LDP-Fed: Federated Learning with Local Differential Privacy

Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, Wenqi Wei
Georgia Institute of Technology, Atlanta, GA 30332

ABSTRACT

This paper presents LDP-Fed, a novel federated learning system with a formal privacy guarantee using local differential privacy (LDP). Existing LDP protocols are developed primarily to ensure data privacy in the collection of single numerical or categorical values, such as click count in Web access logs. However, in federated learning model parameter updates are collected iteratively from each participant and consist of high dimensional, continuous values with high precision (10s of digits after the decimal point), making existing LDP protocols inapplicable. To address this challenge in LDP-Fed, we design and develop two novel approaches. First, LDP-Fed's LDP Module provides a formal differential privacy guarantee for the repeated collection of model training parameters in the federated training of large-scale neural networks over multiple individual participants' private datasets. Second, LDP-Fed implements a suite of selection and filtering techniques for perturbing and sharing select parameter updates with the parameter server. We validate our system deployed with a condensed LDP protocol in training deep neural networks on public data. We compare this version of LDP-Fed, coined CLDP-Fed, with other state-of-the-art approaches with respect to model accuracy, privacy preservation, and system capabilities.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; *Trust frameworks*; • **Computing methodologies → Learning settings**.

KEYWORDS

privacy-preserving machine learning, federated learning, local differential privacy, neural networks

ACM Reference Format:
Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, Wenqi Wei. 2020. LDP-Fed: Federated Learning with Local Differential Privacy. In *3rd International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20), April 27, 2020, Heraklion, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3378679.3394533

1 INTRODUCTION

Traditionally, machine learning (ML) algorithms have required that all relevant training data be held by a trusted central party. However, in the age of IoT, data is often generated and captured from distributed edge locations with different ownerships from multiple independent parties. Distributed systems were therefore developed for the distributed training of ML models through cluster nodes with shared data access or capabilities for data sharing with one or a few trusted central master node(s). However, when the edge nodes are owned by independent parties, there may not exist such a centralized point of trust. Furthermore, legal restrictions such as HIPAA [3], CCPA [15], or GDPR [18] and business competitiveness may further limit the sharing of sensitive data.

In response, federated learning (FL) has emerged as an attractive collaborative learning infrastructure. In a FL system, data owners (participants) do not need to share raw data with one another or rely on a single trusted entity for distributed training of ML models. Instead, participants collaborate to jointly train a ML model by executing local training algorithms on their own private local data and only sharing model parameters with the parameter server. This parameter server serves as a central aggregator to appropriately aggregate the local parameter updates and then share the aggregated updates with every participant. While FL allows participants to keep their raw data local, recent work has shown it is insufficient in protecting the privacy of the underlying training data from known inference attacks [16]. Model parameters exchanged during the training process [16] as well as outputs from the trained model [21, 25] remain as attack surfaces for privacy leakage.

Existing solutions to protect FL systems from such privacy attacks require trusted aggregators [17] or heavy cryptographic techniques [6, 23] which do not allow individual participants to define different local privacy guarantees, are insufficient for meaningfully protecting each high dimensional parameter vector against privacy leakage in the presence of high dimensional parameter vectors [6, 19], or have focused on low dimensional models [2, 26].

In this paper, we proposed LDP-Fed, a novel FL system for the joint training of deep neural network (DNN) models under the protection of the formal local differential privacy framework. LDP-Fed allows participants to efficiently train complex models such that each participant is formally protected from privacy inference attacks according to their own locally defined privacy setting. This paper makes two original contributions. First, we develop a federated training approach that can perform LDP-based perturbation on complex model parameter updates according to the local privacy budget while minimizing the overwhelming impact of noise on the joint ML training process. Second, we also present our parameter update sharing method for the selective sharing of model parameter updates at various rounds of the iterative LDP-Fed training process. We evaluate LDP-Fed against state-of-the-art privacy-preserving FL approaches in both accuracy and system features.

