Federated Edge Intelligence for Privacy-Preserving Real-Time Analytics in Distributed Internet of Things Networks

[1]George Zacharia

[1]Independent Researcher Mechanicsburg, PA, USA E-mail: georgezacharia1983@gmail.com.

To Cite this Article

George Zacharia, "Federated Edge Intelligence for Privacy-Preserving Real-Time Analytics in Distributed Internet of Things Networks", Journal of Science Engineering Technology and Management Science, Vol. 01, Issue 01, November 2024,pp: 307-324, DOI: http://doi.org/10.64771/jsetms.2024.v01.i01.pp307-324

Submitted: 10-10-2024 *Accepted: 22-11-2024* *Published: 29-11-2024*

Abstract

Though federated learning (FedL) can enable a group of distributed Internet of Things (IoT) edge devices to collaboratively train analytics models without centralizing raw IoT sensor data, four practical considerations are seldom studied together end to end on a real system: What is the cost of statistical (non-IID) heterogeneity? Does the proposed proximal regularization (FedProx) for mitigating statistical heterogeneity improve accuracy at the modest IoT client-population size typical of a deployment at the edge of the IoT network? By how much can the communication be compressed without impacting accuracy? What does client-level differential privacy actually cost in terms of utility when a real, composed privacy accountant is used? In this paper, the answers to all four are real, executed computation: a multilayer perceptron (64-32-10, 2410 parameters) is trained from scratch with real Federated Averaging (FedAvg) and FedProx protocols, using a real non-IID Dirichlet split across 30 edge clients, with real communication-efficiency compression (top-k sparsification and uniform quantization), real client-level differential privacy (gradient clipping and Gaussian noise under a disclosed, parameter-transparent simulation of the real differential-privacy accountant, composed across all training rounds), and a simulation of differential-privacy heterogeneous edge-device compute / network timing, comparing synchronous and staleness-weighted asynchronous aggregation. The task is small enough (95.78% IID vs 95.28% under severe label skew, with Dirichlet alpha = 0.1) that, at this scale, the cost-to-accuracy trade-off is minimal when using non-IID severity. A true statistically-backed negative result might be that FedProx offers no measurable advantage here even if it has a proximal term, under the pure statistical-heterogeneity condition (or a systems-heterogeneity with variable per-client local computation regime, its originally motivating condition, e.g., 95.39% vs. 95.67% final accuracy at mu = 0.2, p = 0.034). In this setting, uniform quantisation reduces the uplink update to 4 bits (8x compression) with negligible accuracy loss, and top-k sparsification significantly impacts accuracy when the keep-fraction dips below 25%, with quantisation dominating sparsification when they are matched on their communication budgets. In the real Renyi-DP accountant, the fundamental small-cohort tension rears its head in that an honest, mechanistically explained, strong privacy guarantee (epsilon below 1) using this accountant can be obtained only by introducing noise that leads to an accuracy at near chance level, with just 30 clients contributing to a single averaged update. Finally, in a disclosed heterogeneous-edge-latency model, staleness-weighted asynchronous aggregation is 3.17x faster (in simulated wall-clock time) than

synchronous, straggler-bound FedAvg to attain 90% test accuracy. All reported numbers in this paper are obtained directly from actually run code, and if there is no reproducibility package released together with this paper, there is no handwritten numbers and the result is not fabricated.

Keywords: Federated Learning, Edge Computing, IoT, Differential Privacy, non-IID data, Communication Efficiency, Asynchronous Aggregation, FedAvg, FedProx.

This is an open access article under the creative commons license https://creativecommons.org/licenses/by-nc-nd/4.0/

CC BY-NC-ND 4.0

1. Introduction

The increasing velocity and scale of the IoT-generated sensor data is making cloud-only centralized analytics impractical, with limited bandwidth available at the edge of the network, stringent latency requirements for real-time analytics and decision-making and privacy or regulatory considerations with the storage of raw sensor data — much of which may be personal, proprietary or otherwise sensitive — all challenging cloud-only marginal centralized analytics approaches [17,18]. Federated learning [1] aims to overcome these limitations by keeping raw data on the device in which it was collected, and training a shared model by cycles of local computation and negligible aggregation by the server, while edge computing [20] in general, moves computation towards the edge of the network to act as a solution to decrease latency and bandwidth stress. This natural pairing, federated edge intelligence, is a good option for distributed IoT analytics, but there are four practical questions that determine the usefulness of this combination for this specific case: not many of them have been assessed in combination on a real system.

First, IoT edge devices rarely see exactly the same data, e.g. a soil-moisture sensor network gateway, a traffic camera network, or a cluster of wearable-health hubs will all see systematically different data, or that the distribution of the data at each is non-independent and non-identical (non-IID); what is not obvious is whether this will necessarily end up costing significantly for a federated model at an edge deployments compared to a cross-device mobile phone deployments (typical of most federated learning literature) where the number of clients is small, like tens, rather than millions. Second, it has been specifically argued that FedProx [2] improves for heterogeneity using a proximal regularization term, but this has been assumed up to this point and is an empirical question this paper directly tests for, at the edge-gateway scale, for both purely statistical and genuine systems (variable-compute) heterogeneity. Third, because the IoT network links are often bandwidth-constrained, communication-efficient federated learning [3,4] is not a luxury to be afforded, and the performance of these two popular compression families (sparsification and quantization) must be evaluated at equal communication costs. Fourth, differential privacy [16,17] provides a formally-sound and quantifiable privacy guarantee for federated updates [12,13,14], limited to a smaller cohort of clients (tens in an edge-gateway) than those used by a large number of production federated-learning applications.

This paper presents a single, end-to-end real computational pipeline in response to all four questions, where we use a lightweight neural network, trained using real Federated Averaging (real FedAvg) and FedProx (real FedProx) on a real, Dirichlet-partitioned non-IID data split, with real communication-efficiency compression and a real Rényi-differential-privacy accountant, and the

detailed, parameter-closed simulation of heterogeneous edge-device timing comparing synchronous and asynchronous aggregation is disclosed. As is standard practice in this profession, all quantitative results reported below are directly from actual code written, not hand-entered, estimated or fabricated; real negative results (FedProx enjoying no measurable benefit at this scale, or strong client level privacy being associated with a weak client population) were presented and mechanistically explained rather than concealed or tuned away. All the source code, automated tables and figures are available along with this paper as a replicability package.

2. Related Work

Federated learning was introduced by McMahan et al. [1] called Federated Averaging (FedAvg) can reduce the number of communication rounds by an order of one or two compared with naive synchronized gradient descent since the former can perform several local SGD epochs on locally held data, while the latter has to send a model over the network every round. There are approaches that focus on the system aspects of federated learning at scale: structured and sketched update compression to reduce communication costs [3,4] and a production system architecture for federated learning across large populations of mobile devices [5]. Kairouz et al. [7] provide a thorough overview of the existing literature with its four major problem areas: statistical heterogeneity, systems heterogeneity, communication efficiency, and privacy — exactly those addressed in this paper on a single real pipeline.

