Federated Learning for Non-IID Data: Comparative Study of Optimization Algorithms and Initialization Strategies

Francesco Ceccucci, Alexis Do, Veit Weidinger
CS-439 Optimization for Machine Learning, EPFL, Switzerland

***Abstract*—In federated learning (FL), multiple clients collaboratively train a global model without sharing local data. This approach has emerged as a promising solution for privacy-sensitive domains such as healthcare, where raw data cannot be centralized. However, federated optimization remains challenging under non-independent and identically distributed (non-IID) conditions, where client data distributions significantly differ. Our work evaluates and compares under realistic non-IID scenarios generated via Dirichlet sampling multiple federated optimization algorithms including: FedAvg, FedProx, FedAvgM, and FedYogi. We investigate the impact of initialization strategies such as shared global datasets and warmup training on model convergence and accuracy. Our results show that careful selection of warmup epochs and shared dataset fractions can substantially improve convergence speed and final performance, especially in extreme non-IID settings (Dirichlet $\alpha \approx 0.001$).**

I. Introduction

Federated learning enables decentralized training over multiple devices or institutions while preserving privacy by keeping raw data local[1]. This paradigm is particularly relevant in healthcare, where sensitive patient data must remain confined to individual clinics or hospitals. Federated learning offers advantages in terms of data security, while still enabling collaborative model development.

However, federated optimization is fundamentally different from centralized training. The absence of data sharing leads to statistical heterogeneity across clients, a critical challenge in practice. In realistic scenarios such as digit recognition with MNIST, the digit classes can be distributed unequally across users, mimicking hospital-specific biases in patient populations. This non-IID nature results in poor generalization and unstable training.

To address this, researchers have proposed variants of standard federated algorithms. Our study investigates the performance in non-IID environments of four such algorithms: FedAvg[2], FedAvgM[3] (with server-side momentum), FedProx [4] (with proximal regularization), and FedYogi[5] (adaptive learning rate optimization). Furthermore, we evaluate the benefits of introducing a small globally shared dataset and model warmup strategies, which aim to stabilize and accelerate convergence. This study provides a practical foundation for improving federated learning deployments in real-world, heterogeneous environments.

II. Methods and Experimental Design

A. Algorithm Description

As stated, we use four federated optimization algorithms in our experiments:

FedAvg 1 Each round the server sends the current model to a sample of clients; they train locally for a few epochs and send the new weights back. The server simply averages those weights.

FedAvgM 2 Identical client workflow to FedAvg, but the server blends the incoming updates with a running "momentum" average.

FedProx 3 Clients train almost as in FedAvg, but their loss function includes a gentle pull toward the global model. That small regularization term keeps local models from drifting too far.

FedYogi 4 After collecting client updates, the server adapts the learning rate for every model parameter based on past update history (a Yogi-style variant of Adam).

B. Dataset and Preprocessing

We use the MNIST dataset [6], a collection of grayscale handwritten digit images (28×28), with 10 classes. To simulate non-IID distributions across clients, we adopt Dirichlet partitioning [3] with varying concentration parameters (α). Smaller α values lead to stronger client-specific label imbalance, simulating realistic heterogeneity. The shared dataset, if used, is randomly sampled from the global distribution and made available to all clients at the start of training.

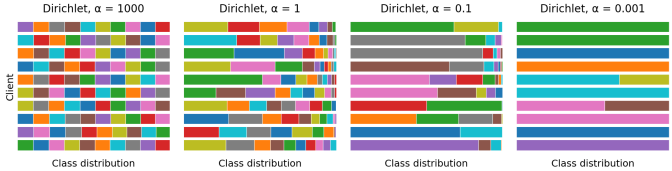


Fig. 1: Class distribution across 10 clients for different Dirichlet concentration values (α). Lower α leads to more heterogeneous and imbalanced class distributions, simulating stronger non-IID conditions.

C. Federated Setup

We implemented our federated learning setup using the Flower framework [7] with PyTorch Lightning [8] for model training. The pipeline consists of a central server and multiple simulated clients, each performing local training on non-IID data. Model updates are aggregated by the server using algorithms like FedAvg or FedYogi. Experiments are configured using Hydra [9] YAML files for flexible and reproducible parameter sweeps.

