The following publication H. Zheng, H. Hu and Z. Han, "Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning?," in IEEE Intelligent Systems, vol. 35, no. 4, pp. 5-14, 1 July-Aug. 2020 is available at https://doi.org/10.1109/MIS.2020.3010335.

Preserving User Privacy For Machine Learning: Local Differential Privacy or Federated Machine Learning?

Huadi Zheng, Haibo Hu, *Senior Member, IEEE*, Ziyang Han

Abstract—The growing number of mobile and IoT devices has nourished many intelligent applications. In order to produce high-quality machine learning models, they constantly access and collect rich personal data such as photos, browsing history and text messages. However, direct access to personal data has raised increasing public concerns about privacy risks and security breaches. To address these concerns, there are two emerging solutions to privacy-preserving machine learning, namely local differential privacy and federated machine learning. The former is a distributed data collection strategy where each client perturbs data locally before submitting to the server, whereas the latter is a distributed machine learning strategy to train models on mobile devices locally and merge their output (e.g., parameter updates of a model) through a control protocol. In this paper, we conduct a comparative study on the efficiency and privacy of both solutions. Our results show that in a standard population and domain setting, both can achieve an optimal misclassification rate lower than 20% and federated machine learning generally performs better at the cost of higher client CPU usage. Nonetheless, local differential privacy can benefit more from a larger client population ($>$ 1k). As for privacy guarantee, local differential privacy also has flexible control over the data leakage.

Index Terms—Federated Machine Learning, Local Differential Privacy

1 Introduction

The pervasive application of mobile, wearable and IoT devices has encouraged the boosting amount of generated data. Together with the drastic development of machine learning, mobile apps are empowered to provide personalized and self-evolving AI service such as voice assistant, word suggestion, facial recognition, and smart video feeds. In most of these applications, the machine learning model is refined by continually feeding in new user data (as features) and their feedback (as labels) from their mobile devices. However, these data, such as type history, web access logs, and frequently visited locations, are often sensitive and private information. To combat privacy infringement, US Federal Trade Commission has called for a national law against general violation of privacy after the testimony of Facebook–Cambridge Analytica scandal [1]; and EU has adopted the more stringent "General Data Protection Regulation" (GDPR) to supersede the "Data Protection Directive" in May 2018 [2].

Despite of strict legislation on personal data protection and the efforts made by most service providers, hosting personal data in a centralized location can still be highly risky due to security breach, internal theft or corporate dishonesty. A famous incident is the leakage of celebrity photos from iCloud in 2014. Unfortunately, centralized sanitation (e.g., generalization) and encryption schemes are shown vulnerable to various attacks, such as deanonymizing Netflix challenge dataset with IMDb data [3].

More recently, two distributed data analytical tools are proposed to protect privacy, namely, local differential privacy [4] and federated machine learning [5]. Both tools avoid direct access of personal data while still retaining high utility, e.g., high accuracy on statistics estimation or the trained model. Their mechanisms are summarized as follows:

1) Local differential privacy (LDP): Each user perturbs her data locally before sending them to an untrusted service provider for data collection and analytics. LDP achieves plausible deniability of each individual under a measurable and rigorous mechanism. LDP is heavily investigated in the literature of privacy-preserving statistics collection.
2) Federated machine learning (FML): It trains a globally shared model over a large number of distributed clients using an efficient control protocol with the central server. Only model parameter updates calculated on local data are submitted to the server, who aggregates them to improve the shared global model. This approach not only protects users' local data but also leverages on the computing resources on mobile devices.

Although both tools avoid direct access, their methodologies are essentially different. LDP is a theoretical privacy notation that can be achieved by different algorithms, while FL is a generic distributed learning framework without theoretical provable privacy. To conduct a comparative study of both tools, we deploy them to solve a common set of classification problems in mobile scenarios. This allows us

• *The authors are with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong and PolyU Shenzhen Research Institute (Corresponding author: Haibo Hu). E-mail: {huadi.zheng@connect.polyu.hk, haibo.hu@polyu.edu.hk, ziyang.han@connect.polyu.hk }*

to gain important insights of their performance in terms of classification performance, privacy loss, CPU/power consumption, and bandwidth consumption. In particular, to unify the privacy model of both solutions, we design a privacy loss metric through a general sample inference attack. To summarize, our primary contributions are as follows:

- We implement two competing solutions that learn from user data without submitting the original user data to the server and extensively discuss the unification of two solutions.
- We design a unified privacy loss metric for both solutions through a general sample inference attack.
- We conduct extensive experiments to compare both solutions in a set of machine learning problems in mobile scenarios.