2 PRELIMINARIES

2.1 Deep Neural Network Training

Deep neural network (DNN) models are composed of many layers of basic building block nodes such as affine functions or simple

EdgeSys '20, April 27, 2020, Heraklion, Greece

ACM ISBN 978-1-4503-7132-2/20/04…$15.00
https://doi.org/10.1145/3378679.3394533

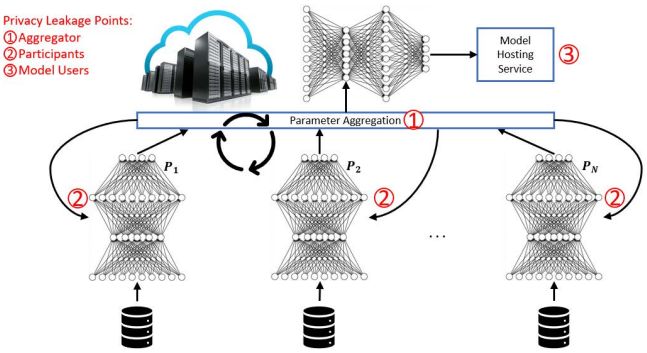

Figure 1: Privacy leakage in federated learning systems.

non-linear functions (e.g. sigmoids, rectified linear units (ReLU), etc.). A DNN model is therefore trained by fitting the parameters of these nodes to a known set of training inputs (provided to the first layer of nodes) and outputs (desired output from the last layer).

Specifically, a loss function $\mathcal{L}$ is quantifies the error between the desired outputs and the DNN generated output. Given a DNN with parameters θ, the loss $\mathcal{L}(\theta)$ of the DNN on the training set $\{x_1, x_2, \ldots, x_N\}$ is the average loss over the set, i.e. $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. DNN training therefore seeks the parameters θ which minimize this loss. While ideally training will result in the loss global minima, training in practice is rarely expected to reach this global value and instead finds an acceptably small loss point.

The process of minimizing the loss $\mathcal{L}$ is often done through applying the technique known as stochastic gradient descent (SGD) iteratively to subsets of the training data known as minibatches. At each step a batch B is selected and an estimation of the gradient $\nabla_\theta \mathcal{L}(\theta)$ is computed as $\mathbf{g}_B = \frac{1}{|B|}\sum_{x\in B} \nabla_\theta \mathcal{L}(\theta, x)$. The training algorithm then updates θ in the direction $-\mathbf{g}_B$ toward a local minima. Multiple systems are available to enable efficient training and evaluation of these DNNs models [1, 8, 13].

2.2 Federated Learning

As privacy concerns and legislation continues to mount, FL systems such as [5] have seen increased attention. FL systems remove the necessity of a central data location to train DNNs. Model parameters which minimize loss across multiple datasets are instead identified through model training that is done locally at the edge.

In a FL setting, N participants, each with independent datasets containing the same features and output classes, agree on a DNN model architecture. A central server (aggregator) then randomly initializes the model parameters θ_0 which are then distributed to each participant so that each may initialize their own copy of the model. At each round $r \in [0, E)$ of training, participants receive a copy of the aggregator's model parameters θ_r. Each participant P_i then conducts model training locally as described in Section 2.1 to generate updated parameters $\theta_{r+1,i}$ and uploads them to the aggregator. The aggregator then computes the average of value for each parameter and updates the global model with the parameters $\theta_{r+1} = \frac{1}{N}\sum_i \theta_{r+1,i}$. This process is continued either for a pre-determined number of rounds E or until the model converges.

While FL allows for private data to remain local to each participant, this data locality approach proves insufficient in protecting training data privacy as FL systems remain vulnerable to privacy inference attacks. Figure 1 highlights the multiple points of potential privacy leakage in federated learning. Information may leak to the central aggregator service (leakage point 1) as well as other participants (leakage point 2) by way of the shared parameter updates which are a type of encoding of each participant's private data. Recent work has indeed demonstrated that effective membership inference privacy attacks may be launched given access to these shared model updates [16]. Additionally, the final model itself will also leak with prediction outputs (leakage point 3) leading attackers to infer information about the underlying training data points [21, 25].

