

C. Differential Privacy

For implementing Differential privacy, noise is added using Laplace distribution with sensitivity based on the average of individual weights divided by threshold (hyper-parameter). This threshold is varied to analyze the convergence and

subsequent accuracy. The sensitivity of a function f is the amount f's output changes when its input changes by 1. The scale of the noise will be calibrated to the sensitivity of f (input))/ ϵ. So this would mean that the more sensitive the input, the more noise would be added. Instead of setting hard-coded numbers as input, we make the sensitivity a function of the input data. As a base, choose the average of the difference of the array. We do not choose min or max as that would make the sensitivity skewed. Setting sensitivity to only the mean of diffs leads to a high noise being added leading to bad accuracy. After a few trials, we find mean/20 to be a good parameter for the two models we test on - CNN and LR.

IV. Experiments and Results

The experiments are carried out on AMD Ryzen 9 3900X 12-Core Processor with NVIDIA RTX3080 GPU and 32GB memory. As mentioned earlier, federated learning is simulated in this single computer.

A. Experimental Setup

For federated learning, we implement TiFL on the FedML framework. For both CNN and LR models, the MNIST data set is used. It consists of 1000 clients with varied data sizes. Based on their data sizes, they are logically partitioned into fast, medium, and slow tiers. Credits are assigned in the ratio of (0.5, 0.3, 0.2) to fast, medium, and slow respectively. Pyfhel CKKS is used for Homomorphic encryption and Laplace distribution is used for Differential Privacy.

B. Results

The training accuracy results for CNN and LR are shown in figures 2 and 3. As expected, differential privacy indicates the maximum variation.

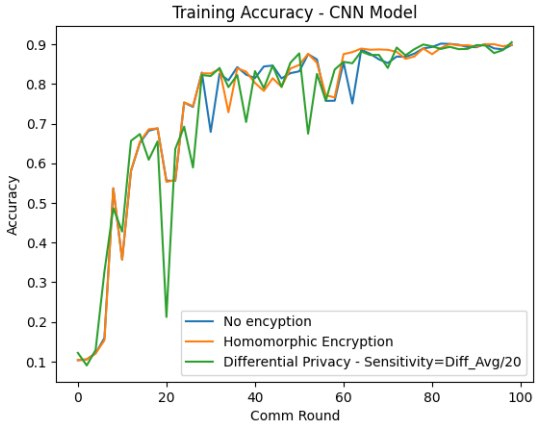


Fig. 2: CNN Training Accuracy

The testing accuracy results for CNN and LR are shown in figures 4 and 5.

The overall time comparison for various schemes is shown in figure 6.

The tierwise accuracy variation on the training data set is shown in figures 7.

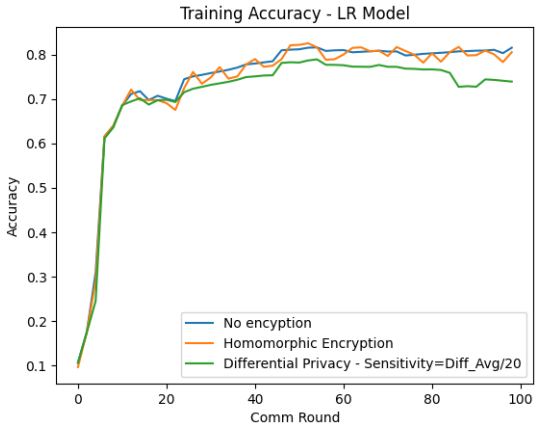


Fig. 3: LR Training Accuracy

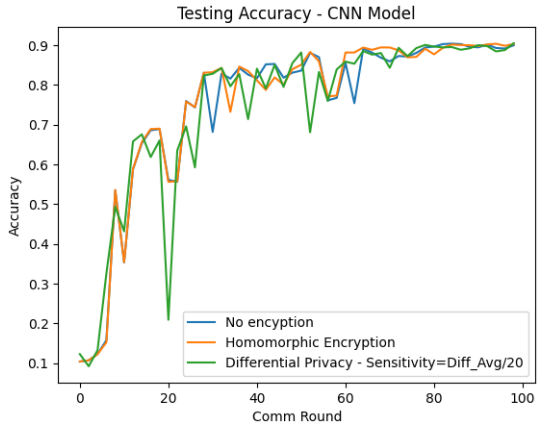


Fig. 4: CNN Testing Data Accuracy

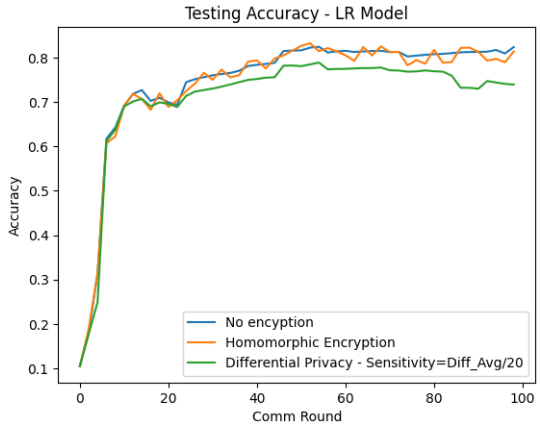


Fig. 5: LR Testing Data Accuracy

Also, the effect of sensitivity is evaluated for differential privacy and is shown in figures 8. Hence, an optimal value is chosen for both models.