The rest of the work is organized as follows. In Section 2, we introduce the fundamental principles of LDP and federated machine learning, and point out their problems. Section 3 presents the methodology of our comparative study of the two techniques. The experimental results shown in Section 4 compare their performance for given learning tasks with respect to various model and dataset parameters. We discuss a unification strategy in Section 5 and related work in Section 6. And finally, the findings of the study are concluded in Section 7.

2 PRELIMINARIES

2.1 Local Differential Privacy

LDP [4] extends the notion of differential privacy by perturbing local data with noise determined by a predefined parameter. In a nutshell, a perturbation algorithm $\mathcal{A}$ probabilistically modifies a local raw value ν_i to another value in the same domain of possible outputs κ. The modified value is then submitted to the server. A learning task on the statistical features (e.g., frequency and mean) of such data retains certain accuracy after the server collects all perturbed values. Meanwhile, each individual can have plausible privacy guarantee bounded on a privacy budget of ϵ.

Formally, the perturbation algorithm suffices ϵ-LDP principle if and only if for any two individuals' inputs ν_i and ν_j, we have

$$Pr[\mathcal{A}(\nu_i) = s] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(\nu_j) = s],$$

where $s \in \kappa$. Obviously, perturbed data is closer to the original data with a larger privacy budget ϵ and user population. Since the noises are applied to the data set directly, this strategy may have a strong impact on model performance when the budget is low.

2.2 Federated Machine Learning

In a task of federated machine learning, each mobile device initializes its own training using the shared model downloaded from the server and builds a new model using its local data. The updated model parameters will then be returned to the server, averaged with other peer devices and merged as the new shared model. This process is repeated multiple rounds to satisfy a learning objective until the desired set of model parameters are obtained.

Formally, a typical supervised machine learning objective function can be expressed as

$$\arg\min_{W} \frac{1}{N} \sum_{j \in J} \mathcal{L}(f(x_j, W), y_j),$$

where a learning algorithm is stated as f and its corresponding parameters W are estimated from the dataset J with a total sample size of N by minimizing the loss $\mathcal{L}$ between predictions on all input x_i and true label y_i in the training set.

In federated machine learning, data are assumed to be distributed over a set of M mobile devices and each of them can be considered as a partition P with $n = |P|$ training samples. The objective in this setting evolves to minimize the aggregated loss:

$$g(W) = \sum_{m \in M} \frac{n_m}{N} F(P_m, W),$$

where F is the local loss defined by

$$F(P_m, W) = \frac{1}{n_m} \sum_{k \in P_m} \mathcal{L}(f(x_k, W), y_k).$$

To train such an objective, a straightforward gradient descent algorithm can be applied to estimate model parameters using the iterative rule below:

$$W_{t+1} \leftarrow W_t - \eta \nabla g(W),$$

which is a full-batch gradient descent using all client data to generate an update in round t. However, this is not practical since it takes a long time for each iteration and even multiple times longer under the case of potentially high latency and limited bandwidth of the mobile network. To improve communication efficiency, federated machine learning commonly increases individual client computation by asking each mobile device to iterate over local data several times with stochastic gradient descent before submitting the parameter updates to the server for averaging [5].

3 METHODOLOGY

3.1 Problem Statement

We aim to tackle a machine learning problem in a distributed data setting where companies such as Google and Apple would like to improve their AI service accuracy, such as word auto-complete suggestion, through the data (e.g., keyboard input) from millions of distributed data points. To minimize the risk of privacy leakage, these companies adopt either of the two strategies: local differential privacy to allow users to perturb data before submitting to them or federated learning to train the machine learning model locally and only update the model parameters to them. Table. 1 summarizes the main characteristics of both strategies. A typical data record for classification task is in the form of $\{X_1, X_2, ..., X_l\}$ where X_i $(i < l)$ are feature dimensions and the last one X_l is the classification label of this record.