2.3 Local Differential Privacy

To combat inference attacks against shared data values, companies including Google, Apple, and Microsoft employ local differential privacy (LDP) [9, 11, 22], the state-of-the-art in privacy-preserving data collection. Rather than uploading raw data values, users in an LDP system perturb their data v using an algorithm Ψ and instead upload $\Psi(v)$. This perturbed value $\Psi(v)$ is then guaranteed to protect v from inference attacks according to a privacy parameter ϵ where a lower ϵ value indicates a higher level of privacy. This guarantee is formalized as follows.

Definition 2.1. (ϵ-LDP). A randomized algorithm Ψ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon > 0$, if and only if for any inputs v_1, v_2 in universe $\mathcal{U}$, we have:

$$\forall y \in Range(\Psi) : \frac{Pr[\Psi(v_1) = y]}{Pr[\Psi(v_2) = y]} \leq e^{\epsilon}$$

where Range(Ψ) is the set of all possible outputs of algorithm Ψ.

2.3.1 Condensed Local Differential Privacy. In [12], authors propose a specialization of LDP, Condensed Local Differential Privacy (CLDP). CLDP ensures privacy according to a privacy parameter α where, as with ϵ, a lower α value indicates a higher level of privacy. CLDP, however, also considers a distance metric d in its perturbation algorithm Φ. Specifically, let $\mathcal{U}$ denote the finite universe of possible values for user data v. Additionally, let $d : \mathcal{U}\times\mathcal{U} \rightarrow [0, \infty)$ be a distance function that measures the distance between any two items $v_1, v_2 \in \mathcal{U}$. CLDP is then formalized as follows.

Definition 2.2. (α-CLDP). A randomized algorithm Φ satisfies α-condensed local differential privacy (α-CLDP), where $\alpha > 0$, if and only if for any inputs $v_1, v_2 \in \mathcal{U}$:

$$\forall y \in Range(\Phi) : \frac{Pr[\Phi(v_1) = y]}{Pr[\Phi(v_2) = y]} \leq e^{\alpha \cdot d(v_1, v_2)}$$

where Range(Φ) is the set of all possible outputs of algorithm Φ.

While the definitions of LDP and CLDP are similar, their privacy parameters and indistinguishability properties vary as α-CLDP, indistinguishability is also controlled by the items' distance $d(\cdot, \cdot)$ in addition to α. Therefore, as d increases, α must decrease to compensate, making $\alpha \ll \epsilon$. Previous work [12] provides details for converting ϵ to α. To guarantee α-CLDP, the Exponential Mechanism (EM) is applied to a raw user value v end as follows.

Exponential Mechanism (EM). Let $v \in \mathcal{U}$ be the raw user data, and let the Exponential Mechanism Φ_{EM} take as input v and

