Proceedings of CONF-CIAP 2026 Symposium: Applied Mathematics and Statistics
DOI: 10.54254/2753-8818/2026.CH29902

Evaluating Local Update Strategies in FedAvg: Effects of Local Epochs and Client Participation on CIFAR-10

Xi Zeng

College of Letters and Science, University of California, Davis, USA
xizeng@ucdavis.edu

Abstract. This work studies federated learning on CIFAR-10 using a ResNet-18 classifier and three local update modes: plain Stochastic Gradient Descent (SGD), simulated Synchronous updates (Sync), and simulated Local updates (Local). An IID split provides a baseline to verify training and logging. The main experiments use a non-IID split and vary two factors while keeping all other hyperparameters fixed: the number of local epochs (E, tested at 2, 4, and 8) and the client participation rate (p, tested at 0.1, 0.2, 0.5, and 1.0). After each round, test accuracy, test loss, and per-round communication time (download, upload, aggregation) are recorded. Results show a consistent accuracy ranking across settings, with Local > Sync > SGD, most evident at high participation. Increasing local epochs from 2 to 4 and 8 preserves or slightly improves final accuracy and lowers per-round communication; the most stable communication profile occurs at E=8, with E=4 also markedly lower than E=2. Raising (dparticipation from 0.1 to 1.0 improves final accuracy and reduces round-to-round variance; at full participation, communication ordering is clearest, where Local lowest and Sync highest. A practical operating point emerges for this task and hardware: moderate-to-high participation with eight local epochs balances accuracy and communication.

Keywords: Federated learning, Federated averaging (FedAvg), Non-IID data, Communication efficiency

1. Introduction

Federated learning (FL) trains models without sharing raw data: data remain on clients while only model updates are exchanged with a central server, which is attractive for privacy-sensitive domains such as healthcare, finance, and mobile devices [1]. Among existing methods, Federated Averaging (FedAvg) is the most widely used baseline: clients perform several local steps of stochastic gradient descent (SGD), and the server aggregates the returned parameters by weighted averaging to produce a new global model [2]. Compared with synchronous distributed SGD, this design substantially reduces communication while enabling collaborative training.

Despite its popularity, FedAvg faces challenges on non-independent and identically distributed (non-IID) data, where client distributions differ. Local updates may point in different directions, and the aggregated model can drift away from any client optimum, a phenomenon often termed client drift [3]. Prior studies show that drift and system heterogeneity can slow or even destabilize convergence [4,5]. Even under IID partitions, FedAvg must balance stability against communication cost, highlighting the need to examine how local training strategies behave across both IID and non-IID regimes [6].

A key control is the number of local steps between aggregations. When this number equals one, the procedure resembles synchronous distributed SGD, which is stable but communication-intensive [6]. Increasing local steps reduces communication but can amplify drift under non-IID data [3,7]. These trade-offs make the design of local-update strategies central to federated optimization.

This work evaluates three local training strategies—standard SGD on a single client, a synchronous variant that averages every step, and a local-update variant that performs multiple steps before synchronization—on CIFAR-10 with a ResNet-18 backbone under IID and non-IID partitions. The study varies local epochs and participation rate while keeping other hyperparameters fixed, enabling a controlled comparison of communication cost, convergence behavior, and accuracy.

2. Method

2.1. Dataset preparation

CIFAR-10 is used in this study and is loaded with `torchvision.datasets [8]. CIFAR10`, keeping the native 32×32 resolution. Images stay in color; there is no grayscale conversion, resizing, or normalization. The classifier head outputs ten classes to match the label space. The training transform applies a random crop (size 32, padding 4), then a random horizontal flip, and finally converts images to tensors; the test transform performs only the tensor conversion. The dataset split is fixed at 50,000 training images and 10,000 test images, and evaluation always uses the global test set. No sample thumbnails are exported.

2.2. Federated learning-based ResNet-18

Federated learning is implemented with a round-based protocol. In each round, a subset of clients is sampled by the configured participation rate [9,10]. Each selected client trains locally and returns updated parameters with timing statistics. After aggregation, the global model is evaluated on the test loader, and per-round results are logged for analysis. This procedure optimizes the weighted global objective in Eq. (1).