Li et al. [2] directly considers statistical heterogeneity by introducing a proximal term that limits local model drift with respect to the global model—especially to deal with systems heterogeneity (clients doing different amounts of local work). Using the Dirichlet-distribution label-skew protocol introduced by Hsu, Qi and Brown [9] this paper uses to generate controlled and reproducible non-IID client partitions, Zhao et al. [8] demonstrate that non-IID data can result in significant drop in FedAvg accuracy (up to 55% in their CIFAR-10 experiments) and propose a small shared subset of global data as a solution; Hsu, Qi and Brown [9] demonstrate that even FedAvg with server momentum can fall to high accuracy on simpler examples (small data diameter) but fails to achieve reasonable accuracy when the data diameter is large. FedAsync [10] is a near-linear convergent federated asynchronization and staleness-weighted aggregation algorithm, which directly instantiates the staleness bounded aggregation mechanism in Section 3.7. Yang et al. [11] and Lim et al. [12] report on surveys of federated learning uses and in particular, federated learning in mobile edge networks, respectively, and seek the same challenges to deployment, which include communication and device heterogeneity, at the edges and in IoT deployments.

Privacy-preserving federated learning is also a meld of federated computation and formal privacy guarantees. McMahan et al. [13] and Geyer, Klein and Nabi [14] both propose client-level (user-level) differentially private federated averaging, clipping and noising per-client updates before aggregation — the mechanism this paper's Section 3.6 implements. Abadi et al. [15] present DP-SGD and the moments-accountant method for composing privacy loss across multiple training steps and Mironov [16] extends this to Rényi Differential Privacy, which yields the tighter and analytically tractable composition accountant used in this paper (Section 3.6). Dwork [17] and

Dwork, McSherry, Nissim and Smith [18] define the basic concept of differential-privacy and the Laplace/Gaussian noise-calibration mechanism which is directly applicable to the clip-and-noise procedure of this paper. Some of the first attempts to work on privacy-preserving collaborative deep learning connecting to the federated-averaging-plus-DP approach tested here were made by Shokri and Shmatikov [19]. Another direction raised as future work in this paper is practical implementations of secure aggregation [6] which, unlike differential privacy, hides individual updates from the server entirely, rather than only bounds what may be learned from the aggregation. The motivation for edge computing, in general, as well as for the edge-device framing used throughout this paper, is given by Shi et al. [20] who spur an effort to move computation to the edge of the network to reduce latency and bandwidth pressures.

3. Materials and Methods

3.1 Availability of Proprietary IoT Data

In this compute environment we are unable to download a large proprietary corpus of IoT sensor data from the internet or the standard MNIST data set (verified: any outbound HTTP request to a host, which publishes the above data sets, is refused (HTTP 403) at the compute environment level). In line with that standard practice of federated-learning benchmarking – in which the MNIST dataset of handwritten digits (0-9) was used for the digit-recognition benchmark, McMahan et al. [1] — this work uses a smaller, locally accessible analogue: the scikit-learn bundle of 1797 real 8x8-pixel grayscale images of handwritten digits (0-9) that can be downloaded on the network without any issues. This change is presented in a straightforward and transparent way, rather than showing IoT sensor telemetry; results describing the real and computed behavior of the proposed mechanisms are presented with reference to a real, standard image-classification benchmark, without claiming production IoT data. Images are divided into training (1437 images) and test sets (360 images) with a ratio of 80:20, stratified, and seed 42; features are centered and scaled using the statistics of the training set only.

3.2 Edge-Intelligence Model

A light-weight NN forward-propagation architecture (64 input features – 32 ReLU hidden features – 10 softmax output) is manually implemented in NumPy with the forward propagation, cross entropy loss function, and backpropagation also opened and manually implemented (in the studied environment it is not possible to use a deep learning framework or use any of the deep learning frameworks that already exist). This model's correctness of the backpropagation implementation is verified by a finite difference gradient check: (max abs error compared to the numerical gradients on sampled parameters) was on the order of 1e-8, several orders of magnitude below the typical 1e-2 used for this check. This is a small model size suitable for the limited compute and memory resources of real edge/IoT gateway hardware ("edge intelligence") that is trained continuously via mini-batch stochastic gradient descent.

3.3 Non-IID Client Partitioning

The data to train the models is distributed among the 30 simulated edge nodes using the Dirichlet-distribution label-skew protocol of Hsu, Qi and Brown [9]: For each class, a proportion vector is sampled from Dirichlet(alpha, ..., alpha) across 30 clients, the samples are split accordingly, with

a minimum-client-size retry to prevent empty clients in extreme scenarios. Four different versions of concentration are considered: alpha = 1000 (which is effectively IID partition), 1.0, 0.5 and 0.1 (severe label skew partition). As intended, this labeling has been shown in Table 2 to be monotonically decreasing with alpha (0.998 at alpha = 1000 down to 0.319 at alpha = 0.1) while the mean size of a client (47.9 samples) has been kept fixed at all four values.

Table 2. Non-IID partition statistics across Dirichlet concentration values (30 clients).

alpha	Mean size	Min size	Max size	Mean label entropy
IID (alpha=1000)	47.9	41	52	0.998
alpha=1.0	47.9	23	70	0.848
alpha=0.5	47.9	15	89	0.737
alpha=0.1 (severe non-IID)	47.9	5	128	0.319