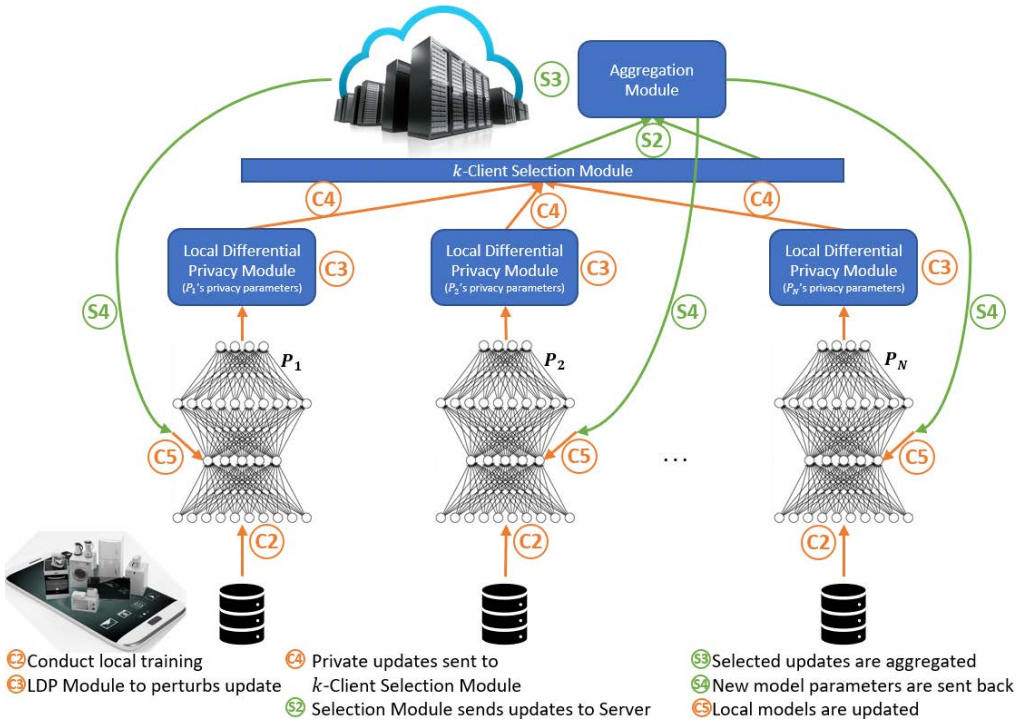

Figure 2: Private Federated Learning with LDP-Fed

output a perturbed value in $\mathcal{U}$, i.e., $\Phi_{EM} : \mathcal{U} \rightarrow \mathcal{U}$. Then, Φ_{EM} that produces output y with the following probability satisfies α-CLDP:

$$\forall y \in \mathcal{U} : Pr[\Phi_{EM}(v) = y] = \frac{e^{\frac{-\alpha \cdot d(v,y)}{2}}}{\sum_{z \in \mathcal{U}} e^{\frac{-\alpha \cdot d(v,z)}{2}}}$$

2.3.2 Privacy Accounting. In differentially private federated training of DNN models, an important issue arises in accounting for the multiple iterations of the training algorithm. That is, each participant and therefore each private dataset will be queried during multiple rounds of training. The composability of differential privacy allows for such iterations to be accounted for by accumulating the privacy cost at each round of the training as well as the multiple parameters included in each model update.

To account for the iterative nature of DNN training, the Sequential Composition theorem states that for functions $f_1, \ldots, f_n$ where f_i satisfies ϵ_i-DP for each $i \in [1, n]$, the release of the outputs $f_1(D), \ldots, f_n(D)$ satisfies $(\sum_{i=1}^{n} \epsilon_i)$-DP. The privacy amplification theorem [4, 14] additionally states that if random samples are selected rather than all available data, then each round satisfying ϵ-DP incurs only a cost of $(q\epsilon)$ against the privacy budget where $q = L/N$ is the sampling ratio.

3 FEDERATED LEARNING WITH LDP-FED

The LDP-Fed system coordinates the federated learning of a DNN with N participants (clients) and one parameter server. LDP-Fed integrates a LDP privacy guarantee into the general architecture of the FL algorithm as shown in Figure 2 to protect participants' data from inference attacks.

Specifically, consider N participants with the same dataset structure and learning task who wish to collaboratively train a DNN model in a federated fashion. That is, each participant wishes to perform local training on its own private data and only share parameter updates to the server. Additionally, participants wish to address FL privacy risks (Section 2.2) with an individualized LDP guarantee (Section 2.3). To accomplish these goals, we present the federated training process of our system LDP-Fed, from both client (participant) and server perspectives:

On the individual client side:

(1) Participants initialize local DNN instances with model parameters θ_0 and each local LDP Module is initialized with privacy parameters according to individual preferences.
(2) Each participant locally computes training gradients according to their private, local dataset.
(3) Each participant performs perturbation on their gradients according to their local instance of the LDP Module.
(4) Model parameter updates are anonymously sent to the k-Client Selection Module which uniformly at random accepts or rejects updates with probability $q = k/N$.
(5) Each participant waits to receive aggregated parameter updates from the parameter server. Upon receiving the aggregated updates, each participant updates its local DNN model, and proceeds to step 2 to start the next iteration.