Global objective:

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{N} F_k(w), \tag{1}$$

$$\text{where } \; F_k(w) = E_{((x,y)\sim\mathscr{D})} \; [l(f_w(x), y)]$$

Server aggregation (FedAvg):

$$w^{(r+1)} = \sum_{k\in\mathscr{S}^{(r)}} \frac{n_k}{\sum_{j\in\mathscr{S}^{(r)}} n_j} \, w_k^{(r+1)} \tag{2}$$

Server-side aggregation follows weighted averaging by local sample counts (Eq. (2)). Three local training modes share a common interface and differ only in how often parameters are updated. The modes are denoted SGD, Sync, and Local. Two internal controls determine the number of sub-batches processed within a global batch and the frequency of parameter updates. A launcher runs each mode under comparable settings and records logs for downstream plotting.

The backbone is ResNet-18 initialized from scratch. Its final classifier is a linear layer with ten outputs to match CIFAR-10. The model is moved to the selected compute device before training and evaluation.

Local optimization uses SGD with cross-entropy loss. Optional gradient clipping can be applied immediately before each update to improve stability. Mini-batches are transferred efficiently to the device to avoid unnecessary stalls.

Within a global batch, the computation is simulated across multiple sub-batches to control the update schedule: in SGD mode, one parameter update is performed per global batch (and this reduces to the same behavior when only a single sub-batch is used); in Sync mode, gradients from all sub-batches are accumulated and then applied once, emulating synchronous averaging; in Local mode, an update is performed for each sub-batch, producing more frequent local steps between synchronizations.

Two strategies define client datasets. In the IID setting, samples are evenly divided across clients. In the non-IID setting, client proportions are drawn per class from a Dirichlet distribution with concentration α, samples are assigned accordingly, and the order within each client is shuffled. Each client builds a data loader over its subset with shuffling enabled and a configurable number of worker threads; when workers are used, persistent workers and limited prefetching keep throughput stable, and page-locked memory is disabled on CPU/MPS to match the device characteristics.

Under the non-IID setting, two experimental factors are varied: the participation rate (the fraction of clients selected per round) and the local training depth (the number of local epochs per round). Exact values are reported in the Results section.

2.3. Hyperparameter configuration

Experiments use PyTorch with torch and torchvision. Training and evaluation run on CPU or Apple MPS, chosen by a device setting. Local optimization compares three SGD-based modes under the same interface: SGD, Sync, and Local. The optimizer is stochastic gradient descent with a tunable learning rate, momentum, and weight decay. The loss is cross-entropy. Optional gradient clipping can be applied immediately before each parameter update to improve stability.

The training batch size is set once and used for all clients. The test loader uses a batch size that is exactly twice the training batch and turns off shuffling to keep evaluation stable across rounds. Client data loaders enable shuffling for training and allow a configurable number of worker threads; when workers are used, persistent workers and a small prefetch are enabled to stabilize throughput. Pinned memory is disabled on CPU/MPS.

The learning rate is specified globally, and mode-specific values can be supplied when needed; if not supplied, all modes use the global setting. After each federated round, the server evaluates on the global test set and logs test accuracy and test loss. The same log records total and per-phase communication time, which enables consistent comparisons across SGD, Sync, and Local.

For the ablations on participation rate and local epochs, all other hyperparameters within each comparison remain fixed. This includes batch size, learning-rate policy, optimizer and loss settings, evaluation loader configuration, and device selection.

2.4. Training protocol, logging, and reproducibility

Per-round communication time:

$$\text{comm\_ms} = \text{down\_ms} + \text{up\_ms} + \text{agg\_ms} \tag{3}$$

Communication is measured as wall-clock time (see Eq. (3)). Each round follows one fixed order: sampling -> local training -> aggregation -> evaluation. Clients are sampled by the configured participation rate. Selected clients run local training and return updated parameters and timing statistics. The server aggregates on device and evaluates the aggregated model on the global test loader. This order does not change across experiments or modes.

