2025 10th International Conference on Machine Learning Technologies

Federated Learning With Individualized Privacy Through Client Sampling

1st Lucas Lange
ScaDS.AI Dresden/Leipzig
Leipzig University
Leipzig, Germany
lange@informatik.uni-leipzig.de

2nd Ole Borchardt
ScaDS.AI Dresden/Leipzig
Leipzig University
Leipzig, Germany
ob14dupe@studserv.uni-leipzig.de

3rd Erhard Rahm
ScaDS.AI Dresden/Leipzig
Leipzig University
Leipzig, Germany
rahm@informatik.uni-leipzig.de

***Abstract*—With growing concerns about user data collection, individualized privacy has emerged as a promising solution to balance protection and utility by accounting for diverse user privacy preferences. Instead of enforcing a uniform level of anonymization for all users, this approach allows individuals to choose privacy settings that align with their comfort levels. Building on this idea, we propose an adapted method for enabling Individualized Differential Privacy (IDP) in Federated Learning (FL) by handling clients according to their personal privacy preferences. By extending the SAMPLE algorithm from centralized settings to FL, we calculate client-specific sampling rates based on their heterogeneous privacy budgets and integrate them into a modified IDP-FedAvg algorithm. We test this method under realistic privacy distributions and multiple datasets. The experimental results demonstrate that our approach achieves clear improvements over uniform DP baselines, reducing the trade-off between privacy and utility. Compared to the alternative SCALE method in related work, which assigns differing noise scales to clients, our method performs notably better. However, challenges remain for complex tasks with non-i.i.d. data, primarily stemming from the constraints of the decentralized setting.**

***Index Terms*—differential privacy, federated learning, privacy-preserving machine learning, user privacy, personalization**

I. Introduction

With the adoption of Machine Learning (ML) in everyday life and the increasing demand for representative training data, concerns about data security and privacy for contributing individuals have become more significant [1]–[4]. In this context, decentralized training algorithms, such as Federated Learning (FL), which ensure that data remains on devices (e.g., smartphones), and provable privacy guarantees, such as Differential Privacy (DP), are gaining prominence [5]–[9].

FL addresses the issue of centralizing training data but does not inherently guarantee data privacy, as attacks on model parameters remain possible [10]. To mitigate this, proposed modifications provide DP guarantees by introducing noise into clients' model updates [11]. However, DP introduces a trade-off between privacy and model utility, where optimizing this trade-off is critical for usability [12]. Since individuals generate data with varying sensitivity levels and privacy requirements, Individualized Differential Privacy (IDP) has emerged as a promising approach, allowing users to select privacy levels (e.g., low, medium, or high) [13]. This enables models to learn more effectively from data with lower privacy requirements, rather than applying the strictest privacy guarantees uniformly.

Existing IDP algorithms in FL focus on local DP [14], impose additional restrictions [15], or adjust noise multipliers [16]. However, such approaches have shown suboptimal results in centralized DP training [17]. Instead, findings suggest that individualized sampling rates, which alter the probability of including data in a training step, offer better performance.

We propose an approach to leverage individualized client sampling rates in FL, enabling personalized privacy guarantees for clients. To transfer the sampling method from centralized to federated settings, we introduce an updated training algorithm that determines client participation based on their privacy requirements. Additionally, we address challenges such as the realistic distribution of training data and privacy budgets across clients in our experiments. Our results show that individual guarantees, applied through our sampling techniques improve the privacy-utility trade-off of standard DP and also outperform the alternative technique of noise multiplier scaling [16].

Section II provides an overview of the fundamentals, while Section III reviews related work. Section IV introduces our algorithm, and Sections V and VI present the experimental setup and results, respectively. Finally, we discuss and summarize our findings in Sections VII and VIII.

II. Background

A. Federated Learning

Federated Learning (FL) [18] is a decentralized ML approach that enables multiple participants, referred to as clients, to collaboratively train a shared model without transferring their local data to a central server. Instead, each client performs training locally, and only model updates, such as gradients or parameters, are shared for aggregation. The most common approach is `FedAvg` [18] that averages across client gradients to update the global model. This process ensures that raw data remains on device, addressing privacy concerns and reducing the risks associated with data centralization [5].