On the parameter server side:

(1) The parameter server generates initial model parameters θ_0 and sends to each participant.
(2) The server waits to receive k parameter updates randomly selected by the k-Client Selection Module.
(3) Once parameter updates are received, the Aggregation Module aggregates the updates, i.e. averages the gradient updates to determine new model parameters.

(4) The parameter server updates model parameters and sends updated values back to participants to update local models.

The above steps iterate for both the N clients and the parameter sever until a pre-determined condition is reached such as reaching a maximum number of rounds (iterations) or a public test set no longer reporting improved performance (convergence). Compared with traditional FL systems, LDP-Fed introduces two new components: (1) the Local Differential Privacy Module running on each of the N clients and (2) the k-Client Selection Module.

Local Differential Privacy Module. For each client, the LDP Module takes as input the high dimensional vector of model parameter updates, say 29,034 distinct values, and outputs a vector containing the perturbed updates according to the participant's chosen privacy context. In the first prototype of LDP-Fed, we set the default privacy definition to be α-CLDP-Fed, a variation of α-CLDP. While the definition of α-CLDP in [12] is provided for LDP perturbation on single integer values in finite spaces, gradient values are instead real values with high precision (10s after decimal points). Therefore the α-CLDP-Fed Module introduces a precision parameter ρ and a clipping range parameter c such that each parameter update is converted to an integer in the range $[-c \cdot 10^{\rho}, c \cdot 10^{\rho}]$. By transforming the clipped parameters into integers according to the precision parameter ρ and clipping range parameter c, we can define the α-CLDP-Fed system with Ordinal-CLDP using EM from [12]. Larger c and ρ values will result in a larger universe space but allow for more specificity in the model update.

Another problem with applying α-CLDP from [12] to FL is that its protocol only accounts for single item uploads. In FL, LDP-Fed needs to iteratively upload a high dimensional parameter vector, which has typically 10,000 or more real valued parameters of high precision. Assume $k = N$ in the k-Client Selection Module, let E be the total number of iterations for a FL task, and let α be the total privacy budget. To guarantee α-CLDP, we must partition α into E small budgets, one for each of the E total iterations such that $\alpha = \sum_{i=0}^{E-1} \alpha_i$. Let θ_i be the total number of parameter updates to be uploaded to the parameter server at the ith iteration from any of the k selected clients, with α_i denoting the portion of the overall privacy budget α allocated to the ith iteration. To guarantee privacy in LDP-Fed, we therefore must set $\alpha_p = \frac{\alpha_i}{|\theta_i|}$ as the privacy budget when applying Ordinal-CLDP to each parameter update in θ_i.

k-Client Selection Module. Just as conventional FL systems do not require every participant to share their local training parameter updates in each round, training in LDP-Fed results in only k participants' parameter updates being uploaded to the parameter server for any given round with $k <= N$. As the discarded updates do not introduce any privacy cost, sampling amplification allows for a tighter bound of $\alpha = \sum_{i=0}^{E-1} q \cdot \alpha_i$ with $q = \frac{k}{N} <= 1$.

4 EXPERIMENTAL RESULTS

All experiments were conducted on an example FL system with $N = 50$ participants and the k-client Selection Module set to randomly select $k = 9$ updates at each round. The DNN model architecture used has two convolutional layers each followed by a batch normalization layer and a 2D max-pool layer. The final network layer is a single fully connected layer with 1,568 hidden units. We conduct 80 total rounds of training, i.e. E is set to 80. To evaluate

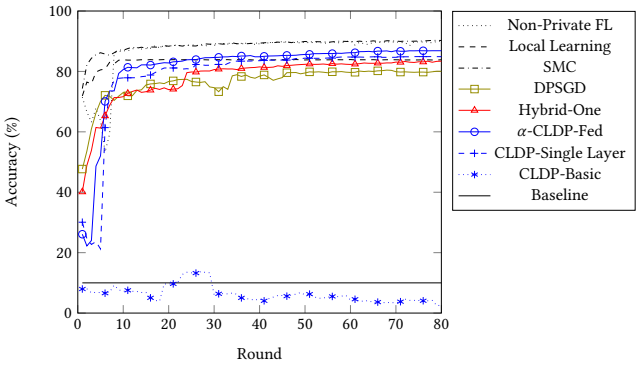

Figure 3: α-CLDP-Fed compared to other FL methods.