Communication is measured on wall-clock time. For every round, the logger records the total communication time comm_ms and its three components: download, upload, and aggregation. Wall-clock time for the round is also recorded. Model quality is tracked by test loss and test accuracy on the global test set after aggregation. All values are appended to a CSV with stable fields: round, time_sec, val_loss, val_acc, comm_ms, down_ms, up_ms, agg_ms.

All experiments run with three concurrent processes to shorten wall-clock time, and this concurrency is kept constant across runs. Reproducibility is ensured by a fixed random seed (42), explicit device selection (CPU or Apple MPS), and stable data-loading settings. Client loaders shuffle during training and keep test loading deterministic. When worker threads are used, persistent workers and a small prefetch (factor 2) stabilize throughput; pinned memory is disabled on CPU/MPS. The test loader uses a batch size equal to twice the training batch and disables shuffling so that evaluations remain comparable round to round.

All non-IID comparisons share the same data partition with a fixed Dirichlet α and the same seed (42) under this uniform concurrency. Only the participation rate or the number of local epochs is varied; the logging format remains identical across runs.

3. Results and discussion

3.1. The results of baseline (IID)

An IID partition sets the reference (local epochs = 2, participation = 1.0). Under identical hyperparameters, the three modes—SGD, Sync, and Local—show smooth accuracy gains over rounds. By round 60, local reaches approximately 82–83%, while SGD and Sync plateau near 77–78% (Fig. 1). Communication cost per round is markedly different: Local stays around 300–400 ms with small variance, whereas SGD and Sync typically lie near 1.8 to 2.6 s, with occasional spikes approaching 4 s (Fig. 2). These results confirm a stable pipeline and provide the scale against which the non-IID studies on local epochs (E) and participation (p) are compared.

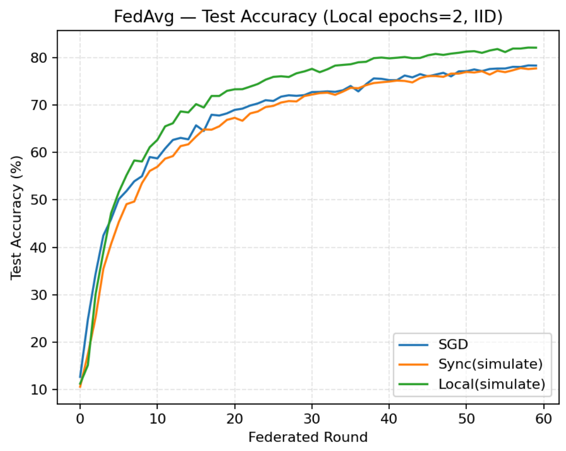

Figure 1. Accuracy vs round (IID; local epochs = 2; participation = 1.0)

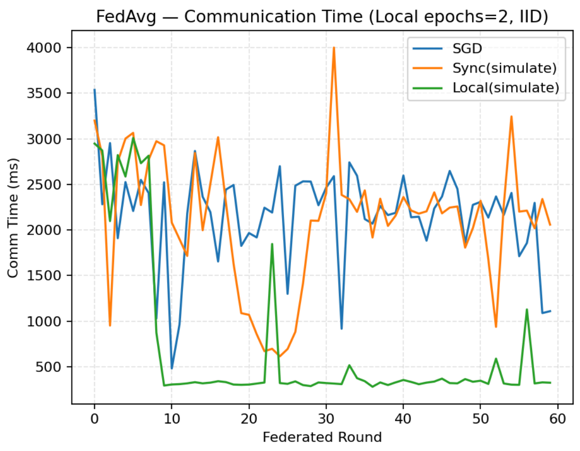

Figure 2. Communication time vs round

3.2. Effect of different local epochs under non-IID

In this subsection, the setting use a non-IID partition with a fixed participation rate. Hyperparameters are kept constant across runs. The comparison varies the number of local epochs $E \in \{2,4,8\}$.