B. Differential Privacy

Differential Privacy (DP) [19] is a mathematical framework to provide formal privacy guarantees when analyzing or sharing data. It ensures that the inclusion or exclusion of a single

979-8-3315-3671-8/25/$31.00 ©2025 IEEE

individual's data in a dataset has a limited impact on the output of an algorithm, thereby protecting individual privacy. A mechanism M satisfies (ε, δ)-DP if, for datasets D_1 and D_2 differing in at most one element, and for all output subsets S:

$$\Pr[M(D_1) \in S] \leq e^{\varepsilon} \cdot \Pr[M(D_2) \in S] + \delta,$$

where the privacy loss ε controls the strength of the privacy guarantee, with smaller values indicating stronger privacy. Parameter δ accounts for the probability of privacy failing.

C. Differentially Private Stochastic Gradient Descent

In DP-compliant algorithms, the core idea is to introduce controlled noise into the computation to obscure the contribution of single data points at the cost of overall utility. Differentially Private Stochastic Gradient Descent (DP-SGD) [20] is an optimizer adaptation that ensures that ML training satisfies DP. It works by first clipping individual gradients to a fixed norm, which limits their influence on the model updates. Then, random noise is added to the aggregated gradients to obscure contributions before finally updating the model parameters.

III. Related Work

This section reviews the state-of-the-art through relevant previous work in the context of FL with DP and IDP.

In general, IDP allows for varying privacy budgets across users, which may improve utility due to not applying the strictest setting to everyone, while still respecting user-specific privacy requirements. For centralized settings, Boenisch et al. [17] propose two methods, `SAMPLE` [21] and `SCALE` [22], where `SAMPLE` adjusts sampling probabilities for data points, while `SCALE` focuses on varying the added noise per data point. In their evaluation, they show that `SAMPLE` slightly outperforms on differing privacy budget distributions for their user group simulations, which are inspired by earlier studies on user behavior [23], [24]. Both studies underscore the gap between users' stated privacy concerns and their actual behaviors, highlighting the need for accessible mechanisms.

While FL as a decentralized learning setup inherently reduces privacy risks by avoiding direct data sharing, it remains vulnerable to attacks such as membership inference and reconstruction attacks [10]. DP provides a robust defense against such threats and McMahan et al. [11] thus extend the aggregation of client weights with a private `DP-FedAvg` variant by clipping client gradients and adding Gaussian noise.

Shifting FL to IDP mainly shifts the view from data points to a client-level privacy perspective. As in central settings, IDP enables better model utility through allowing user-specific privacy requirements. [14] and [15] propose methods for local DP with individualized budgets, which leads to noising gradients already outside of aggregation and limits potential performance. In [15] they additionally rely on limiting the ε-DP guarantee to a range τ. [25] focuses on reducing communication costs by projecting private client updates through gradient projection. They combine this notion with standard DP-SGD but halt training for clients once their privacy budgets are exhausted. Aldaghri et al. [16] implement individualized noise multipliers for clients, which is evaluated at two privacy levels and translates the `SCALE` method of [17] to FL.

Our method should be free of additional restrictions from local DP and thus favors a global approach with central aggregation steps for orchestration, different from [14] and [15]. Our closest related work in FL [16], already implemented a comparable solution to the `SCALE` method from the central setting. However, [17] found that `SAMPLE` performed on par or better in their central evaluation. We thus elevate the `SAMPLE` approach from the central to the FL scenario, and thereby from data point-level privacy to client-level privacy. For this we revise the `DP-FedAvg` algorithm to an `IDP-FedAvg` variant. We further extend the evaluation from [16] through more realistic privacy distributions from user studies [23], [24].