We simulate FL with 10 clients. Each round, a fraction (e.g., 10% or 50%) of clients is selected to perform local training using mini-batch stochastic gradient descent. Key hyperparameters include Dirichlet α (degree of non-IIDness, e.g., 1000–0.001), local epochs E (local training passes per client, e.g., 1–5), client fraction C (fraction of clients per round, e.g., 0.1, 0.5), warmup epochs (initial global pretraining epochs), and share fraction (proportion of centrally collected shared data).

D. Model Architecture

We use a compact convolutional neural network (CNN) designed for efficient training in federated settings, consisting of two convolutional layers (32 and 64 filters, 3×3 kernels), a 2×2 max-pooling layer, dropout layers (0.25 after convolutional blocks, 0.5 after the first fully connected layer), a fully connected layer (128 units, ReLU), and a final output layer with 10 units (MNIST classes). Despite its small size ($\approx$ 0.99M parameters), when training the model we achieved high accuracy on MNIST ($\sim$ 0.99) and serves as a suitable baseline for computationally constrained federated environments.

E. Global Sharing Strategy

To alleviate non-IID effects, we introduce a globally shared dataset G [10]:

1) Centralize a small sharing fraction $\beta = |G|/|D|$ of the total client data in the cloud, where D represents the total data from the clients.
2) Ensure G is class-balanced by sampling uniformly per class.
3) Distribute only a fraction α_{dist} of G to each client at for their local model.

Clients augment their private data with their $\alpha_{\text{dist}} * G$ samples from G during local training. In real-world scenarios, each client possesses only their local data, and a global IID training set is typically unavailable at the outset. However, the concept of "global sharing" in FL assumes that a small, curated subset of data can be centrally collected and used to initialize training. In healthcare, G could consist of anonymized, population-level data that poses no privacy risk but provides common feature coverage across clients. This shared data serves a dual purpose: (1) as an initial warm-up phase to align local models and (2) to be mixed with local updates to reduce class distribution skewness.

F. Warmup Training

We first *warmup* a global model on G for E_{warm} epochs:

$$w^0 \leftarrow \arg\min_w \frac{1}{|G|} \sum_{(x,y)\in G} \ell(f_w(x), y).$$

The resulting parameters w^0 are used to initialize the FL process. This pretraining should improve initial feature representations and reduce divergence across clients. After warmup epochs, local updates are resumed, improving gradient alignment and reducing inter-client model divergence.

III. Results

A. Experiment: Sensitivity to Hyperparameters and Algorithm Choice

We evaluated multiple configurations varying the number of local epochs ($E = 1,\ 5$), data heterogeneity ($\alpha = 1000,\ 1,\ 0.1,\ 0.001$), client participation rate ($C = 0.1,\ 0.5,\ 1.0$), and aggregation algorithm (*FedAvg*, *FedAvgM*, *FedProx*, *FedYogi*). Local models were trained with the Adam optimizer (learning rate 0.001) for 50 communication rounds per configuration. Each configuration was run with two different seeds, averaging results or discarding abnormal runs for stability. Figure 3 summarizes the outcomes. Convergence behaviors of selected methods are shown in Figure 4.

B. Experiment: Global sharing

In our second experiment, we fixed a highly non-IID setting that, without any shared data, yields near–random performance: local training epochs per round were set to $E = 5$, Dirichlet heterogeneity was $\alpha = 10^{-3}$, the client participation fraction was $C = 0.1$, and we tested two aggregation strategies, namely *FedAvg* and *FedAvgM*.

We then augmented this baseline with our global-sharing strategy by varying two key parameters: the number of warmup epochs on the shared dataset, $E_{\text{warm}} \in \{0, 1, 10\}$, and the global share fraction, $\beta = |G|/|D| \in \{0.05, 0.10\}$; the per-client share sampling fraction was fixed at $\alpha_{\text{dist}} = 0.5$.

Aside from these additions, all other settings (optimizer, learning rate, number of rounds, random seed protocol, etc.) matched those of Experiment 1. Figure 2 shows the obtained learning curves.

IV. Discussion

A. Convergence Behavior

FedAvg, FedAvgM, FedProx, and FedYogi each collapse to five recurring curve types ①–⑤ in Fig. 2:
Fast & stable—IID or mild skew ($\alpha = 1000,\ 1$), high client fraction, moderate E: $\geq 95\%$ accuracy within ≤ 10 rounds.
Gradual but steady—slight non-IID: smooth climb to to test accuracy of 85–90% by round 50.
Slow/partial—moderate heterogeneity or mismatched E, C: plateau around 60–80%.
Poor/oscillatory—high skew or large E: 40–60% with visible loss swings.