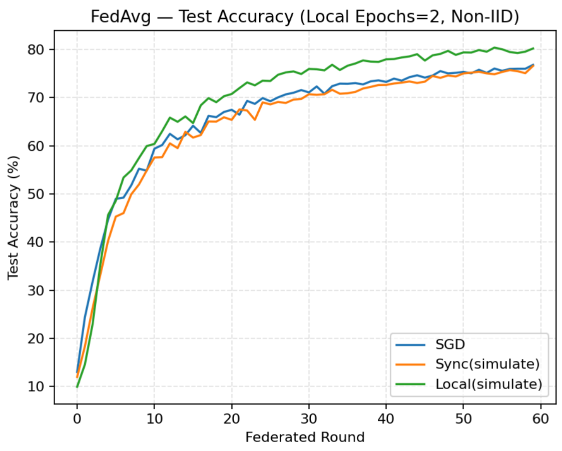

Figure 3. Test accuracy vs round under a non-IID partition with a fixed participation rate. Panels show local epochs E=2, curves: SGD, Sync, Local

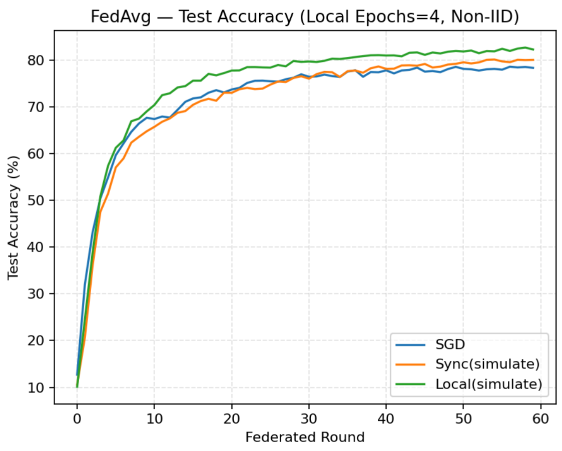

Figure 4. Test accuracy vs round under a non-IID partition with a fixed participation rate. Panels show local epochs E=4, curves: SGD, Sync, Local

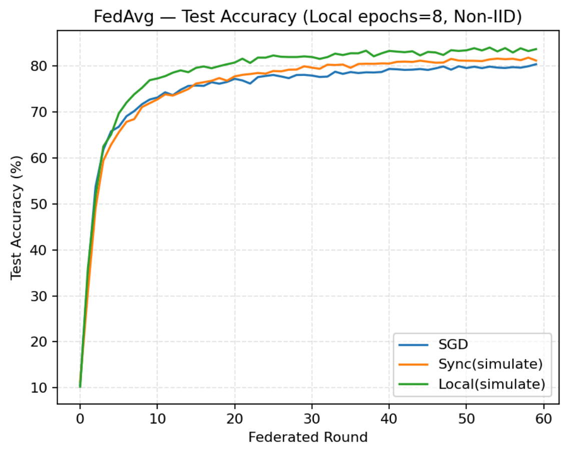

Figure 5. Test accuracy vs round under a non-IID partition with a fixed participation rate. Panels show local epochs E=8, curves: SGD, Sync, Local

For all three values of E, the local curve stays above the others. With E=2, the final accuracies of SGD and Sync are very close (Fig. 3). With E=4, a small gap appears between them while both improve over E=2 (Fig. 4). With E=8, the difference becomes clearer, and the final ordering is Local > Sync > SGD (Fig. 5). These trends show that larger E maintains or slightly improves final accuracy while preserving the same relative ranking across modes.

Communication time per round changes noticeably with E. With E=2, all modes lie in the multi-second range with noticeable variability; SGD shows the largest spikes (occasionally very high), Sync is moderate, and Local is the lowest but still around a couple of seconds on average. With E=4, the separation becomes clear: Sync drops to a low and stable sub-second band, Local remains low but higher than Sync (around the one-second level), while SGD stays much higher and exhibits several large peaks. With E=8, the curves remain in a generally low band; Local is the most stable, Sync is also low, and SGD presents sporadic peaks (Fig. 6, Fig. 7 and Fig. 8).