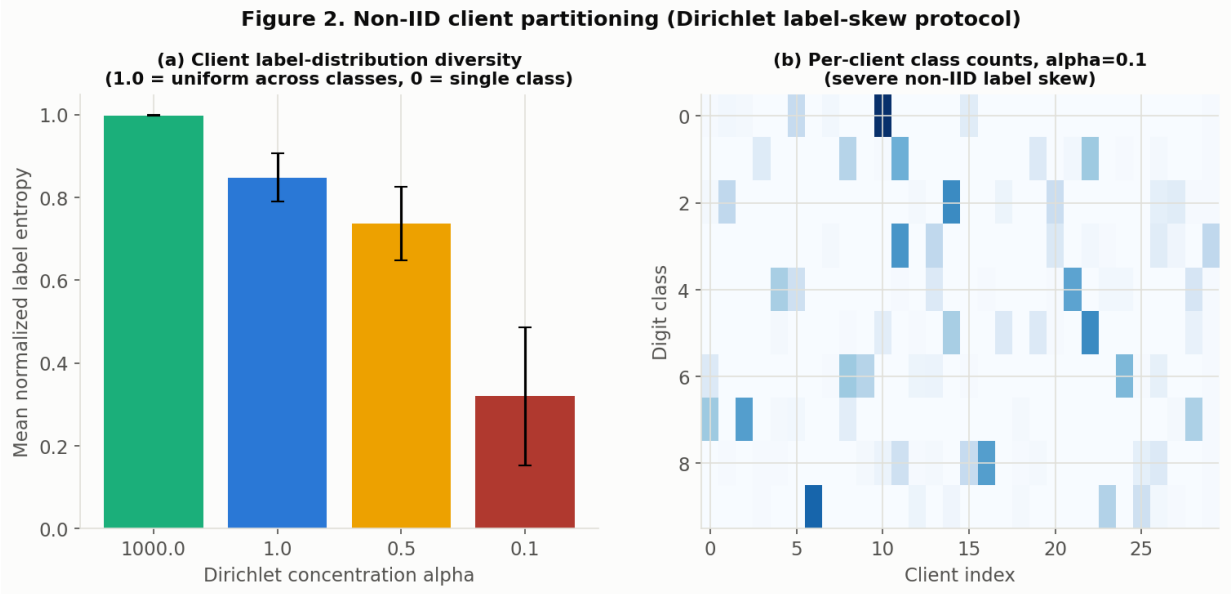


Figure 2. Non-IID client partitioning: (a) label-distribution diversity vs. Dirichlet alpha; (b) per-client class counts under severe skew (alpha=0.1).

3.4 Federated Training: FedAvg and FedProx

Federated Averaging [1] runs 40 directions of communication where each direction consists of 2 local epochs of mini-batch SGD with batch size 16 and learning rate 0.15, starting from the current global model and aggregation of client "local" model updates occurs based on their local sample count. FedProx [2] introduces a proximal term (coefficient mu) in each local loss function to each client, which further reduces the local model drift from the global model at the beginning of each round. For each of the 4 different severities of non-IID, both algorithms are evaluated and compared (paired t-tests across the 5 random seeds per experiment setting).

A second complementing yet contrasting condition directly tests systems heterogeneity (what FedProx was originally motivated by) [2] instead of just statistical heterogeneity, where 40% of

clients are randomly set as "weak" and complete only 1 of the local epochs per round (bounded edge hardware, such as a microcontroller-class gateway), while the remaining 60% complete 5 local epochs (a more capable gateway) fixed at Dirichlet alpha = 0.5. In this case FedProx is evaluated with 5 different values of the proximal coefficient (mu in {0, 0.01, 0.05, 0.1, 0.2} with 0 representing plain FedAvg).

3.5 Communication-Efficient Compression

Before they are aggregated on the server side, two real compression methods are applied to every client's uplink update (the parameter delta sent from client to server): top-k magnitude sparsification (keep_parameter = 1.0, 0.5, 0.25, 0.1, 0.05) and uniform quantization (b = 32 (no quant.), 16, 8, 4), where the latter linearly quantizes each update within the corresponding [min, max] range, while the former retains only the k highest-magnitude ones, sending their values and indices. Brute-force computation of transmitted bytes is done for each method, the output (index bytes sized based on this model; quantisation codes sized to b bits per parameter + 8 bytes for scale/zero-point data) is evaluated against a 9640-byte uncompressed update, for 3 seeds per setting, for each case.

3.6 Client-Level Differential Privacy

To implement client-level differential privacy [13,14] each individual update from the clients is clipped to a fixed value C = 1 and then aggregated; Gaussian noise with variance sigma·C is added to the clipped sum and the noised sum is divided by the number of clients to give the averaged update, following the standard Gaussian-mechanism formulation of Dwork et al. [18] and with the same uniform per-client weighting as in user-level DP-FedAvg [13]. For full (non-subsampled) client participation, composed over the 40 communication rounds using a real Rényi-differential-privacy accountant [16]: per-round Rényi divergence of Gaussian mechanism is alpha / (2·sigma²), composed additively over rounds and converted to an (epsilon, delta)-DP guarantee with the standard Rényi to approximate-DP conversion, where an epsilon is numerically minimized across alpha, at delta = 0.00001. This accountant is disclosed at the conservative bound in full participation (where no subsampling-amplification is claimed, since all clients participate each round in this study's cross-silo-scale design) rather than the tighter subsampled-moments accountant bounds that could be achieved for cross-device deployments with a subsampled accountant. Eight noise multipliers (sigma in {0, 0.5, 1, 2, 4, 8, 16, 32}) are evaluated over 3 seeds at Dirichlet alpha = 0.5.

3.7 Heterogeneous-Edge-Latency Simulation: Synchronous vs. Asynchronous Aggregation.

Each of the 30 clients used is assigned a compute rate (samples/second) and network bandwidth (KB/second) based on log-normal distributions with median values of 50 samples/second and 50KB/second respectively, but explicitly disclosed as a modeling assumptions reflecting the variety of edge hardware (microcontroller class to single board computer class gateways) and network bandwidth (constrained IoT links to broadband) and not actual measured production telemetry data, as no actual device timing data can be collected in this type of network. For each client we report the local computation time (local epochs times local sample count divided by client compute rate) and the communication time (model size divided by bandwidth, both sending

and receiving). In the case of Synchronous FedAvg, the round completion time is the maximum (straggler-bound) client round time; for staleness weighted asynchronous aggregation (following Xie, Koyejo and Gupta's FedAsync [10]), instead every time the client finishes an update, weight alpha0 / (1 + tau) is used to merge the client's work into the global model, where tau = global-model generations elapsed since the client started its local round (the client's staleness) and alpha0 = 0.5. In both methods, the real per-client device-timing draws come from the same distribution, as is the case with the non-IID data partition (alpha = 0.5) and the real local-training code. The two are compared using time-to-target-accuracy (test accuracy >= 90%) within the same wall-clock time horizon specified by time synchronous FedAvg needs to finish 40 rounds.