TABLE 1
Characteristics of LDP and FML

	Local Differential Privacy	Federated Machine Learning
Target	Data Collection	Distributed Learning
Computation	Mobile Perturbation, Server Training	Mobile Training, Server Aggregation
Application	Shared Model	Personal/Shared Model
Privacy Preservation	Adaptive Privacy Budget	Model Updates Only
Communication	One-time Submission	Multiple Interactions
Frequent Data Type	Structured Data	Text, Image, Audio

3.2 Strategy *LDP*: Submit Perturbed Data with ϵ-LDP

3.2.1 Client Side:

To perturb each user's data while satisfying ϵ-LDP, a sanitized mechanism is introduced which covers sensitive information with a certain amount of noises. For categorical attributes, each of the attributes has k_i $(1 \leq i \leq l)$ candidate values across all samples. For any dimension X_i, the perturbed output can be X_i' by using a staircase mechanism proposed by [6], namely k-RR:

$$P(X_i'|X_i) = \frac{1}{k_i - 1 + e^{\epsilon}} \begin{cases} e^{\epsilon} & \text{if } X_i' = X_i \\ 1 & \text{if } X_i' \neq X_i \end{cases}$$

where there will be a probability of $\frac{e^{\epsilon}}{k-1+e^{\epsilon}}$ to output the real value, and $\frac{1}{k-1+e^{\epsilon}}$ to output one of the remaining $k-1$ candidate values.

As for numeric attributes normalized in [-1,1], a piecewise mechanism by [7] can be applied as follows:

$$P(X_i'|X_i) = \frac{1}{e^{\epsilon/2}+1} \begin{cases} e^{\epsilon/2} & \text{if } X_i' \in [L_i, R_i] \\ 1 & \text{if } X_i' \in [-\delta, L_i) \cup (R_i, \delta] \end{cases}$$
$$\delta = \frac{\exp(\epsilon/2)+1}{\exp(\epsilon/2)-1}$$
$$L_i = \frac{\delta+1}{2} \cdot X_i - \frac{\delta-1}{2}$$
$$R_i = L_i + \delta - 1$$

where there will be a probability of $\frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$ to output a value sampled in $[L_i, R_i]$, and $\frac{1}{e^{\epsilon/2}+1}$ to output one in $[-\delta, L_i) \cup (R_i, \delta]$. After perturbation, the sanitized data $\{X_1', X_2', ..., X_l'\}$ will be submitted to the server when a high-speed network is available such as Wi-Fi.

3.2.2 Server Side:

The server receives a set of perturbed data from clients and concatenates them into one large dataset. Different from the statistics collection which usually has a calibration, the sanitized data won't have such a post-processing step since we aim at generating a perturbed dataset. All data points will be checked for any invalid or erroneous values produced by the client side. Features can be further extracted and put into learning pipeline to train a new model. To make use of the current model, its parameters will be used to initialize the new model.

3.3 Strategy *FML*: Train Locally with Federated Machine Learning

3.3.1 Client Side:

The client receives an instruction for model update task with a set of training parameters like local batch size and the number of training passes. The current service model with weights W will be downloaded into this device. Local data will be formulated into a proper input form and put into the training pipeline. In the current round t, the client m may iterate through the local data E passes with learning rate η before uploading results using the following gradient descent,

$$W_{m_t} \leftarrow W_{m_t} - \eta \nabla F(P_m, W),$$

where P_m is local data $\{X_1, X_2, ..., X_l\}_m$ used in one iteration but this can be controlled by the server to avoid using the whole local dataset in one batch.

3.3.2 Server Side:

The server sends out an invitation to a fraction C of current online devices M at each round of training and starts sending service model to $C \cdot M$ devices after confirmation. Updates received from the selected clients will be merged, which is equivalent to:

$$W_{t+1} \leftarrow \sum \frac{n_m}{N} W_{m_t}$$

This server-client interaction will be repeated for multiple times until the changes of parameters meet the pre-defined threshold.

4 Evaluation

4.1 Setup

We evaluated three public datasets in this comparative study.

- *NYC Taxi* [8]: This dataset contains $1.4m$ samples of 2016 yellow taxi trips in New York City. Based on the 8 attributes (e.g., number of customers, starting location of a trip), a model is trained to predict the duration of each trip.
- *BR2000* [9]: This dataset has $38k$ samples of census data collected in 2000 Brazil demographic census. Based on the 13 attributes (e.g., household, disability), a model is trained to predict a person's monthly income.
- *Adult* [10]: This dataset consists of $45k$ samples of census data from UCI Machine Learning Repository. 14 attributes (e.g., education level, occupation) are provided to determine whether a person earns over $50k$ a year.

All datasets contain categorical and continuous attributes. To ensure the data are applicable to *LDP*, we perturbed the numberic attributes and categorical attributes using corresponding mechanisms. Missing values and outliers were removed. For both tasks, the machine learning model was a neural network with 2 hidden layers containing 30 units, followed by *relu* activation function. The output layer was a *softmax* activation to produce classification results. Both strategies were given an initial model trained by 10% of

data. The remaining 70% of data were distributed to clients for local training and 20% were used for testing. Similarly, *LDP* only perturbed the 70% of the data and use the original 20% for testing.