IV. Client Sampling Algorithm

In this section, we develop the `IDP-FedAvg` algorithm to train ML models with heterogeneous privacy guarantees in FL. The algorithm implements IDP through customized sampling rates within `DP-FedAvg`, inspired by the non-FL methods in [17], [21]. The main algorithm consists of two steps:

1) **Privacy Step:** Compute the noise multiplier and individualized sampling rates for each client, as described in Algorithm 1. Sampling rates are derived from client-specific privacy budgets using the `SAMPLE` algorithm by [17]. We adapt this procedure by mapping epochs to FL rounds and training samples to clients, transitioning guarantees from data point-level to client-level privacy.
2) **Training Step:** In Algorithm 2, training is performed using `DP-FedAvg` aggregation with adaptive clipping [26]. However, clients are sampled based on their individualized sampling rates before a global noise multiplier is applied uniformly across all sampled clients.

In [17], the authors demonstrate that training with individualized sampling rates derived from a user's specified privacy budget may achieve better results than individual scaling of noise multipliers or clipping norms. Their algorithm for the central training scenario samples data points with differing probabilities while maintaining the expected value per epoch and applying a constant noise multiplier.

This implementation can be directly adapted to calculate sampling rates for clients, as shown in Algorithm 1. Clients are grouped into P groups according to their respective privacy budgets, where $|P|$ is the number of unique values. The function `GetGroupSamplingRates` outputs a uniform noise multiplier σ_{SAMPLE} and individualized sampling rates $\{q_1, \ldots, q_P\}$, which describe the probability of selecting each client for a training round. To determine these values, the algorithm starts with an initial σ_{SAMPLE} and calculates the corresponding intermediate $\{q_1, \ldots, q_P\}$ needed to satisfy each client's ε using this multiplier. As a condition, the expected sampling rate q is based on the number of clients participating in training each round. The initial multiplier σ corresponds to the strictest privacy budget ε_1. If the resulting average sampling rate across clients does not satisfy q, the noise multiplier is too large for the given privacy budget

Algorithm 1: `GetGroupSamplingRates`: calculating per-group sampling rates regarding privacy budgets, as presented in Algorithm 2 [17, p.6] but adapted to clients instead of data points. Subroutine `getSampleRate` translates DP noise multipliers to sampling rates as in Algorithm 4 [17, p.17].

Input: Per-group target privacy budgets $\{\varepsilon_1, \ldots, \varepsilon_P\}$, target δ, iterations I, sampled clients per round c, total number of clients N, per-privacy group clients $\{|\mathcal{G}_1|, \ldots, |\mathcal{G}_P|\}$

Output: Sampling noise multiplier σ_{SAMPLE}, sampling rates $\{q_1, \ldots, q_P\}$

$q \leftarrow \frac{c}{N}$;
$\sigma_{\text{SAMPLE}} \leftarrow$`getNoise` $(\varepsilon_1, \delta, q, I)$;
foreach $p \in [P]$ **do**
  $q_p \leftarrow$ `getSampleRate` $(\varepsilon_p, \delta, \sigma_{SAMPLE}, I)$;
end
while $q \not\approx \frac{1}{N}\sum_{p=1}^{P} |\mathcal{G}_p| q_p$ **do**
  scaling factor $s_i \lessapprox 1$: $\sigma_{\text{SAMPLE}} \leftarrow s_i \sigma_{\text{SAMPLE}}$;
  foreach $p \in [P]$ **do**
    $q_p \leftarrow$ `getSampleRate` $(\varepsilon_p, \delta, \sigma_{SAMPLE}, I)$;
  end
end
return $\sigma_{\text{SAMPLE}}, \{q_1, \ldots, q_P\}$;

distribution. The noise multiplier is then iteratively reduced using a scaling factor $s_i \lessapprox 1$ (slightly smaller) until the resulting rates comply. `getSampleRate` originally uses a modified PyTorch Opacus function that ensures that privacy budgets are exhausted after I iterations of sampling. We translate that to the dp-accounting library for our Tensorflow version. [17] prove that their sampling algorithm satisfies $(\{\varepsilon_1, \ldots, \varepsilon_P\}, \delta)$-DP, extending their DP guarantees to groups of data points with heterogeneous sampling rates at constant noise.