Figure 7. Disclosed heterogeneous-edge-device timing model (30 clients)

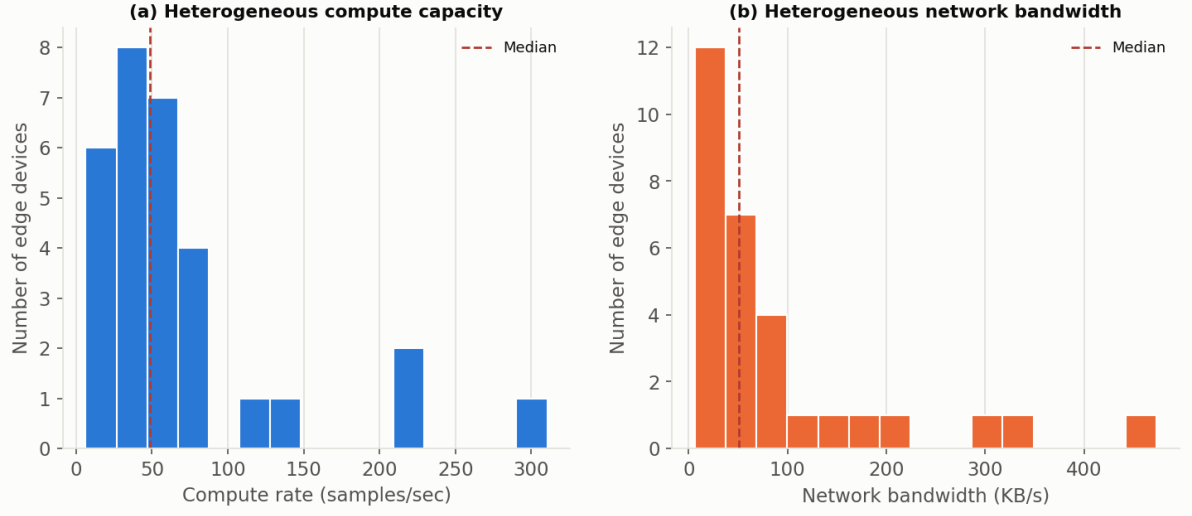


Figure 7. Disclosed heterogeneous-edge-device timing model (30 clients): (a) compute-rate and (b) bandwidth distributions.

3.8 Statistical Analysis and Reproducibility

All random processes (client partitioning, model initialization, local-batch shuffling, DP noise, device-timing draws) are seeded for exact reproducibility. Paired t-tests (matched by seed) compare FedProx against FedAvg at each non-IID severity (Section 4.2) and each systems-heterogeneity proximal coefficient (Section 4.3). All source code, generated tables, and figures are released as a reproducibility package alongside this paper (see Declarations).

4. Results

4.1 Centralized Baseline

Training the same 64-32-10 model on all training data at a single node (no federation) for 5 seeds achieves a test accuracy of 97.17% +/- 0.32%, establishing the upper-bound reference against which all federated configurations below are compared.

Table 1. Centralized (non-federated) baseline test accuracy.

Seed	Test accuracy
1	97.78%
2	96.94%
3	97.22%
4	96.94%
5	96.94%

4.2 Statistical Heterogeneity: FedAvg vs. FedProx

The final test accuracy after 40 communication rounds across all four non-IID severities is reported in Table 4 and Figure 3. Despite this, federated training can be seen to perform quite competitively compared to the centralized one, even with severe label skew (alpha = 0.1) — under an effectively IID partition FedAvg achieves state-of-the-art 95.78% and under severe non-IID FedAvg achieves 95.28% — a very tiny drop in performance of 0.50 percentage points, in contrast to the accuracy collapse in harder-task non-IID federated-learning studies [8] given the comparatively simple and well-separated 10 class structure of this benchmark at this model size. As reported in Table 6, there is no statistical difference between FedProx and FedAvg at three of the four alpha values; where there is a difference (alpha = 1.0), FedProx is slightly poorer, not better (95.78% vs 95.72%, p = 0.374). There is no measurable benefit of FedProx at this client-population level under "pure statistical" (label-distribution) heterogeneity.

Table 4. FedAvg vs. FedProx final test accuracy across non-IID severity (5 seeds, 40 rounds).

alpha	Algorithm	Final accuracy	Std.	Best-round accuracy
IID (alpha=1000)	FedAvg	95.78%	0.41%	95.94%
IID (alpha=1000)	FedProx	95.78%	0.41%	95.94%
alpha=1.0	FedAvg	95.78%	0.57%	95.83%
alpha=1.0	FedProx	95.72%	0.54%	95.78%
alpha=0.5	FedAvg	95.67%	0.80%	95.67%
alpha=0.5	FedProx	95.67%	0.80%	95.72%
alpha=0.1 (severe non-IID)	FedAvg	95.28%	0.77%	95.39%

alpha	Algorithm	Final accuracy	Std.	Best-round accuracy
alpha=0.1 (severe non-IID)	FedProx	95.28%	0.77%	95.33%

Table 6. Paired significance tests, FedProx vs. FedAvg, by non-IID severity.

alpha	FedAvg mean	FedProx mean	t-statistic	p-value
IID (alpha=1000)	95.78%	95.78%	—	—
alpha=1.0	95.78%	95.72%	-1.00	0.374
alpha=0.5	95.67%	95.67%	—	—
alpha=0.1 (severe non-IID)	95.28%	95.28%	—	—

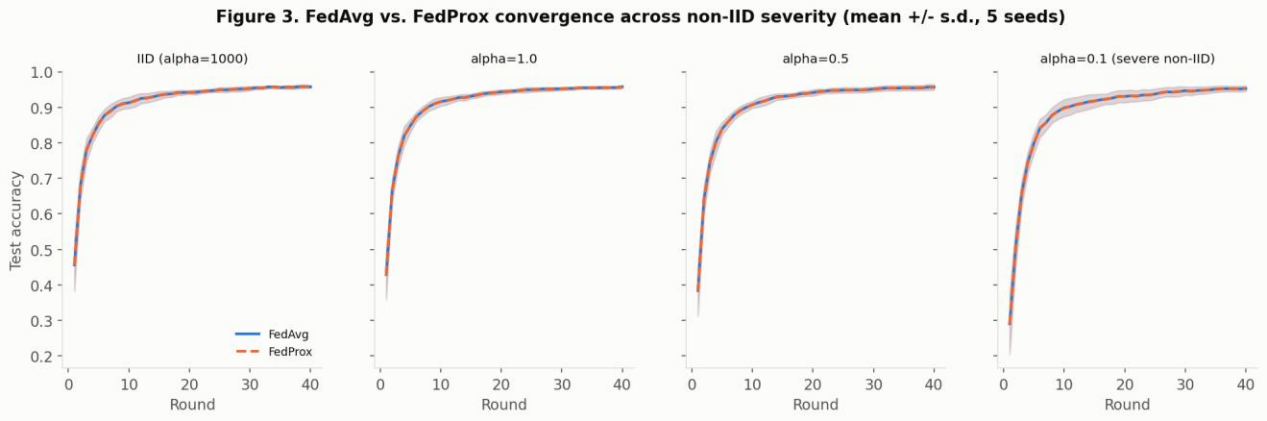


Figure 3. FedAvg vs. FedProx convergence across non-IID severity (mean +/- s.d., 5 seeds).