All experiments are implemented with Python 3.6 on a desktop computer running Windows 10 with Intel Core i7-7700 3.6GHz CPU and 32G DDR4 RAM. Federated learning is simulated with TensorFlow r1.13. As the experiments require thousands of mobile devices to participate, which we do not own, we use multiple server machines and multithreading to simulate these devices. For privacy budget in *LDP*, we demonstrate the results of ϵ set to 2, 4 and 8, which are common budgets adopted by industries [11]. For LDP, the central model is trained with 500 iterations for maximum 100 epochs using a learning rate of 0.1. As for FML, by default, we pick 20% of clients in each round for maximum 200 rounds and iterate 20 local passes with learning rate 0.1 in each device before uploading the updates.

4.2 Classification Performance

To explore the performance of two strategies, we evaluated the misclassification rate with respect to the number of clients. The rate was reported when it converged during training or exceeded the maximum number of server epochs in *LDP* (resp. maximum communication rounds in *FML*).

As shown in Fig. 1, both strategies reduce misclassification rate with the change of client numbers from $0.1k$ to $1.6k$. The rate of *LDP* does not change much for the budget of 2 until it reaches around 1300 clients where the rate achieves the optimal 34% in *BR2000* dataset. The case with budget of 4 converges slightly quicker to a misclassification rate of 27% while the budget of 8 reaches the optimal performance of 15% in *Adult* dataset and eventually outperforms *FML* in most datasets. This is consistent with the perturbation mechanism where relaxed privacy guarantee, i.e., greater budget, leads to lighter noises. Most of the misclassification rates saturate after the client size exceed $1k$. It indicates that the model performance of *LDP* mainly benefits from an environment with a large scale of distributed data.

For *FML*, IID and non-IID setups were evaluated, that is, to distribute the data in a way where most labels evenly exist in each device or cluster in different devices. In both setups, misclassification rate decreases faster with more participants and stays at saturated level on 14% (IID) and 19% (non-IID), optimal in *BR2000* and *NYC Taxi* respectively when the number of clients reaches between 700 and 1000. *FML* can learn a useful model even if there are only a few clients at the early stage compared to *LDP*. It is obvious that the uneven distribution of data leads to a negative impact on *FML*, while *LDP* is free from the influence of data distribution since this strategy collects all data in the first place.

4.3 Privacy Loss

To understand the privacy loss of both strategies, inference accuracy is evaluated using general sample inference attacks. In this attack, we assume an adversary (e.g., untrusted aggregator) is able to decrypt the communication channel in both strategies and has basic knowledge about the types of the local training set (e.g., attribute type, candidate value).