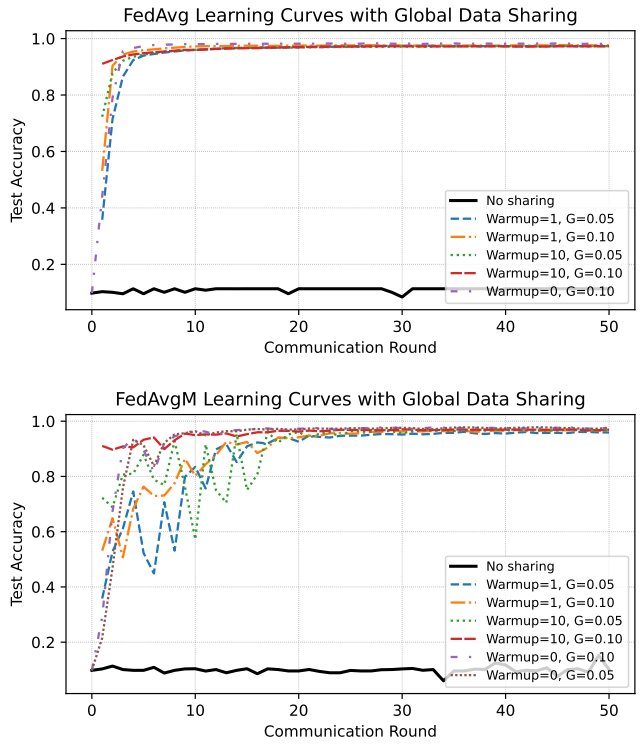


Fig. 2: **(Above)** FedAvg; **(Below)** FedAvgM. Learning curves (test accuracy vs. communication rounds) for various warm-up epochs and global share fractions G. "No sharing" (black solid) corresponds to classical FedAvg/FedAvgM on purely Dirichlet non-IID MNIST. All other curves combine each client's private data with G uniformly sampled global data ($\beta = 5\%$ or 10% of total), and a warm-up model trained for $E_{\text{warm}} \in \{0, 1, 10\}$ epochs.

Failure/divergence—extreme skew ($\alpha = 0.001$) or bad hyper-parameters: near-chance accuracy.

We use FedAvg as the baseline. When we compare methods, FedProx is the most resilient, often lifting class ④⑤ runs into class ③ (e.g. $\alpha = 0.001$, $E = 5$ reaches $\sim 50\%$). FedYogi can jump from class ② to class ⑤ when $E = 5$, $C = 1.0$. Lowering C sometimes regularises training—FedAvgM shifts from class ④ to ② when C drops from 0.5 to 0.1 under moderate skew. Interestingly, lowering the client fraction sometimes regularises training—for example, FedAvgM moves from oscillatory to stable when C drops from 0.5 to 0.1 under moderate skew.

B. Final Accuracy & Hyperparameter Dependencies

Heat-map extremes mirror these classes. Accuracy stays near $\sim 98\%$ for $\alpha \geq 1$ (classes ①–②) but collapses below 10% in class ⑤. Increasing local epochs helps mild skew (FedProx: $75.9\% \rightarrow 96.6\%$ when $E : 1 \rightarrow 5$, $\alpha = 0.1$) yet hastens divergence in severe skew (FedYogi, $E = 5$, $C = 1.0$). Client fraction mostly affects speed, though partial participation can raise the plateau (FedProx, $\alpha = 0.1$, $E = 5$: 98.7% at $C = 0.1$ vs. 83.0% at $C = 1.0$). Data heterogeneity remains the dominant limiter; proximal regularisation and prudent tuning of E and C can shift runs upward within the five-class hierarchy.

C. Impact of global sharing

Figure 2 clearly illustrates the impact of a small globally shared dataset on federated learning with highly skewed, Dirichlet-partitioned MNIST ($\alpha = 10^{-3}$). Several key observations emerge:

- **Global sharing drastically improves accuracy.** Without any shared data (black solid), both FedAvg and FedAvgM remain stuck near random-guess performance ($\approx 10\%$). Injecting just 5% of a balanced global set G lifts test accuracy into the $70-80\%$ range within a handful of rounds, ultimately converging above 90%.
- **Warm-up epochs accelerate convergence.** A single warm-up epoch on G (dashed/dash-dot) already produces a rapid initial climb, but taking $E_{\text{warm}} = 10$ epochs (dotted) shifts the curve right up to $85-90\%$ at round 1 and nearly eliminates the "ramp-up" phase.
- **Larger global fraction smooths and lifts final accuracy.** Comparing $G = 5\%$ vs. $G = 10\%$, the latter not only reduces variance in early rounds but also nudges the asymptotic test accuracy higher by a couple of percentage points—at the expense of more centralized data.