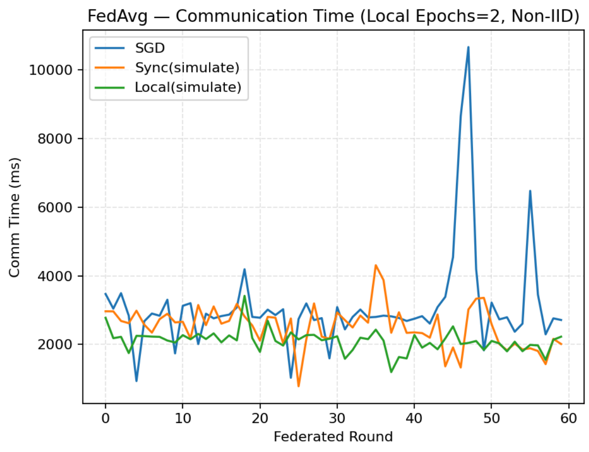

Figure 6. Per-round communication time vs round under the same setting. Panels show local epochs E=2, curves: SGD, Sync, Local

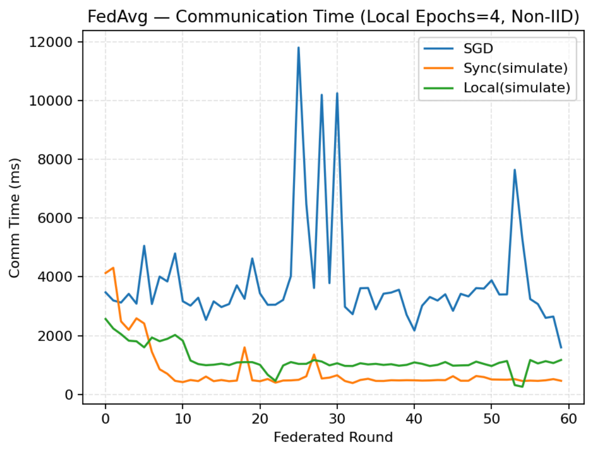

Figure 7. Per-round communication time vs round under the same setting. Panels show local epochs E=4, curves: SGD, Sync, Local

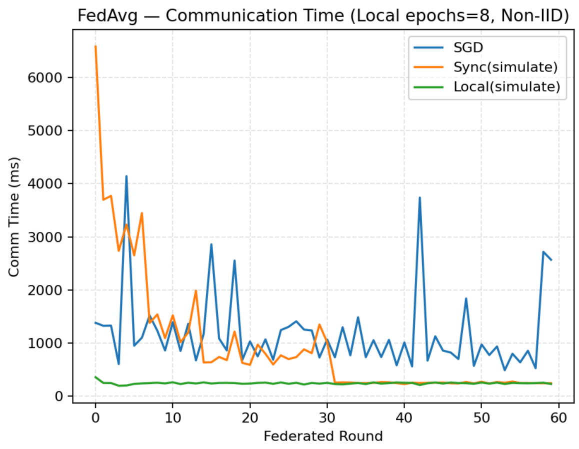

Figure 8. Per-round communication time vs round under the same setting. Panels show local epochs E=8, curves: SGD, Sync, Local

Therefore, under this non-IID split with fixed participation, increasing E reduces per-round communication while keeping, and in this setup slightly improving, final accuracy; across all E, the consistent ranking is Local > Sync > SGD.

3.3. Effect of different participation under non-IID

For this subsection, the setting uses a non-IID partition and fixed local epochs. Only the participation rate varies ($p \in \{0.1,0.2,0.5,1.0\}$), and all other hyperparameters remain the same. For the accuary in this part, at p=0.1 the curves are noisy and the final accuracies of the three modes are close, with no clear winner. At p=0.2 the trajectories become more stable; local is slightly higher, while SGD and Sync stay close to each other. At p=0.5 all three improve further and the separation becomes clearer, with Local on top and sgd/sync a bit lower. At p=1.0 the ranking is most evident: Local > Sync > SGD and the curves are smoother (see Fig. 9, Fig. 10, Fig. 11 and Fig. 12). Overall, increasing p raises the final accuracy and reduces variance across rounds.