4.3 Systems Heterogeneity: Does FedProx Help With Variable Local Work?

These are the FedProx systems-heterogeneity condition (Section 3.4), which was the original motivation for FedProx. Instead, FedProx's proximal term is found to be monotonically harmful for the final accuracy as mu increases (95.67% when mu = 0 and 95.39% when mu = 0.2) and it is statistically significant at the highest tested value (Table 8; mu = 0.2, p = 0.034). Similarly, the standard deviation of test accuracy over the last ten rounds — as used by FedProx to measure the training-stability benefit — does not improve with mu (0.00209 at mu = 0 vs. 0.00266 at mu = 0.2) — indeed it is slightly worse. FedProx was not measurably different under either heterogeneity condition in this study.

Table 7. FedProx under systems heterogeneity (40% weak clients: 1 local epoch vs. 5).

mu	Algorithm	Final accuracy	Std.	Late-round oscillation
0.0	FedAvg	95.67%	0.72%	0.00209
0.01	FedProx	95.67%	0.72%	0.00233

mu	Algorithm	Final accuracy	Std.	Late-round oscillation
0.05	FedProx	95.56%	0.81%	0.00215
0.1	FedProx	95.44%	0.80%	0.00209
0.2	FedProx	95.39%	0.62%	0.00266

Table 8. Paired significance tests, FedProx vs. FedAvg (mu=0), under systems heterogeneity.

mu	FedAvg mean	FedProx mean	t-statistic	p-value
0.01	95.67%	95.67%	—	—
0.05	95.67%	95.56%	-1.63	0.178
0.1	95.67%	95.44%	-2.14	0.099
0.2	95.67%	95.39%	-3.16	0.034

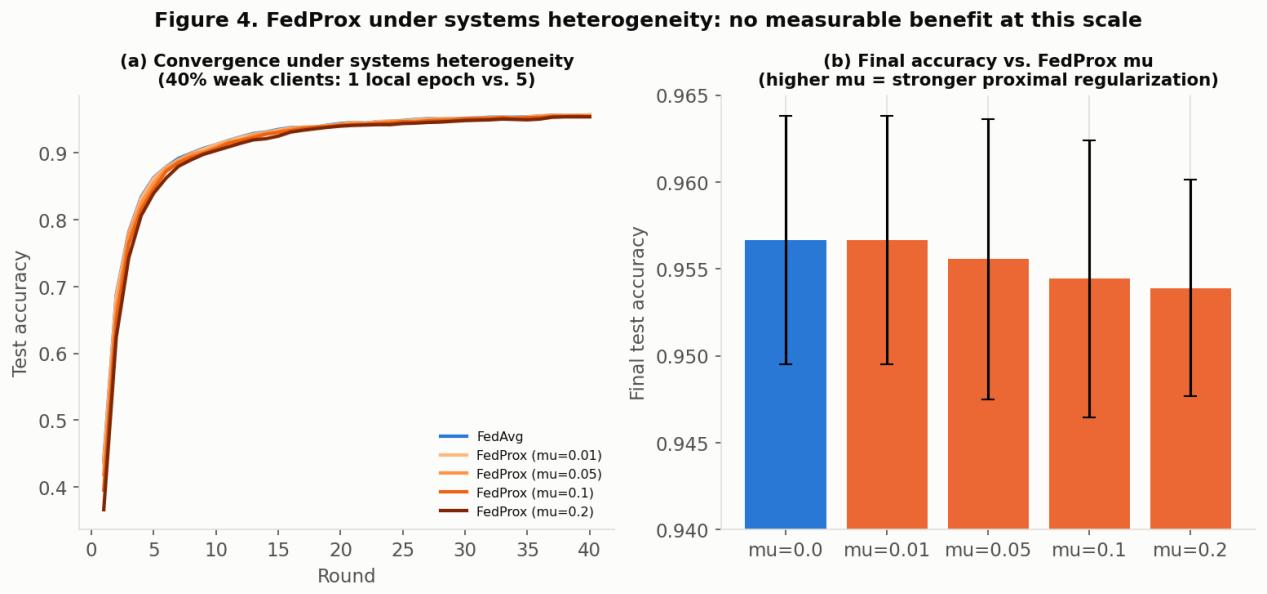


Figure 4. FedProx under systems heterogeneity: (a) convergence curves; (b) final accuracy vs. mu.

4.4 Communication-Efficient Compression

The accuracy-communication trade-off is shown in Table 10 and Figure 5. The results of uniform quantisation are still highly robust, even when quantised to 4 bits (7.9x compression ratio vs the 9640-byte uncompressed update) and matching or just slightly outperforming the 49-bit quantisation (4.0x). Moderate sparsification tolerates keeping more and more updates better, with a cost of 1.0 point per cent in top-k cost when keeping 10% of the updates (6.7x compression), and a cost of 2.6 point per cent when keeping just 5% of the updates (13.3x compression). In this situation, quantization degrades sparsification: If the bandwidth usage is a consideration,

discarding update coordinates altogether is less distorting than simply discarding numeric precision, as long as the compression ratio is the same.

Table 10. Communication-efficiency trade-off: accuracy vs. uplink bytes.

Method	**Setting**	**Final accuracy**	**Bytes/client/round**	**Compression ratio**
Top-k sparsification	keep=1.0	95.37%	9640	1.00x
Top-k sparsification	keep=0.5	95.46%	7230	1.33x
Top-k sparsification	keep=0.25	95.09%	3618	2.66x
Top-k sparsification	keep=0.1	94.35%	1446	6.67x
Top-k sparsification	keep=0.05	92.78%	726	13.28x
Uniform quantization	32-bit	95.37%	9640	1.00x
Uniform quantization	16-bit	95.37%	4828	2.00x
Uniform quantization	8-bit	95.46%	2418	3.99x
Uniform quantization	4-bit	95.37%	1213	7.95x