the effectiveness of LDP-Fed, we also implemented a number of related methods, such as Non-Private FL, Local Learning, and secure multiparty computation (SMC) methods for comparison and analysis. Related methods requiring ϵ values were set with the ϵ value equivalent to $\alpha = 1.0$ given the appropriate ρ and c settings according to the conversion approach provided in [12].

Non-Private FL. In non-private federated learning, the LDP Module is not activated and the k-Client Selection Module receives complete model parameter updates from participants in the clear. **Local Learning**. The results of local learning are reported as the average accuracy results received by the individual participants if they were to train the DNN model on their own local datasets without sharing parameter updates. **Baseline**. Random guess baseline of 10%. **SMC**. With SMC, the same process as Non-Private FL is followed except that model updates are encrypted when sent to the k-Client Selection Module and then decrypted only post-aggregation in the Aggregation Module. Here parameter updates again need to be integers and therefore only ρ =10 digits after the decimal are preserved. **Differentially Private Stochastic Gradient Descent** (DPSGD). Authors in [2] propose a centralized approach to differentially private deep learning wherein noise is added to each gradient by the optimizer. We compare the impact of using LDP-Fed with the impact of using such a differentially private optimizer on each participant. **SMC and DPSGD Hybrid** (Hybrid-One). Authors in [23] propose a FL system which leverages an optimizer similar to that in the DPSGD method. However, the hybrid approach leverages SMC to decrease the scale of noise required at each participant.

All experiments are carried out on the FashionMNIST dataset, consisting of 60,000 training examples and 10,000 testing examples [27]. Each example is a 28 ×28 size gray-level image depicting an item from one of ten different fashion classes.

4.1 Limited Updates with LDP-Fed

We first evaluate the effectiveness of LDP-Fed with α-CLDP-Fed, a version of LDP-Fed with α-CLDP in the LDP Module. The comparison study includes six existing federated learning scenarios and three FL settings using CLDP: CLDP-Basic, CLDP-Single Layer, and our recommended α-CLDP-Fed. All private methods have a total privacy budget equivalent to $\alpha = 1.0$.

Figure 3 reports the results. CLDP-Basic refers to a baseline implementation of α-CLDP wherein participants provide updates at each round for all parameters in the DNN. Therefore, the budget α in CLDP-Basic must be divided amongst all the 29,034 parameters.

# of Cycles	Accuracy	Std Deviation
1	86.85%	0.12
2	86.20%	0.61
4	86.89%	0.10
5	**86.93%**	**0.12**
10	86.30%	0.24
16	85.28%	0.11

Table 1: Impact of introducing cycle-based approach in α-CLDP-Fed. A minimum of c' = number of cycles rounds allocated to each layer.

As shown in Figure 3, the CLDP-Basic displays the worst accuracy, below the random guest baseline of 10%. This indicates that applying the privacy budget uniformly across all parameter updates can cause untenable loss of training accuracy for large, complex models. Instead, α-CLDP-Fed presents a novel and intelligent algorithm for local differential privacy budget allocation and perturbation at each iteration throughout a FL workflow. In α-CLDP-Fed, participants upload only a subset of the parameters at each round, resulting in a higher budget allocated to individual parameter uploads. We first describe CLDP-Single Layer.

In CLDP-Single Layer, rather than sending a complete set θ of parameter updates at every round, each of the selected k participants at round i only perturbs and shares $\theta_i \subset \theta$ with the parameter server, where θ_i contains parameter updates for only a single layer of the DNN. The budget allocated to each parameter can then be increased to $\alpha_p = \frac{\alpha_i}{|\theta_i|}$ where α_i is the budget allocated to round i. Figure 3 shows that the CLDP-Single Layer algorithm significantly outperforms the CLDP-Basic algorithm and results in a final accuracy of 84.89%. In CLDP-Single Layer, each layer is allocated an even number of rounds and each round an even slice of the budget. Specifically, given ℓ layers, the updates sent during the first $\frac{E}{\ell}$ rounds include only parameter updates for the parameters in the DNN output layer. During each subsequent $\frac{E}{\ell}$ set of rounds, updates are for parameters one layer backward in the network.