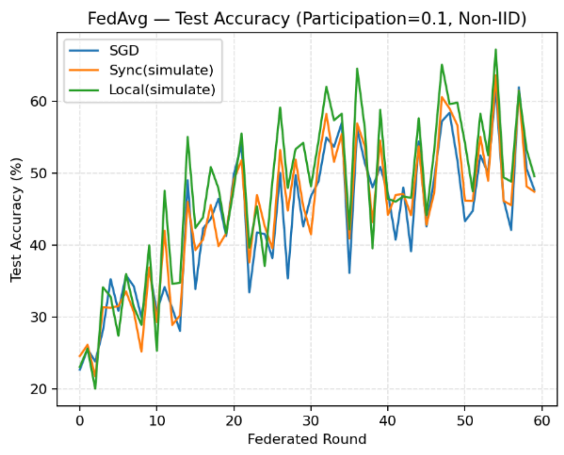

Figure 9. Test accuracy vs round under a non-IID partition with fixed local epochs. Panels show participation rates p=0.1. curves: SGD, Sync, Local

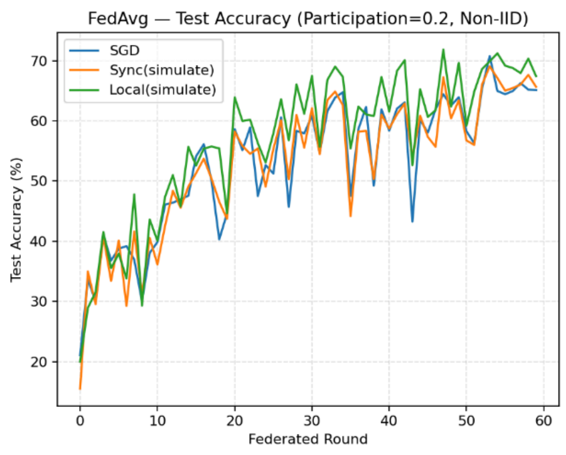

Figure 10. Test accuracy vs round under a non-IID partition with fixed local epochs. Panels show participation rates p=0.2. curves: SGD, Sync, Local

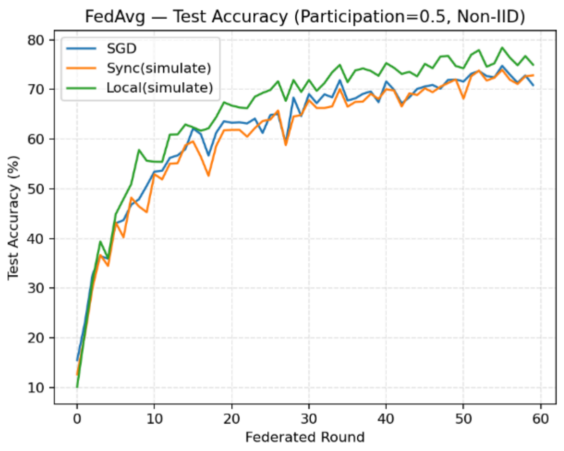

Figure 11. Test accuracy vs round under a non-IID partition with fixed local epochs. Panels show participation rates p=0.5. curves: SGD, Sync, Local

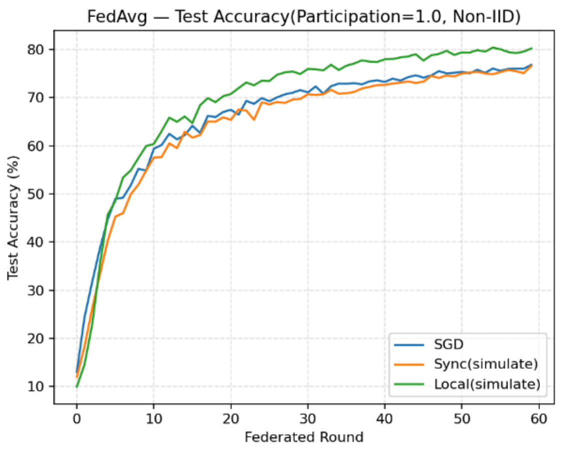

Figure 12. Test accuracy vs round under a non-IID partition with fixed local epochs. Panels show participation rates p=1.0. curves: SGD, Sync, Local

Then about the communication time, at p=0.1 the three modes show large fluctuations with frequent spikes. At p=0.2 and p=0.5 the local curve stays in a low, stable band (hundreds of milliseconds), SGD exhibits intermittent high peaks, and Sync remains moderate with occasional spikes. At p=1.0 the difference across modes is most pronounced: Sync reaches the largest and most variable times (often multi-second), SGD is lower but still spiky, and Local remains the lowest band overall (see Fig. 13, Fig. 14, Fig. 15 and Fig. 16). These plots describe per-round cost; cumulative time to reach a target accuracy can be read by aligning rounds across the accuracy and communication figures.