These findings underscore a practical *trade-off* in real-world federated deployments (e.g. multi-site medical imaging): one may choose small G to preserve data locality, then compensate with more warm-up epochs for rapid convergence, or vice versa. Ultimately, tuning $\beta = |G|/|D|$ and E_{warm} allows practitioners to balance privacy, communication cost, and model performance to suit their application constraints.

V. Summary

We tested several algorithms and evaluated their convergence behavior; noticeable differences in convergence speed can be observed. These results, should be interpreted with caution: in [11] various algorithms were compared, and besides the number of server rounds one should also consider the computational cost, which can differ drastically across methods. In practice, one must balance convergence speed, client resource constraints, and privacy requirements when selecting an optimizer. Crucially, our experiments demonstrate that a small, globally shared dataset G, combined with a lightweight warm-up phase, can restore convergence even in configurations that otherwise fail completely. With as little as 5% of the total data held out as a balanced set, and only a handful of warm-up epochs, all four methods recover from near–random accuracy to over 90% in just a few rounds. This simple extension thus provides a practical lever to salvage federated training under extreme non-IID conditions—transforming "unsolvable" setups into fully convergent ones without heavy communication or computational overhead.

REFERENCES

[1] K. B. *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, 2017, pp. 1175–1191.

[2] H. B. M. *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2017, proc. AISTATS 2017. [Online]. Available: https://arxiv.org/abs/1602.05629

[3] T. M. H. H. *et al.*, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019. [Online]. Available: https://arxiv.org/abs/1909.06335

[4] T. L. *et al.*, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2020, introduces the FedProx algorithm; MLSys 2020. [Online]. Available: https://arxiv.org/abs/1812.06127

[5] S. J. R. *et al.*, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2021, introduces FedYogi (and FedAdam, FedAdagrad); ICLR 2021. [Online]. Available: https://arxiv.org/abs/2003.00295

[6] Y. L. *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] D. J. B. *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020, version 5, last revised 5 Mar 2022. [Online]. Available: https://arxiv.org/abs/2007.14390

[8] W. F. *et al.*, "Pytorch lightning: The lightweight PyTorch wrapper for high-performance ai research," Software, Zenodo, Version 0.7.6, 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3828935

[9] O. Yadan, "Hydra – a framework for elegantly configuring complex applications," GitHub Repository, 2019. [Online]. Available: https://github.com/facebookresearch/hydra

[10] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018. [Online]. Available: https://arxiv.org/abs/1806.00582

[11] G. Cheng, K. Chadha, and J. Duchi, "Federated asymptotics: A model to compare federated learning algorithms," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. Ahumada, M. Belkin, P. Liang, and T. Ma, Eds., vol. 206. PMLR, 23–29 Jul 2023, pp. 4332–4358. [Online]. Available: https://proceedings.mlr.press/v206/cheng23b.html

APPENDIX

TABLE I: Notation used in the algorithm formulas

Symbol	Description
w_t	Global model after round t
S_t	Set of clients selected in round t ($\lvert S_t\rvert{=}m$)
n_k	Number of local examples on client k
n	$\sum_{k\in S_t} n_k$ (total examples in S_t)
$w_{t+1}^{(k)}$	Model returned by client k after local training
g_t	Aggregated model update at the server
$m_t,\ v_t$	First/second moment buffers (server)
η_c	Client learning rate (local SGD)
$\lambda,\ \eta$	Server learning rates (FedAvgM / FedYogi)
$\beta,\ \beta_1,\ \beta_2$	Momentum / decay coefficients
μ	Proximal term coefficient (FedProx)
τ	Small constant to avoid divide-by-zero (FedYogi)
E	Local epochs per client per round

A. FedAvg

$$w_{t+1} \;=\; \sum_{k\in S_t} \frac{n_k}{n}\, w_{t+1}^{(k)} \tag{1}$$

B. FedAvgM

$$\begin{aligned} g_t &= w_t \;-\; \sum_{k\in S_t} \frac{n_k}{n}\, w_{t+1}^{(k)}, \\ m_t &= \beta\, m_{t-1} + (1-\beta)\, g_t, \\ w_{t+1} &= w_t - \lambda\, m_t. \end{aligned} \tag{2}$$

C. FedProx

$$w_{t+1}^{(k)} \;=\; \arg\min_{w} \Big[F_k(w) \;+\; \tfrac{\mu}{2}\,\|w - w_t\|^2 \Big], \tag{3}$$

where each client k solves the objective locally, the server then aggregates the $w_{t+1}^{(k)}$ with Eq. (1).