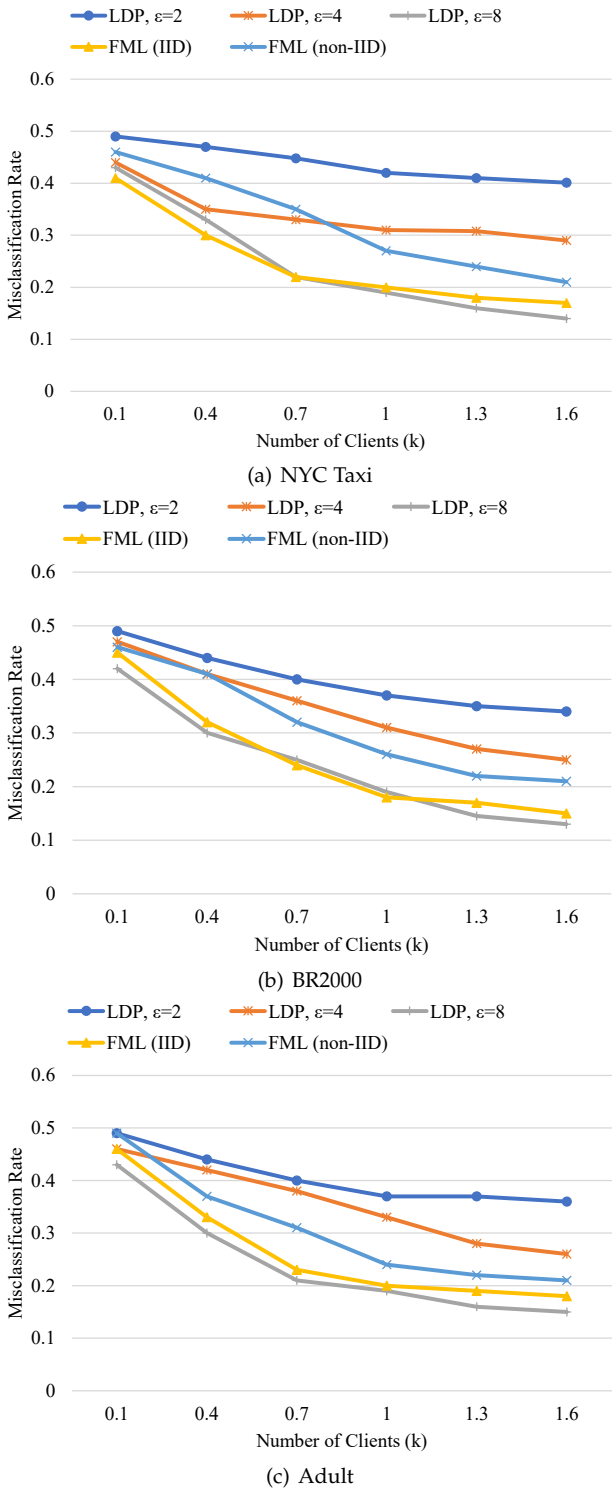
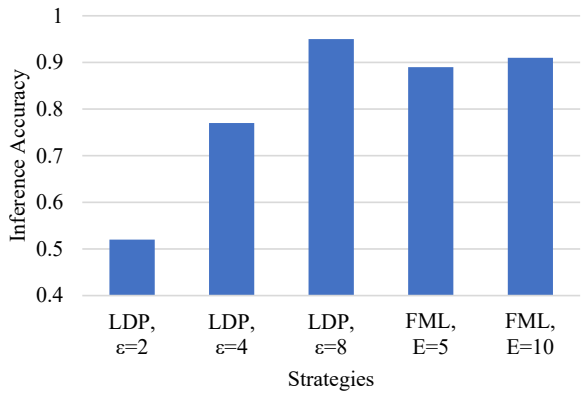


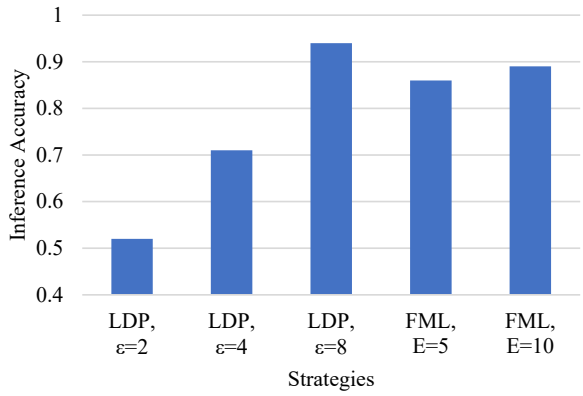
Fig. 1. Misclassification rate of different strategies

By observing the data transferred between a client and a server, i.e., perturbed data in *LDP* and model parameters in *FML*, the adversary can perform inference attack to determine which samples drawn from the same distribution belong to the client training set. A higher inference accuracy leads to greater privacy loss.

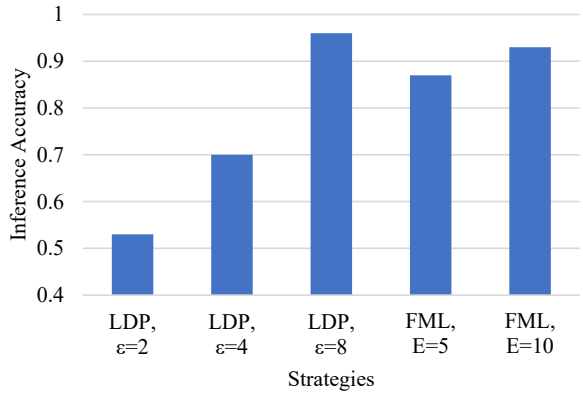
In *LDP*, the inference is performed by measuring the Manhattan distance between testing data and perturbed data, and a testing record is considered as a member of



(a) NYC Taxi



(b) BR2000



(c) Adult

Fig. 2. Privacy of different strategies (E: number of local passes in FML)

local dataset when its minimum distance is less than a threshold. As for *FML*, since adversary can obtain both global model and local updated one, by comparing the membership inference [12] on the two models, the local samples can be exposed. That is, given a threshold, if a record is recognized as a member in the inference on the local model but non-member on the global model, it is likely that this record belongs to the local set. We evaluated all settings on a testing dataset with half of the samples used in local training and the other half outside of the device, such that the random guess is 0.5. All results were reported under optimal threshold in their settings.