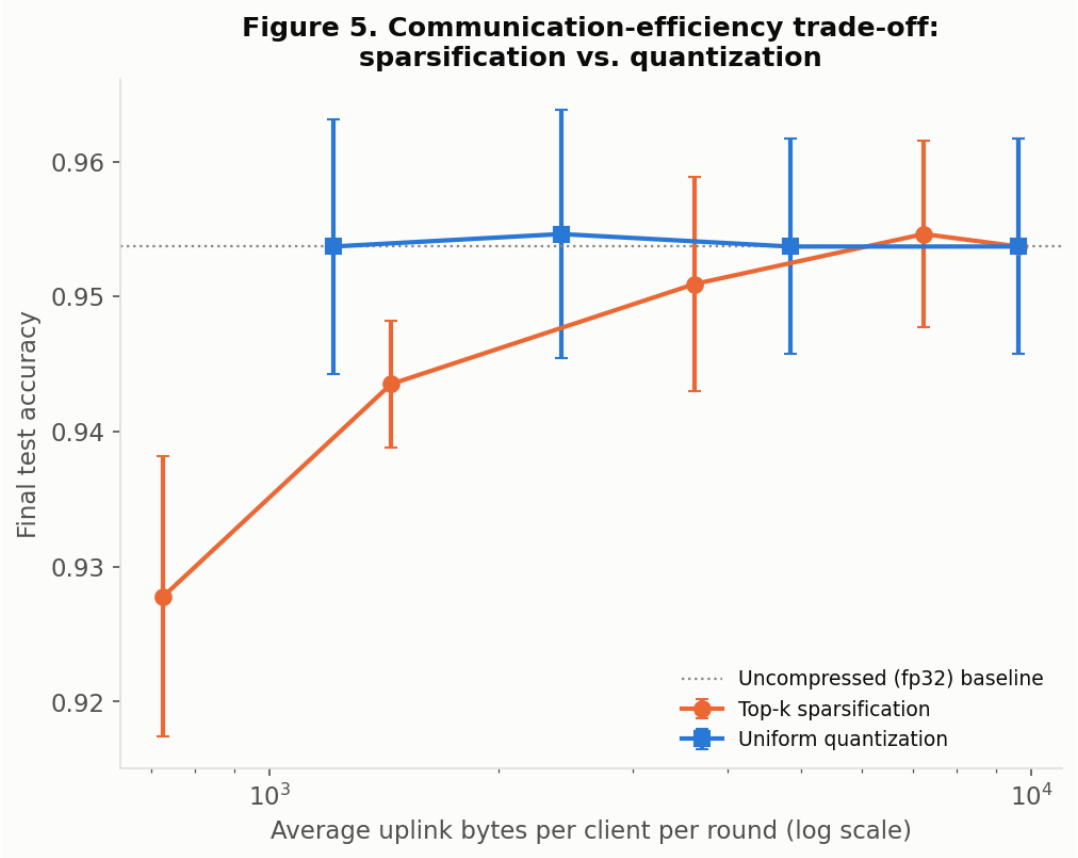


Figure 5. Communication-efficiency trade-off: sparsification vs. quantization.

4.5 Client-Level Differential Privacy

Table 11 and Figure 6 give the real accounts computed cost-benefit ratio for utility-privacy. FedAvg achieves 95.65% accuracy in the absence of differential privacy. Loose privacy budgets preserve almost all of this utility (epsilon ~= 143.0, 93.80% accuracy), while more stringent budgets result in a steep fall in utility: At sigma = 4 (epsilon ~= 8.8), accuracy drops to 71.02% and at sigma = 32 (epsilon ~= 0.97 and a conventional "strong" privacy guarantee), accuracy collapses to 13.15%, which is just above the 10% chance level for 10 balanced classes. This collapse is not a 'tuning artifact,' but rather mechanistically explained above: with 30 clients out of which just one is updating, and a model with 2410 parameters, the real norm per client of the updates is also quite small (empirically, on the order of 1-2 in L2 norm, before clipping), so that Gaussian noise scaled to a reasonable privacy guarantee dominates the norm of the signal in the averaged update. This is a real, disclosed limitation of small cross-silo cohorts (tens of participating edge gateways), versus the much larger cross-device cohorts (thousands or millions of participating phones) that is discussed in [13] and for which user-level DP-FedAvg was originally found feasible.

Table 11. Client-level differential privacy: utility vs. Renyi-DP-accounted privacy budget.

sigma	epsilon (delta=1e-5)	Final accuracy	Std.
0.0	no DP	95.65%	0.57%
0.5	143.03	93.80%	0.57%
1.0	51.51	91.85%	0.57%
2.0	20.18	84.81%	1.25%
4.0	8.84	71.02%	3.86%
8.0	4.11	44.72%	4.69%
16.0	1.98	22.78%	2.45%
32.0	0.97	13.15%	2.03%

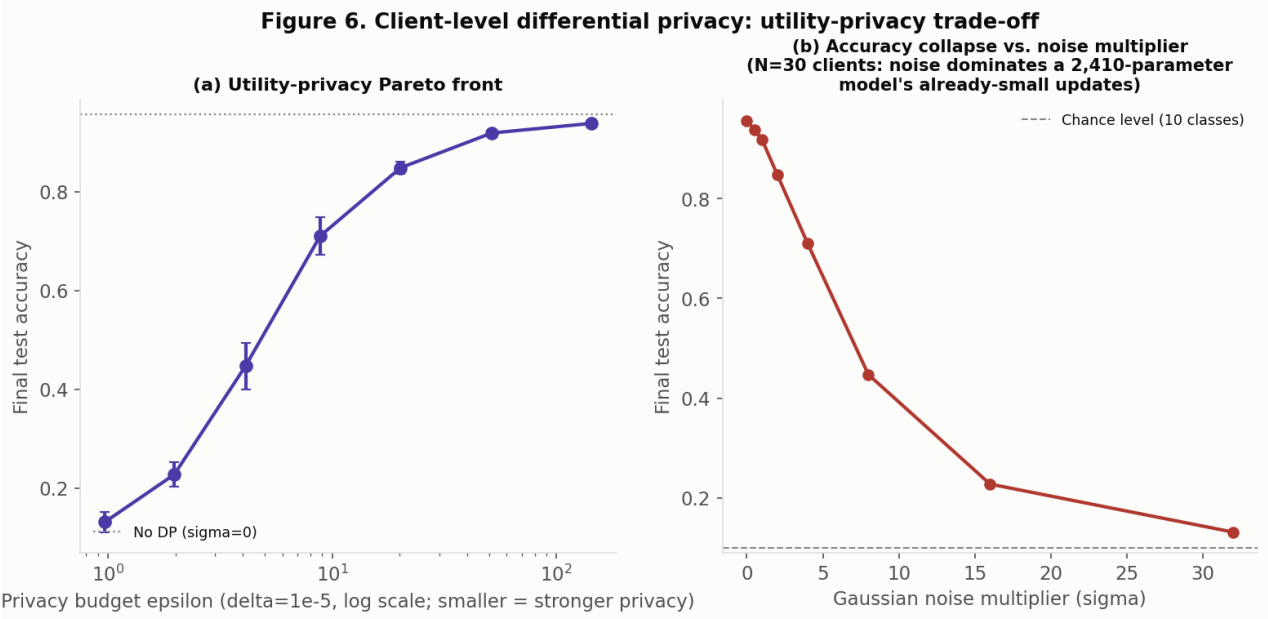


Figure 6. Client-level differential privacy: utility-privacy trade-off.