In contrast to CLDP-Single Layer, α-CLDP-Fed allocates the number of rounds proportionate to the percentage of the model's total parameters contained within that layer, i.e. for layer i, $E_i = \frac{|\theta_i|}{|\theta|}E$ total rounds are dedicated to updating parameters in layer i. A minimum of 1 round is reserved for each layer. The same backward stepping approach is used as in CLDP-Single Layer. In α-CLDP-Fed the budget is also allocated proportionate to layer size. Figure 3 shows that α-CLDP-Fed further improves the training accuracy of CLDP-Single Layer with the highest final accuracy among the privacy-preserving approaches with 86.85% accuracy. Furthermore, Figure 3 shows that both α-CLDP-Fed and CLDP-Single Layer outperform the non-private Local Learning, DPSGD, and even Hybrid methods.

4.2 Impact of LDP-Fed Perturbation Cycles

In LDP-Fed we further introduces cycles to control when different parameter updates are shared with the parameter server. Each cycle is implemented in terms of iteration rounds. That is, let c' = number of cycles. One cycle is then $\frac{E}{c'}$ rounds with each cycle being allocated $\frac{\alpha}{c'}$ of the privacy budget. Rounds and budget are then allocated within each cycle to individual layers according to the strategy in Section 4.1. This allows layers to be revisited for updates within the training process. We report the impact of varying the number of cycles in α-CLDP-Fed in Table 1. This set of experiments shows that setting the number of cycles to 5 will result in a high, stable accuracy of 86.93% averaged across runs with a standard deviation of 0.12. In LDP-Fed, the default cycle value is set to 5.

5 SYSTEM FEATURE COMPARISON

We have reported experimental comparison of our α-CLDP-Fed method for privacy preserving federated learning with several representative approaches. We additionally provide a system feature comparison in Table 2; highlighting the value-added feature benefits of using LDP-Fed. First, LDP-Fed system does not require heavy cryptographic protocols which may not be suitable for edge devices engaged in FL. Second, LDP-Fed allows individual participants to locally define their own privacy level through the LDP Module. This is a valuable feature as previous work [20, 24] has indicated that vulnerability to privacy attacks is not uniform and may be more acute for some participants' datasets, leading to a desire for a stricter privacy guarantee. Last, but not the least, LDP-Fed provides formal protection from known privacy inference attacks while demonstrating an ability to maintain good accuracy in the presence of large, complex models

6 RELATED WORK

The LDP-Fed system relates to both FL and privacy-preserving ML.

Federated Learning Approaches. In [28] authors propose a distribute data mining system with DP, but their results demonstrate a significant accuracy loss and the system requires a trusted aggregator to add the necessary noise. In [17], while several "teacher" models are independently trained, a trusted aggregator must provide a DP query interface to a "student" model with unlabelled public data. [6] introduces cryptographic protocols to protect individual updates from being seen prior to aggregation, but leaves the aggregate updates and final predictive model vulnerable to inference attacks. Additional protocols allow users to leverage such cryptographic techniques to decrease the scale of noise [7, 10, 23]. These approaches require expensive cryptographic operations and either remove the ability of individual participants to identify privacy levels locally or demonstrate higher accuracy loss.

Privacy-Preserving ML. [19] similarly presents a distributed learning system using DP without a central trusted party. However, the DP guarantee is per-parameter and becomes meaningless for models with a large number of parameters. [26] also proposes an LDP protocol for multidimensional continuous data, however their experiments entailed 4 million users and <20 features for training smaller dimensional models.