As shown in Fig. 2, *LDP* achieves a flexible control over privacy loss compared to *FML*. Except for the budget of 8, the inference accuracy is constrained to less than 80% and even 55% as privacy budget drops to 2 in all datasets. As for *FML*, the inference accuracy can reach over 80% among all datasets with 5 local passes and even 90% in *NYC Taxi* and *Adult* when local passes increase to 10. to improve communication efficiency, it commonly adds more computation to clients by iterating local updates multiple times before the aggregation step. This indicates that such fine-grained updates can significantly capture the details of local data and are vulnerable to malicious inference. In this case, *LDP* with low budget has stronger privacy guarantee than *FML* while the performance of classifier is the trade-off by revisiting model misclassification rate.

4.4 CPU Consumption

4.4.1 Client Side

The main client CPU consumption is on perturbation of data for *LDP* while *FML* spends most of the time updating the global model with local data. We review the CPU time against the average local dataset size of each device in Fig. 3. *FML* consumes more CPU to iterate through the data and grows linearly to over $3.8ms$ for *NYC Taxi* (resp. $6.3ms$ for *BR2000*) while *LDP* grows significantly slower and only reaches $1.2ms$ for *NYC Taxi* (resp. $1.7ms$ for *BR2000*). When the size of local dataset is small, the time will approximate preparation time such as parameters initialization since the real processing time is too short. The battery will drain faster under the setting of *FML*.

4.4.2 Server Side

The computation resources of server are spent on pooling client's data and training the model in *LDP*. Obviously it consumes more CPU over server side to train the model compared to *FML* where the server only needs to coordinate clients and aggregates all received updates, since training workload is transferred to clients. For each model, 100 server training epochs take an average of $36s$ with 500 iterations in *LDP* while the aggregation and update process in *FML* take less than $1s$.

4.5 Communication Cost

4.5.1 Client Side

As for data transmission of client device, since *LDP* will collect all data, the transmission amount is constant to the size of local dataset while *FML* sends a number of parameters depending on model size. In Fig. 4, *LDP* has a larger communication cost than *FML* when communication rounds are less than 300. Eventually, *FML* has 3x more cost due to frequent exchange of model updates with server.

4.5.2 Server Side

On the server side, since the communication cost against round change just aggregates all client's, we instead investigate the transferred data size against the number of clients by fixing round number at 1.1k (best model performance) for each client. As shown in Fig. 5, *LDP* grows much faster with more participants than *FML* in communication cost since it is equivalent to collect the whole dataset combined from all clients. Due to the frequent interactions between clients and server, the accumulated transferred data grows