The training in Algorithm 2 uses a modified version of standard `DP-FedAvg` with adaptive clipping by [26]. But instead of uniformly sampling the client subset S for each training round, the calculated sampling rate for each client from Algorithm 1 is used. Each client thereby acquires only the noise needed for their respective individual privacy guarantee. We focus on global aggregation and do not include hyperparameters e.g. for client training in this representation.

V. Experimental Setup

In this section, we detail how we conduct our experiments.

A. Implementation Environment

Reference code is available from our repository at https://github.com/luckyos-code/flidp. We use Python 3.10 with Tensorflow for creating ML models. The Tensorflow Federated library supports our simulations regarding FL, while our DP implementations use Tensorflow Privacy. For hardware we run our experiments on a cluster using an NVIDIA RTX 2080 Ti GPU, 64GB of memory, and an AMD EPYC 7551P CPU.

Algorithm 2: `IDP-FedAvg`: implements `FedAvg` with individualized DP, incorporating client sampling rates into `DP-FedAvg` with adaptive clipping [26].

Input: Per-client target privacy budgets $\{\varepsilon_1, ..., \varepsilon_N\}$, target δ, rounds I_{Fed}, clients $\mathcal{C}$ with $|\mathcal{C}| = N$, sampled clients per round c, client learning rate $\eta_{\mathcal{C}}$, server learning rate η_s, clipping quantile γ, clipping learning rate η_C, global model θ

Output: Updated model θ', clipping quantile γ'

Groups: $\mathcal{E}_{\mathcal{G}} \leftarrow$ `Unique` $(\{\varepsilon_1, ..., \varepsilon_N\})$;
$\sigma_{\text{SAMPLE}}, Q_{\mathcal{G}} \leftarrow$ `GetGroupSamplingRates` $(\mathcal{E}_{\mathcal{G}}$, δ, c, $|\mathcal{C}|$, *group sizes of* $\mathcal{E}_{\mathcal{G}})$;
$\{q_1, ..., q_N\} \leftarrow$ for each client $i \in \mathcal{C}$: get its sampling rate q_i regarding their privacy group from $Q_{\mathcal{G}}$;
foreach *round of training in* I_{Fed} **do**
  Sample client subset $\mathcal{S}$ according to $\{q_1, ..., q_N\}$;
  foreach *client* $i \in \mathcal{S}$ **do**
    Local: $\Delta_i \leftarrow$ `LocalUpdate` $(i, \eta_{\mathcal{C}})$;
    Norm of update: $\|\Delta_i\|$;
    if $\|\Delta_i\| > \gamma$ **then**
      Clip: $\Delta_i \leftarrow$ `ClipUpdate` (Δ_i, γ);
    end
  end
  Aggregate: $\Delta \leftarrow$ `AggregateUpdates` $(\mathcal{S})$;
  Noise: $\Delta \leftarrow$ `AddNoise` $(\Delta, \sigma_{SAMPLE})$;
  Global: $\theta' \leftarrow$ `UpdateGlobalModel` (θ, Δ, η_s);
  $\gamma' \leftarrow$ `AdjustClippingNorm` (γ, η_C);
end
return θ', γ';

B. Federated Datasets

We test on common benchmarking datasets from related work: FMNIST [27], CIFAR-10 [28], and SVHN [29]. Since CIFAR-10 and SVHN are focused on central scenarios, we have to transform them into FL-conform counterparts that are split across clients. For this, we also have to consider creating i.i.d. and non-i.i.d. versions to address both paradigms. In FL, i.i.d. refers to samples that are independent (do not influence another) and identically distributed at clients. In contrast, non-i.i.d. data lacks these properties, with dependencies between samples and varying distributions, as in real-world scenarios.

The Federated MNIST (FMNIST) dataset contains about 380.000 grayscale images of handwritten digits and is an non-i.i.d. Extended MNIST version, where the digits are grouped by their respective authors across 3,383 clients. For an i.i.d. setup, we randomly distribute samples across clients.