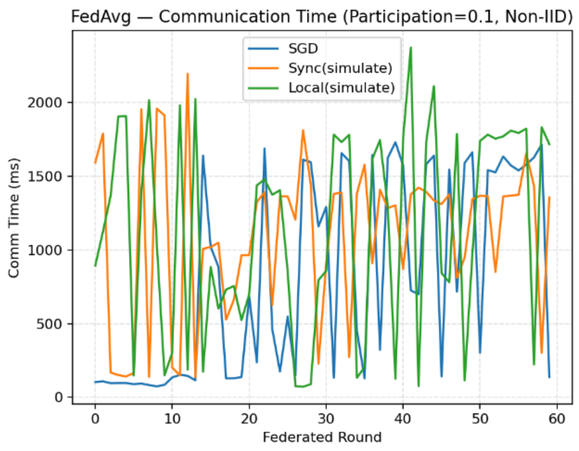

Figure 13. Per-round communication time vs round under the same setting as Fig. 9-12. Panels show participation rates p=0.1, curves: SGD, Sync, Local

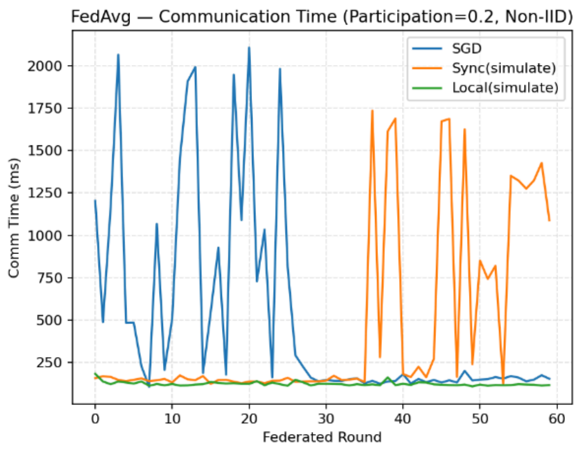

Figure 14. Per-round communication time vs round under the same setting as Fig. 9-12. Panels show participation rates p=0.2, curves: SGD, Sync, Local

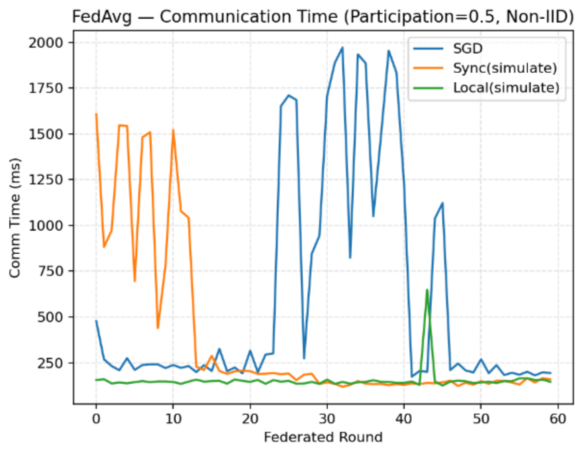

Figure 15. Per-round communication time vs round under the same setting as Fig. 9-12. Panels show participation rates p=0.5, curves: SGD, Sync, Local

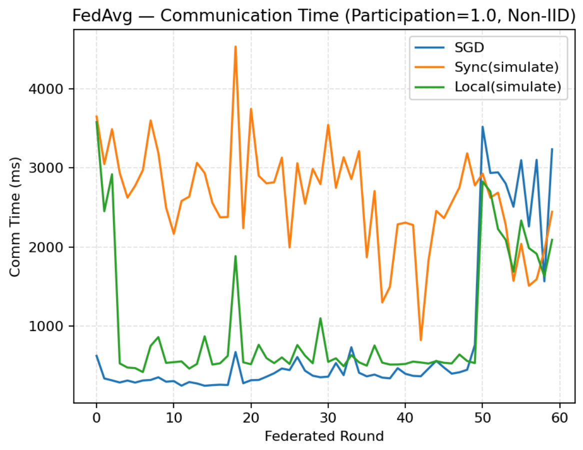

Figure 16. Per-round communication time vs round under the same setting as Fig. 9-12. Panels show participation rates p=1.0, curves: SGD, Sync, Local