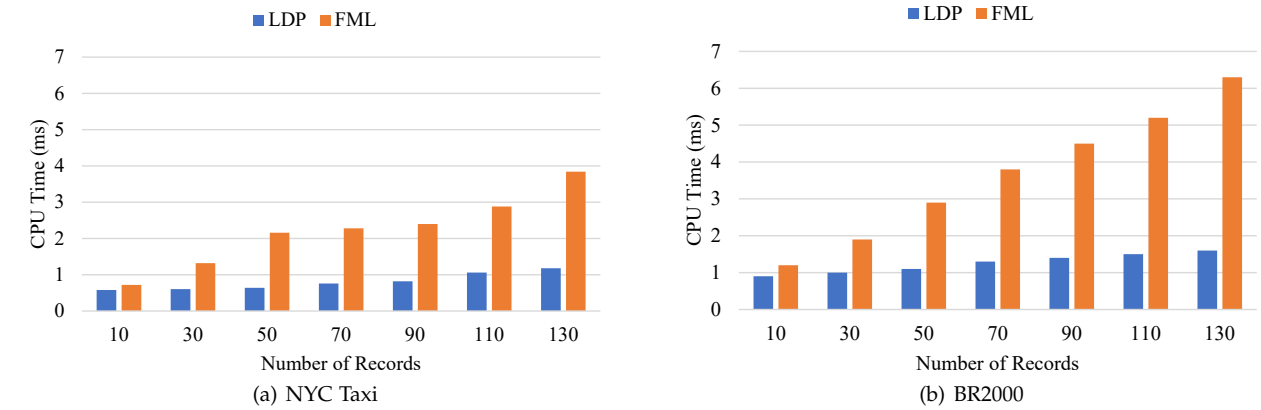


Fig. 3. Client CPU time

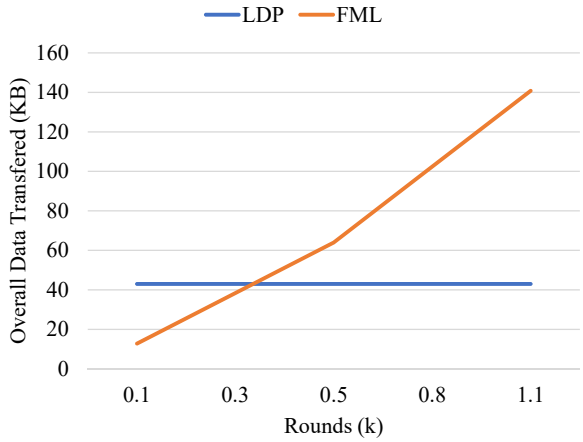


Fig. 4. Client bandwidth consumption, NYC Taxi

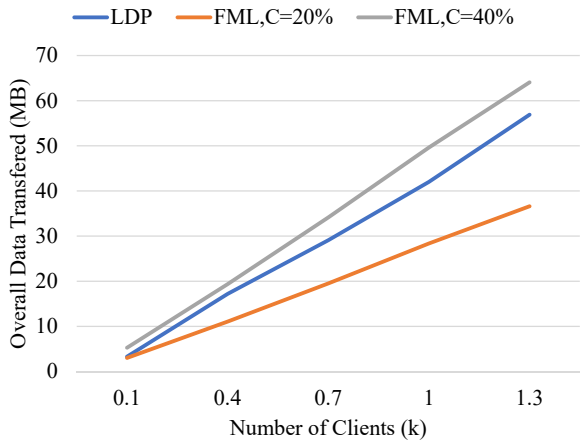


Fig. 5. Server bandwidth consumption, NYC Taxi

quickly as well and reaches over 30MB when 20% of clients participate in each round and can outgrow *LDP* with 40% participation rate.

5 Discussion

Impact of Data and Training Procedure In this comparative study, we evaluated moderate type of data for generality. For "heavy data" like images and audio, we expect the trend of computation/network overhead will be similar to current comparative study but with widening gap. On the one hand, client CPU usage in FML will grow drastically as the model complexity also increases for such data while LDP remains the same. On the other hand, LDP will consume higher network usage given that perturbed data has a similar size of the original one. For LDP, we adopt the same straightforward training as FML for fair comparison. However, the model performance may be volatile to the privacy budget. Alternative training procedure using frequency-based statistics [2] can be adopted to improve the model quality and stability. The main idea is to generate synopsis such as histogram from perturbed data and synthesize training data from that synopsis.

Privacy Challenges in FML Even though *FML* provides a good property of intrinsic preservation of local data while delivering high-quality model, this strategy still faces many challenges on privacy protection and the reasons are threefold. First, as shown in our empirical analysis, privacy control is limited for the submitted updates in *FML*, since the change of local pass number does not produce a significant influence over the privacy loss. Second, current *FML* heavily relies on encryption schemes to deliver secure aggregation and is susceptible to the inherited vulnerabilities of that designated encryption. Third, the system efficiency is liable to be degraded by the secure aggregation scheme, such as multi-party computation (MPC) [13] which is inherently computationally complex.

Unification of *FML* and *LDP*. Essentially, the aggregation step in *FML* is performing mean calculation on scattered data sources. Given that *LDP* has been frequently applied in such distributed analytical task [4], we can consider a unification approach that tackles the above challenges by integrating *FML* with *LDP*. The core idea is to inject ϵ-*LDP* perturbation to model updates before transmission.

Specifically, on the client side, a set of training instructions are provided as usual to perform local training. In addition to batch size and the number of training passes, the client is also notified of *LDP* perturbation mechanism and a privacy budget ϵ. After parameter update W is derived, instead of submitting it immediately, the client will generate a noisy version $W + ldp(\epsilon)$. On the server side, noisy updates received from the selected clients will be merged to canceled

the additive noises. This server-client interaction can repeat for multiple times with different budgets. If the perturbation is produced by a biased mechanism with non-zero mean, the server will further perform a calibration step on the aggregated result to obtain an accurate estimation.