CIFAR-10 focuses on 60,000 color images divided into 10 classes of vehicles and animals. To adapt this dataset to FL, we follow the CIFAR-100 sibling, which has an non-i.i.d. variant. Thus, we sample data for each of the 500 clients using the Pachinko Allocation Method [30], which unevenly distributes classes. For i.i.d., we use the same method as for FMNIST.

The SVHN dataset features over 600,000 color images of

Fig. 1. Examples of label distribution on clients for our datasets (non-i.i.d.).

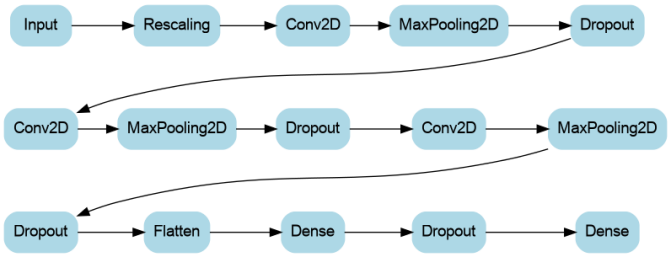

Fig. 2. CNN model architecture used in our experiments.

house numbers extracted from Google Street View. However, there is no non-i.i.d. federated version and we therefore only assume our random i.i.d. distribution over 725 clients. But as shown in Fig. 1, we still see some skew due to some labels being overrepresented in SVHN. We can also expect a very difficult task from CIFAR-10 due to its uneven distribution. On a final note, we use the original test sets for each dataset.

C. Training Parameters

Our ML model is a simple Convolutional Neural Network (CNN) with 14 layers as presented in Fig. 2. For FL server hyperparameters, we generally train for 420 rounds, sample 30 clients each round, and use a server learning rate of 1.0 for all datasets. At the clients, we use a batch size of 128 and learning rate of 0.0005 over 15 epochs. For adaptive clipping we use the standard parameter setup from [26].

D. Privacy Distributions

Regarding the evaluated privacy distributions, we incorporate real-world IDP, as well as, standard DP and non-DP settings. For simulating these distributions, we create three privacy groups of differing ε-values like ε : *1-2-3* and assign them across clients accordingly. As in [17], we also rely on existing user studies for our realistic privacy distributions, which are *34%-43%-23%* [24] and *54%-37%-9%* [23]. We further also include the strictest *100%-0%-0%* and most relaxed *0%-0%-100%* DP settings using only the respectively ranked ε from the privacy groups. To put these differentially-private results into context we also train federated models without privacy restrictions *0%-0%-0%* (non-private) as our last distribution.

TABLE I
ACCURACIES (%) OF THE MODEL SETUPS ON ALL DATASETS

ε-distribution	FMNIST ε : *1-2-3*		SVHN ε : *10-20-30*		CIFAR-10 ε : *10-20-30*	
	i.i.d.	non	i.i.d.	non	i.i.d.	non
0%-0%-0%	98.2	97.0	86.6	–	61.0	44.2
0%-0%-100%	96.5	95.1	80.9	–	50.1	30.1
34%-43%-23%	95.9	94.6	79.3	–	44.6	26.5
54%-37%-9%	95.4	93.5	78.2	–	41.9	25.4
100%-0%-0%	93.9	90.7	76.5	–	38.4	16.8

VI. RESULTS

We gather our experimental results in Table I, where we give the percentage accuracy values for the respective models regarding the different privacy distributions. We test on all three datasets with their i.i.d. and non-i.i.d. versions. With our results, we confirm the effectiveness of our adapted algorithm. This can be derived from the stepwise change in accuracies between the privacy distributions, where *0%-0%-0%* poses the least strict (non-private) and *100%-0%-0%* the strictest. While the strictest DP level gives the lowest performance outcomes, our individualized methods put the realistic distributions in between and closer to the relaxed DP assumption of the weakest ε-group. In non-individualized real-world scenarios we would have to choose the strictest level at all times but with individual privacy groups through our client sampling method, we see an average advantage of 2.2% for our i.i.d. and 5.7% for our non-i.i.d. datasets, when assuming a *54%-37%-9%* distribution. With the slightly less strict *34%-43%-23%*, we see further average improvement by 1.5% and 1.1%.