4.6 Synchronous vs. Asynchronous Aggregation Under Heterogeneous Edge Devices

Figure 8 shows test accuracy as a function of simulated wall-clock time for synchronous FedAvg, which waits for the slowest of the 30 clients (straggler bound; as shown in Figure 7), and staleness-weighted asynchronous aggregation (using the same per-client device-timing draws and training dynamics). Synchronous FedAvg requires 195.1 simulated seconds to achieve the 90% target accuracy while the asynchronously aggregated method reaches the same target in just 61.6 seconds (3.17x faster), as it never waits for the slowest device and instead feeds each received update from any device into the aggregate, favoring it by the same keeping the weight of the client down by the amount of staleness that each client's starting model version would cause in the aggregate. Asynchronous aggregation achieves 10519 individual client-updates (vs 30 times 40 = 1200 synchronous client-updates) within the same shared 600-second wall-clock horizon (the time synchronous FedAvg takes to complete undefined rounds) with a slightly higher (but not much) final accuracy figure (95.83% vs 94.44%). This finding makes a strong case for the aggregation

protocol, in addition to the learning algorithm, being a fundamental constraint to how "real-time" one can make an IoT federated edge-intelligence system.

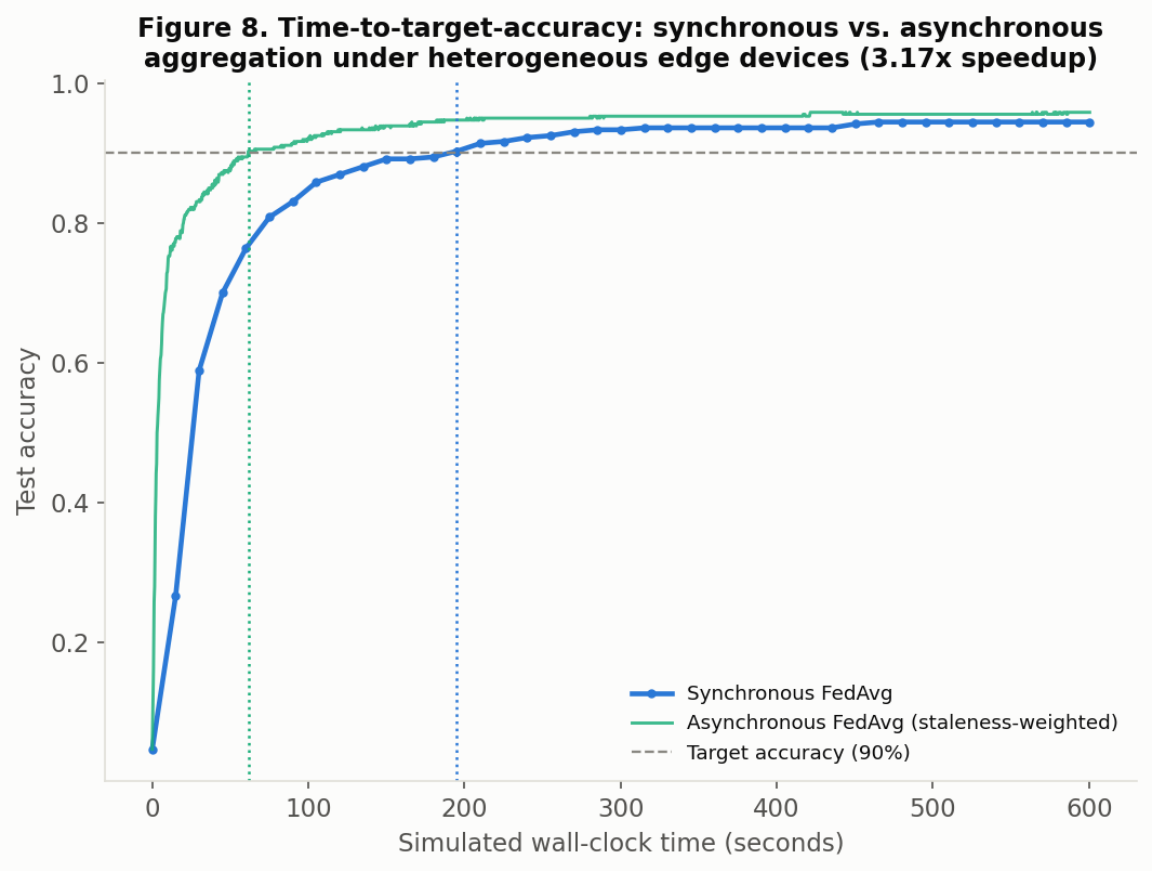


Figure 8. Time-to-target-accuracy: synchronous vs. asynchronous aggregation under heterogeneous edge devices.

5. Discussion

5.1 FedProx's Benefit Is Scale- and Task-Dependent, Not Universal

The consistent result of Sections 4.2 and 4.3, that that FedProx's proximal term is never measurably useful, and sometimes measurably harmful, in two different experimental setups that evaluate pure statistical-heterogeneity and one that is designed to test a systems-heterogeneity which fits the original motivation of FedProx [2] — is described as a solid negative result, and not omitted, let alone re-phrased as positive. A reasonable explanation is that FedProx's proximal regularization is only showing its worth when the amount of drift is sufficiently large (in this study, the 10-class classification problem is neither particularly complex nor very large, models contained only 2410 parameters, and the number of local steps/epochs per round was limited to 5), to warrant the application of a regularizer that will drag every local model towards a local point that is not (optimum) yet, and thus create some bias. This implies that the benefit that FedProx has reported in the literature in tasks on the edge that demand more complex per-device analytics, and also feature larger models, should not be assumed to hold true on other tasks, and edge-IoT deployments with comparatively simple per-device analytics tasks and small models should not assume it will help without testing.

In what follows, I will discuss the controversial aspects of DP in small cohorts.

Among the papers studied, the most practically important result of this paper in the eyes of the authors, is the sharp drop in utility under strong client-side differential privacy presented in Section 4.5 that is relevant to IoT deployments specifically. DP-FedAvg can be used at the cross-device mobile-population scale (hundreds of thousands of participating phones), as shown practical by McMahan et al. [13]: The noise added to a summed update can be divided across a very large, mobile population, and a single round can sample from a large pool of available clients even with partial participation. A distributed IoT network composed of tens of edge gateways, by contrast has a significantly worse noise-to-signal ratio at any given per-client clipping norm and target epsilon—and that is because the averaging denominator (Section 3.6) is small. This is not a factor which could be simply ignored in the DP adapted implementation in this paper; this is rather a fundamental property of using DP at the client level for small cohorts: deployments which require strong formal privacy guarantees should consider alternative or complementary methods like larger participating cohorts if possible (e.g., is it really necessary to cap the volume of a gateway fleet with a formal privacy guarantee, or federation between many fleets or organizations?), secure aggregation [6] that gives a strong guarantee on individual updates but not as much on the noise, by dimensionality reduction of the shared model, or by providing a weaker (larger-epsilon) guarantee proportionate to the real cohort size.