In this way, the adversary can only recover noisy model updates even if the communication channel is intercepted. Besides, the level of perturbation can be flexibly negotiated on the fly. For example, if a participant finds the privacy budget unsatisfied, he/she can reject this round of training until the expectation is met. Furthermore, perturbation noise $ldp(\epsilon)$ is commonly generated with light computation, which can improve the overall efficiency compared to encryption scheme.

Emerging works have tried to leverage such unification but the designs are still limited to particular genres of models [14]. In some aspects, the unification approach can always outperform the two originals given that the perturbation is presented in intermediate values and keep a high resolution of original data. Nonetheless, we leave their empirical study for future work.

6 Related Work

LDP has been widely applied in distributed data collection, such as crowdsourcing scenario. It found main application in statistical analysis tasks, such as frequency estimation over categorical data. Erlingsson *et al.* proposed RAPPOR [15] for this task, which transforms a sensitive string into a Bloom filter and then applies the randomized response method [16] to perturb it. Marginal release has been studied in [17] under LDP, which is a potential alternative to produce synthetic data for machine learning task.

Learning models using distributed resources have been proposed for distributed GPU settings [18]. While they focus on a highly-controlled network inside a data center, Google proposes federated machine learning for a loose federation of multiple mobile clients with scalable design [5] and develops secure aggregation using encryption scheme such as multi-party computation [13]. As for the system aspects, the architecture of federated learning discussed in this paper is horizontal design [19], which enables easy unification with LDP on communication level [14]. Particularly, horizontal federated systems in mobile edge computing have started to study differentially private version by injecting noises to either SGD process or the final updates [20].

7 Conclusion

We investigate two promising data analytic strategies for distributed setting while preserving user privacy. Both strategies are adopted in the same real machine learning problems and evaluated with extensive experiments under various system settings. The results show that local differential privacy mainly benefits from a large user population and consumes less CPU/battery on mobile devices while maintaining a rigorous privacy guarantee. Federated machine learning can adapt itself quickly for a moderate number of users and produce a learning model with higher quality while the fine-grained update is vulnerable to inference. Nonetheless, the data submitted with local differential privacy can be reused indefinitely for other tasks such as marginal release or itemset mining, while the model trained by FL is specified for one type of prediction task. As for future work, we plan to evaluate different unified solutions again each other using similar empirical framework. We also plan to propose new privacy-preserving method based on the comparative study.

References

[1] U. F. T. Commission, "F.T.C. commissioners back privacy law to regulate tech companies," 2019. [Online]. Available: https://www.nytimes.com/2019/05/08/business/ftc-hearing-facebook.html

[2] E. Commission, "Data protection in the EU," 2018. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu

[3] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *S&P*, 2008, pp. 111–125.

[4] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[6] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," *Journal of Machine Learning Research*, vol. 17, pp. 17:1–17:51, 2016.

[7] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*, 2019, pp. 638–649.

[8] N. TLC, "Trip record data," 2016. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[9] IPUMS, "Harmonized international census data for social science and health research," 2018. [Online]. Available: https://international.ipums.org/international/index.shtml

[10] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[11] A. Privacy, "Our approach to privacy," 2019. [Online]. Available: https://www.apple.com/privacy/approach-to-privacy/

[12] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *NDSS*, 2019.

[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *CCS*, 2017, pp. 1175–1191.

[14] V. Pihur, A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, "Differentially-private "draw and discard" machine learning," *CoRR*, vol. abs/1807.04369, 2018.

[15] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS*. ACM, 2014, pp. 1054–1067.

[16] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[17] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "Calm: Consistent adaptive local marginal for marginal release under local differential privacy," in *CCS*, 2018, pp. 212–229.

[18] H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, and E. P. Xing, "Geeps: scalable deep learning on distributed gpus with a gpu-specialized parameter server," in *EuroSys*, 2016.

[19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[20] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No: U1636205, 61572413), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, C1008-16G and 15218919)

Huadi Zheng receives the BEng degree from the School of Data and Computer Science, Sun Yat-sen University, China, in 2012. Currently he is pursuing a PhD degree in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include mobile side-channel security, data privacy and machine learning.

Haibo Hu is an associate professor in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include cybersecurity, data privacy, internet of things, and machine learning. He has published over 80 research papers in refereed journals, international conferences, and book chapters. As principal investigator, he has received over 12 million HK dollars of external research grants from Hong Kong and mainland China. He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, VLDB Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.

Ziyang Han is a Ph.D. student in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His current research work include privacy-aware computing, information hiding, hardware-based security and security issues on databases. He has engaged in security component development of many widely used applications.