7 CONCLUSION

We have presented LDP-Fed, a novel federated learning approach with LDP. Our system allows participants to efficiently train complex models while providing formal privacy protection. The design of LDP-Fed has two unique features. First, it enables participants to customize their LDP privacy budget locally according to their own preferences. Second, LDP-Fed implements a novel privacy preserving collaborative training approach towards utility-aware privacy

Privacy-Preserving Federated Learning Method	Efficient	Locally Defined Privacy Guarantee	Protection from Inference Attacks	Handles Complex Models
SMC [6]	✗	✗	~	✓
ϵ-DP Parameter Sharing [19]	✓	✓	~	✓
Local Optimizer [2]	~	✓	✓	✗
Hybrid-One [23]	✗	✗	✓	~
Continuous ϵ-LDP [26]	✓	✓	✓	✗
LDP-Fed	✓	✓	✓	✓

Table 2: Comparison of methods for private federated model training.

perturbation to prevent uncontrolled noise from overwhelming the FL training algorithm in the presence of large, complex model parameter updates. The α-CLDP-Fed algorithm design also exhibits a successful formal development of extending the traditional LDP theory, intended for single categorical values, to our LDP-Fed algorithm capable of handling high dimensional, continuous, and large scale model parameter updates. We provide empirical and analytical comparison of LDP-Fed with the state-of-the-art privacy-preserving FL approaches in both accuracy and system features.

ACKNOWLEDGMENTS

This research is partially sponsored by NSF CISE SaTC 1564097. The first author acknowledges an IBM PhD Fellowship Award and the support from the Enterprise AI, Systems & Solutions division led by Sandeep Gopisetty at IBM Almaden Research Center. CLDP-Fed is developed on top of the Ordinal-CLDP protocol whose implementation is a part of our CLDP release, publicly available at https://github.com/git-disl/CLDP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[3] Accountability Act. 1996. Health insurance portability and accountability act of 1996. *Public law* 104 (1996), 191.

[4] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. 2014. Bounds on the sample complexity for private learning and private data release. *Machine learning* 94, 3 (2014), 401–437.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1175–1191.

[7] Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin E Lauter, and Peter Rindal. 2017. Private Collaborative Neural Network Learning. *IACR Cryptology ePrint Archive* 2017 (2017), 762.

[8] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS workshop*.

[9] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*. 3571–3580.

[10] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[12] Mehmet Emre Gursoy, Acar Tamersoy, Stacey Truex, Wenqi Wei, and Ling Liu. 2019. Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing* (2019).

[13] Roberto Ierusalimschy, Luiz Henrique De Figueiredo, and Waldemar Celes Filho. 1996. Lua—an extensible extension language. *Software: Practice and Experience* 26, 6 (1996), 635–652.

[14] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[15] KJ Mathews and CM Bowman. 2018. The California Consumer Privacy Act of 2018.

[16] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Stand-alone and Federated Learning under Passive and Active White-box Inference Attacks. In *Security and Privacy (SP), 2019 IEEE Symposium on*.

[17] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. *arXiv preprint arXiv:1802.08908* (2018).

[18] General Data Protection Regulation. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union (OJ)* 59, 1-88 (2016), 294.

[19] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[20] Reza Shokri, Martin Strobel, and Yair Zick. 2019. Privacy risks of explaining machine learning models. *arXiv preprint arXiv:1907.00164* (2019).

[21] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[22] Abhradeep Guha Thakurta, Andrew H Vyrros, Umesh S Vaishampayan, Gaurav Kapoor, Julien Freudinger, Vipul Ved Prakash, Arnaud Legendre, and Steven Duplinsky. 2017. Emoji frequency detection and deep link frequency. US Patent 9,705,908.

[23] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 1–11.

[24] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Wenqi Wei, and Lei Yu. 2019. Effects of Differential Privacy and Data Skewness on Membership Inference Vulnerability. *arXiv preprint arXiv:1911.09777* (2019).

[25] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2019. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing* (2019).

[26] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 638–649.

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:cs.LG/cs.LG/1708.07747

[28] Ning Zhang, Ming Li, and Wenjing Lou. 2011. Distributed data mining with differential privacy. In *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 1–5.