We however also see how differently DP effects our datasets. Even with loosened privacy requirements of ε : *10-20-30* for SVHN and CIFAR-10, there is a significant performance hit already when going from the non-private models to the most relaxed DP models in CIFAR-10. At ε : *1-2-3*, private models failed for both datasets, resolving to random guessing accuracies. FMNIST on the other hand, shows to be less impacted overall. This is comparable to other existing work, where CIFAR-10 clearly showed to be a more difficult FL task that might ask for a higher privacy budget [31]. Additionally DP shows to have an even greater trade-off in FL than in central settings due to the smaller client datasets and resulting larger impact of noise on the global model [32]. With CIFAR-10 being substantially smaller than the others, we can confirm this phenomena in our results. The non-private models are still able to achieve good performance levels but especially the non-i.i.d. version poses a challenge.

Regarding related work, we can compare our results for non-i.i.d. FMNIST to [16], who used the `SCALE` variant of IDP and only two privacy groups: a non-private and a private group ($\varepsilon = 0.6$). Their non-private group constitutes only 5% of their clients and we therefore create another run to match their setup: ε : *0.6*-∞-∞ at *95%-0%-5%*. They achieved 86.9% under these conditions, while we climbed up to 90.8% using the `SAMPLE` strategy. This supports that the advantage from the central setting carries over to FL by a larger margin.

VII. Discussion

Our experimental results demonstrate that individualized client sampling within FL can bridge the gap between strict privacy requirements and model utility. By successfully leveraging client-level sampling rates derived from privacy budgets (`SAMPLE`), our adapted `IDP-FedAvg` algorithm achieves significant accuracy improvements over uniform `DP-FedAvg` setups under realistic privacy distributions. We further confirm the advantage of individualizing sampling rates over noise scales in FL through our 3.9% lead on FMNIST compared to related work in [16]. However, our results also highlight the limitations of incorporating DP in FL for smaller and more complex datasets like CIFAR-10. These datasets suffer from higher sensitivity to the noise introduced by DP.

As others, we do not consider limitations for this approach regarding some common practical issues in FL, such as device heterogeneity and client availability or dropout during training in our experiments. For a more holistic and realistic view, removing our optimistic assumption would make reliable sampling of clients harder for the central server.

The proposed method has several practical implications. Allowing users to define their privacy levels addresses a key concern in privacy-preserving systems: trust. By enabling personalized privacy budgets, users retain control over their data while contributing at a level that aligns with their comfort. This flexibility incentivizes participation, as users are not forced to adhere to uniform and potentially insufficiently strict privacy guarantees. Instead, they can balance their privacy preferences with their willingness to support collaborative learning efforts. From the perspective of ML practitioners, IDP in contrast to traditional approaches enables to benefit from higher-quality updates of users with more relaxed privacy settings. A key challenge for this aspect of IDP is the increased complexity for the user, which necessitates effective and fair communication of privacy risks to enable users to make informed assessments.

VIII. Conclusion

We present an adaptation of the IDP `SAMPLE` algorithm for FL, introducing individualized client sampling to enable heterogeneous privacy guarantees. Our `IDP-FedAvg` approach demonstrates significant utility improvements under realistic privacy distributions compared to traditional `DP-FedAvg`. As in central settings, the sampling method is able to outperform noise parameter scaling. However, the results also highlight the challenges of applying DP in federated settings, particularly for non-i.i.d. datasets and complex tasks. Future work could explore incorporating IDP through sampling rates into alternative aggregation mechanisms beyond `FedAvg`, which may better handle non-i.i.d. and noisy data distributions.

Acknowledgment

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus for ScaDS.AI. This work was supported by DAAD project-related personal exchange under Project-ID: 57701258.