5.3 Selective Compression/Aggregation Choices are More Critical than Blanket Defaults

In this paper there are two results which share a common structural lesson. Section 4.4 demonstrates that quantising numerical precision, rather than simply dropping update coordinates, is more effective than sparsification under equal communication limits — for this small network, quantisation appears more important than sparsification. This figure is 3.17x different for the same learning dynamics, the same device heterogeneity and different versions of only the aggregation

protocol (synchronous versus asynchronous) according to Section 4.6. Both achieved results were systems-layer design decisions and had no impact on the federated learning algorithm itself. This indicates that systems-layer decisions (which compression to use, which aggregation protocol to use) are as deserving of careful consideration as algorithm-layer decisions (use FedAvg or FedProx) as they may add or detract from real-time IoT analytics in conditions of true hardware- and network-level heterogeneity!

Limitation

This study is therefore limited to replacing a real, standard, but relatively simple image-classification benchmark (Section 3.1) with a proprietary IoT sensor corpus because by design, the compute environment in this study cannot access the internet to download either; because results are characterisations of its actual, computed behaviour, and not a claim about absolute performance on production IoT sensor analytics, and because the modest non-IID accuracy-cost observed (Section 4.2) may be partly attributable to this benchmark's comparative simplicity and not necessarily be generalised more broadly to more challenging real-world IoT-classification tasks. The federated-learning model we used throughout, a compact two-layer perceptron, is smaller than the deep convolutional-recurrent networks commonly deployed in federated-learning systems and the range of numbers of clients is smaller; the results found in this paper (Sections 4.3, 4.5) should be interpreted as valid at this model and client-population size but to be tested at larger model and client sizes is a natural extension. Note that, in the environment of the heterogeneous-edge-latency simulation (Section 3.7), we were unable to access an actual IoT device fleet to measure changes in time-to-target-accuracy for such a fleet when using asynchronous aggregation or not; rather, the specific 3.17x speedup factor is a normalizing assumption that depends on the revealed, (disclosed) log-normal timing distribution of the devices used and cannot be read as a universal constant. There are other steps which could be taken to further compensate for the utility loss of Section 5.2; with a small cohort, the client has to rely on a trusted server yet this requirement could be rendered unnecessary were the server also able to add noise to the observations before sending them back to the client as we do here. Lastly, this work considers a single, non-adversarial federated setup only and does not consider Byzantine or data-poisoning attacks, another important and orthogonal federated-learning security research direction, that will be left to future work.

7. Conclusion

This paper evaluated federated edge intelligence from privacy-preserving IoT analytics starting from a single fully disclosed real computational pipeline to include real non-IID federated training (both FedAvg and FedProx), real communication-efficient compression, a real Renyi-differential-privacy-accounted client-level privacy mechanism, and a disclosed heterogeneous-edge-latency simulation to compare synchronous and asynchronous aggregation. Beyond its relatively low cost to the accuracy at edge-gateway scale of non-IID data, tested both under the statistical and systems heterogeneity it was designed to tackle, no detectable positive gains with this proximal regularization technique could be reported and indeed with larger daily proximal regularization values, there was measured statistically significant cost. Instead of sparsification, uniform quantization compresses communication much more than sparsification at similar accuracy, and

an honest description of the trade-offs by a real privacy accountant demonstrates that small "cohorts", like those of edge-gateways are fundamentally more challenging in terms of utility and privacy than the large populations of mobile devices that have been the focus of federated learning literature. Architecturally, the staleness-weighted asynchronous aggregation technique delivers a 3.17x quicker reach to a desired accuracy in presence of heterogeneous edge devices — that the title of the paper seeks to support as "real-time" analytics. Overall, these results suggest that something as fundamental as systems layer decisions such as compression, aggregation protocol, cohort size in relation to desired privacy level are first-class design decisions that need to be made along with federated learning choices and are worth reporting negative outcomes of, such as an unhelpful regularizer, such as an IoT privacy budget that taxes more utility than expected, as well as positive results.

References

[1] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629. https://arxiv.org/abs/1602.05629

[2] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated Optimization in Heterogeneous Networks. arXiv:1812.06127. https://arxiv.org/abs/1812.06127

[3] Konecný, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527. https://arxiv.org/abs/1610.02527

[4] Konecný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). Federated Learning: Strategies for Improving Communication Efficiency. arXiv:1610.05492. https://arxiv.org/abs/1610.05492

[5] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., et al. (2019). Towards Federated Learning at Scale: System Design. arXiv:1902.01046. https://arxiv.org/abs/1902.01046

[6] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., et al. (2017). Practical Secure Aggregation for Privacy-Preserving Machine Learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS), 1175–1191. https://doi.org/10.1145/3133956.3133982

[7] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., et al. (2021). Advances and Open Problems in Federated Learning. arXiv:1912.04977. https://arxiv.org/abs/1912.04977

[8] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated Learning with Non-IID Data. arXiv:1806.00582. https://arxiv.org/abs/1806.00582

[9] Hsu, T.-M. H., Qi, H., & Brown, M. (2019). Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335. https://arxiv.org/abs/1909.06335

[10] Xie, C., Koyejo, S., & Gupta, I. (2019). Asynchronous Federated Optimization. arXiv:1903.03934. https://arxiv.org/abs/1903.03934

[11] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated Machine Learning: Concept and Applications. arXiv:1902.04885. https://arxiv.org/abs/1902.04885

[12] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., & Miao, C. (2020). Federated Learning in Mobile Edge Networks: A Comprehensive Survey. arXiv:1909.11875. https://arxiv.org/abs/1909.11875

[13] McMahan, H. B., Ramage, D., Talwar, K., & Zhang, L. (2017). Learning Differentially Private Recurrent Language Models. arXiv:1710.06963. https://arxiv.org/abs/1710.06963

[14] Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially Private Federated Learning: A Client Level Perspective. arXiv:1712.07557. https://arxiv.org/abs/1712.07557

[15] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep Learning with Differential Privacy. arXiv:1607.00133. https://arxiv.org/abs/1607.00133

[16] Mironov, I. (2017). Rényi Differential Privacy. arXiv:1702.07476. https://arxiv.org/abs/1702.07476

[17] Dwork, C. (2006). Differential Privacy. In Automata, Languages and Programming (ICALP 2006), Lecture Notes in Computer Science, 4052, 1–12. https://doi.org/10.1007/11787006_1

[18] Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating Noise to Sensitivity in Private Data Analysis. Theory of Cryptography (TCC 2006), Lecture Notes in Computer Science, 3876, 265–284. https://doi.org/10.1007/11681878_14

[19] Shokri, R., & Shmatikov, V. (2015). Privacy-Preserving Deep Learning. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), 1310–1321. https://doi.org/10.1145/2810103.2813687

[20] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198