Finally, the split-up times are shown in table

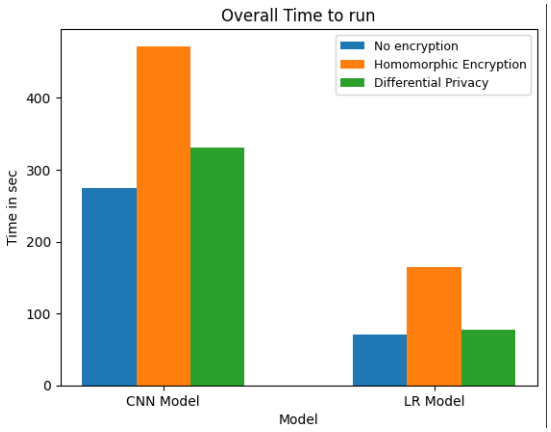
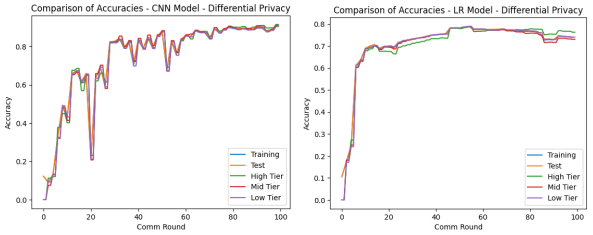
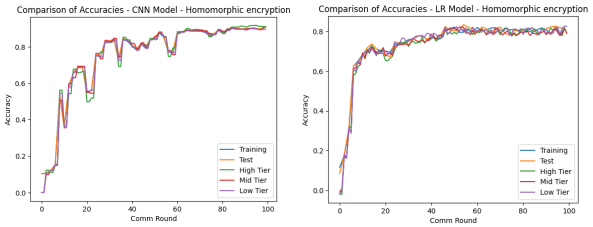


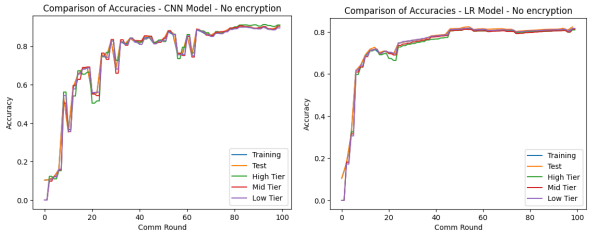
Fig. 6: Overall time for 100 rounds.



(a) CNN Differential Privacy. (b) LR Differential Privacy.



(c) CNN Homomorphic Encryption. (d) LR Homomorphic Encryption.

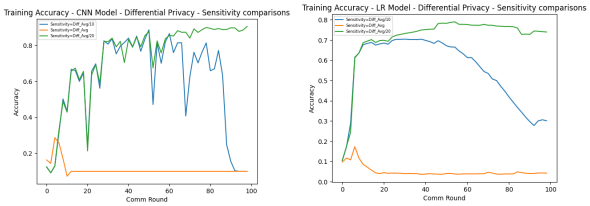


(e) CNN. (f) LR.

Fig. 7: Tierwise accuracy.

	H	DP	None
Avg Overall Time	471	331	273
Avg Enc Time	0.62	1.18	-
Avg Dec Time	0.2	-	-
Avg Aggr Time	0.21488	0.00791	0.00517

TABLE II: Split Up of Time taken for CNN model in sec



(a) CNN. (b) LR.

Fig. 8: Sensitivity effect on Differential Privacy.

	H	DP	None
Avg Overall Time	164	78	70
Avg Enc Time	0.13	0.32	-
Avg Dec Time	0.008	-	-
Avg Aggr Time	0.0701	0.00026	0.00024

TABLE III: Split Up of Time taken for LR model in sec

V. Conclusion

It can be observed that both Homomorphic encryption and Differential Privacy schemes apply well as an additive to TiFL. Homomorphic encryption (on partial weights) does not affect the accuracy or convergence, but the overall time taken increases by 72% for CNN and above 100% for LR model. Also, HE leads to 3.36% and 108% increase in communication bandwidth for CNN and LR respectively. Differential Privacy, though provides a similar overall time as no encryption, it is highly dependent on sensitivity specified. Hence, this limits the usage of differential privacy. To conclude, if the requirement is faster execution with a smaller level of privacy acceptable, Differential Privacy can be used, and if the requirement is stricter privacy with higher execution time and network being tolerable, Homomorphic encryption can be used.

In the future, other privacy-preserving schemes shall be evaluated. Further, the implementation of privacy schemes on Asynchronous Federated Learning frameworks such as FedAT [4] shall be studied. Packing of weight parameters for efficient Pyfhel encryption to reduce the overall time and communication bandwidth shall be explored. Hybrid privacy mechanisms like those in [3] can be implemented in Tier Based FL and the performance with the current implementation.

References

[1] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," *CoRR*, vol. abs/2001.09249, 2020. [Online]. Available: https://arxiv.org/abs/2001.09249

[2] A. Ibarrondo and A. Viand, "Pyfhel: Python for homomorphic encryption libraries," in *Proceedings of the 9th on Workshop on Encrypted Computing amp; Applied Homomorphic Cryptography*, ser. WAHC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 11–16. [Online]. Available: https://doi.org/10.1145/3474366.3486923

[3] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," 2019.

[4] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A high-performance and communication-efficient federated learning system with asynchronous tiers," 2021.