References

[1] Y. Yao et al., "A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly," *High-Confid. Comput.*, 2024.
[2] B. Liu et al., "When machine learning meets privacy: A survey and outlook," *ACM Comput. Surv. (CSUR)*, vol. 54, no. 2, 2021.
[3] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *CSF*. IEEE, 2018.
[4] L. Lange, T. Schreieder, V. Christen, and E. Rahm, "Slice it up: Unmasking user identities in smartwatch health data," in *AsiaCCS*. ACM, 2025, in press.
[5] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1-2, 2021.
[6] J. Zhou et al., "Guest editorial: Federated learning for industrial iot in industry 4.0," *IEEE Trans. Ind. Inform.*, vol. 17, no. 12, 2021.
[7] N. Ponomareva et al., "How to dp-fy ml: A practical guide to machine learning with differential privacy," *J. Artif. Intell. Res.*, vol. 77, 2023.
[8] Apple. (2018) Apple differential privacy technical overview. Accessed 2025-01-21. [Online]. Available: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf
[9] K. Tezapsidis. (2017) Uber releases open source project for differential privacy. Accessed 2025-01-21. [Online]. Available: https://medium.com/uber-security-privacy/differential-privacy-open-source-7892c82c42b6
[10] F. Boenisch et al., "When the curious abandon honesty: Federated learning is not private," in *EuroS&P*. IEEE, 2023.
[11] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *ICLR*, 2018.
[12] L. Lange, M.-M. Heykeroth, and E. Rahm, "Assessing the impact of image dataset features on privacy-preserving machine learning," in *BTW*. GI, 2025, in press.
[13] B. Berendt, O. Günther, and S. Spiekermann, "Privacy in e-commerce: Stated preferences vs. actual behavior," *Commun. ACM*, vol. 48, no. 4, 2005.
[14] G. Yang, S. Wang, and H. Wang, "Federated learning with personalized local differential privacy," in *ICCCS*. IEEE, 2021.
[15] X. Shen et al., "PLDP-FL: federated learning with personalized local differential privacy," *Entropy*, vol. 25, no. 3, 2023.
[16] N. Aldaghri, H. Mahdavifar, and A. Beirami, "Federated learning with heterogeneous differential privacy," *arXiv:2110.15252*, 2023, preprint.
[17] F. Boenisch, C. Mühl, A. Dziedzic, R. Rinberg, and N. Papernot, "Have it your way: Individualized privacy assignment for DP-SGD," *NIPS*, vol. 36, 2024.
[18] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," in *AISTATS*, 2017.
[19] C. Dwork, "Differential privacy," in *ICALP*. Springer, 2006.
[20] M. Abadi et al., "Deep learning with differential privacy," in *CCS*. ACM, 2016.
[21] Z. Jorgensen, T. Yu, and G. Cormode, "Conservative or liberal? personalized differential privacy," in *ICDE*. IEEE, 2015.
[22] M. Alaggan, S. Gambs, and A. Kermarrec, "Heterogeneous differential privacy," *J. Priv. Confidentiality*, vol. 7, no. 2, 2016.
[23] C. Jensen, C. Potts, and C. Jensen, "Privacy practices of internet users: Self-reports versus observed behavior," *Int. J. Hum. Comput. Stud.*, vol. 63, no. 1-2, 2005.
[24] A. Acquisti and J. Grossklags, "Privacy and rationality in individual decision making," *IEEE Secur. Priv.*, vol. 3, no. 1, 2005.
[25] J. Liu et al., "Projected federated averaging with heterogeneous differential privacy," *Proc. VLDB Endow.*, vol. 15, no. 4, 2021.
[26] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, "Differentially private learning with adaptive clipping," in *NIPS*, vol. 34, 2021.
[27] S. Caldas et al., "Leaf: A benchmark for federated settings," *arXiv:1812.01097*, 2018, preprint.
[28] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report, University of Toronto*, 2009.
[29] Y. Netzer et al., "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on DLUFL*, 2011.
[30] W. Li and A. McCallum, "Pachinko allocation: Dag-structured mixture models of topic correlations," in *ICML*, 2006.
[31] L. Sun, J. Qian, and X. Chen, "LDP-FL: practical private aggregation in federated learning with local differential privacy," in *IJCAI*, 2021.
[32] K. Wei et al., "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, 2020.