In conclusion, under non-IID data with fixed local epochs, higher participation improves accuracy and stability. Per-round communication shows a clear mode ordering at p=1.0 (Local lowest, Sync highest), while at lower p the cost is smaller but more irregular due to spikes. The accuracy–communication balance therefore depends on the chosen p: moderate to high participation provides better accuracy with predictable cost, as shown by the paired curves.

3.4. Discussion and analysis

Across all settings, the three modes follow a consistent accuracy ranking, with Local above Sync and SGD. Under the non-IID split, increasing local epochs from 2 to 4 and 8 maintains or slightly improves final accuracy and moves the per-round communication into a lower, more stable range. With the updated runs, the E=4 curves show a clear separation in communication time: Sync forms a low and stable sub-second band, local remains low, and SGD is the highest with sporadic peaks. At

E=8 the communication curves stay generally low; Local is the most stable, while sgd still exhibits occasional spikes.

Varying participation shows a complementary trend. As p increases from 0.1 to 1.0, the accuracy trajectories become smoother and end higher, and the final ordering stabilizes at Local > Sync > SGD. The communication profiles are most distinct at p=1.0: Sync has the largest and most variable times, SGD is lower but still spiky, and Local is the lowest and most stable.

Taken together, these results indicate a practical operating region: moderate-to-high participation combined with E=4 offers a favorable balance—high and stable accuracy with low per-round communication—while preserving the same method ranking observed throughout.

4. Conclusion

This study evaluated federated learning on CIFAR-10 with a ResNet-18 backbone under IID and non-IID splits, comparing three local training strategies: SGD, Sync, and Local. The results show a consistent accuracy ranking—Local > Sync > SGD—most evident at high participation. Increasing local epochs from 2 to 4 and 8 preserved or slightly improved accuracy while lowering per-round communication, with the most stable profiles at E=8. Raising participation from 0.1 to 1.0 improved final accuracy and reduced variance; at p=1.0, communication time clearly ordered across modes (Local lowest, Sync highest). With a fixed seed, uniform logging, and matched settings, these findings are directly comparable. A practical operating point emerges: moderate-to-high participation with E=4 balances accuracy and communication for this task and hardware, providing a solid baseline for future extensions.

References

[1] P. Kairouz et al., “Advances and Open Problems in Federated Learning, ” arXiv preprint arXiv: 1912.04977, 2021.
[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data, ” arXiv preprint arXiv: 1602.05629, 2017.
[3] A. Khaled, K. Mishchenko, and P. Richtárik, “First Analysis of Local GD on Heterogeneous Data, ” arXiv preprint arXiv: 1909.04715, 2019.
[4] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and S. A. T. Suresh, “SCAFFOLD: Stochastic Controlled Averaging for Federated Learning, ” arXiv preprint arXiv: 1910.06378, 2020.
[5] T. Li, A. K. Sahu, V. Smith, and V. Talwalkar, “Federated Optimization in Heterogeneous Networks, ” arXiv preprint arXiv: 1812.06127, 2020.
[6] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, “Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Parallel SGD, ” arXiv preprint arXiv: 1705.09056, 2017.
[7] S. U. Stich, “Local SGD Converges Fast and Communicates Little, ” arXiv preprint arXiv: 1805.09767, 2019.
[8] A. Krizhevsky and G. Hinton, Learning Multiple Layers of Features from Tiny Images, 2009.
[9] L. Li, Y. Fan, M. Tse, and K. Y. Lin, “A review of applications in federated learning, ” Computers & Industrial Engineering, vol. 149, p. 106854, 2020.
[10] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, “A survey on federated learning: challenges and applications, ” International Journal of Machine Learning and Cybernetics, vol. 14, no. 2, pp. 513–535, 2023.