D. FedYogi

$$\begin{aligned} g_t &= w_t - \sum_{k\in S_t} \frac{n_k}{n}\, w_{t+1}^{(k)}, \\ m_t &= \beta_1 m_{t-1} + (1-\beta_1)\, g_t, \\ v_t &= v_{t-1} - (1-\beta_2)\, \operatorname{sign}\!\big(v_{t-1} - g_t^2\big)\, g_t^2, \\ w_{t+1} &= w_t - \eta\, \frac{m_t}{\sqrt{v_t} + \tau}. \end{aligned} \tag{4}$$

Client phase (all algorithms): every selected client runs E local SGD epochs with a learning rate η_c before sending back its model $w_{t+1}^{(k)}$.

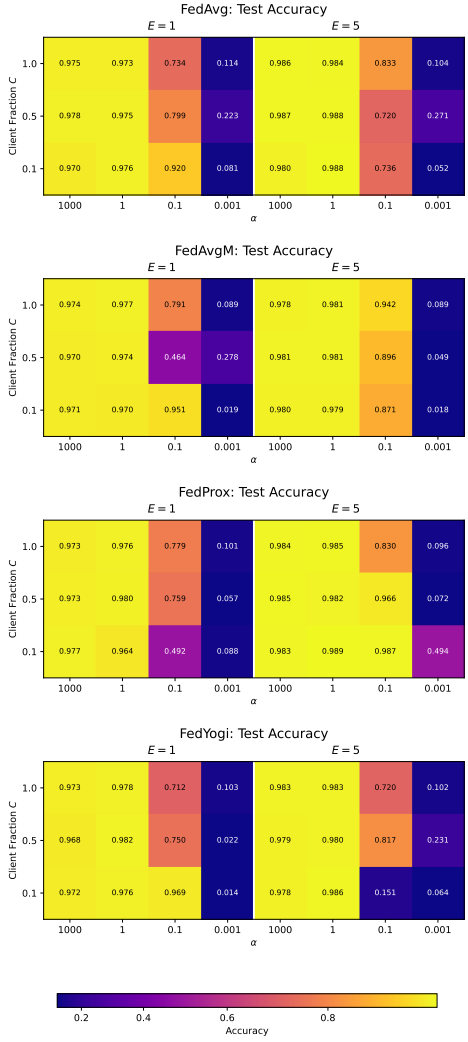
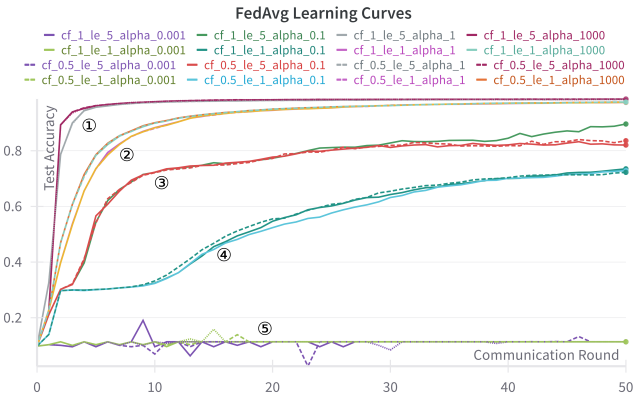
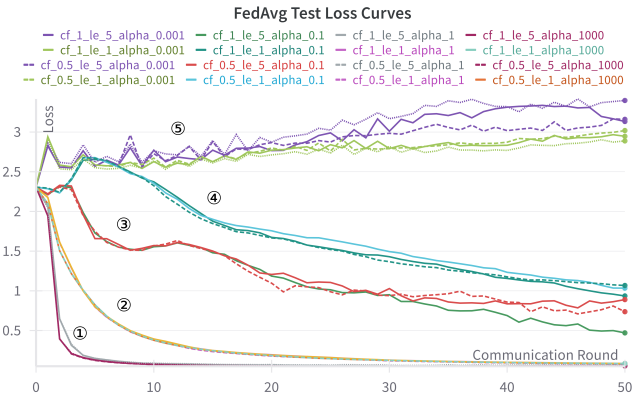


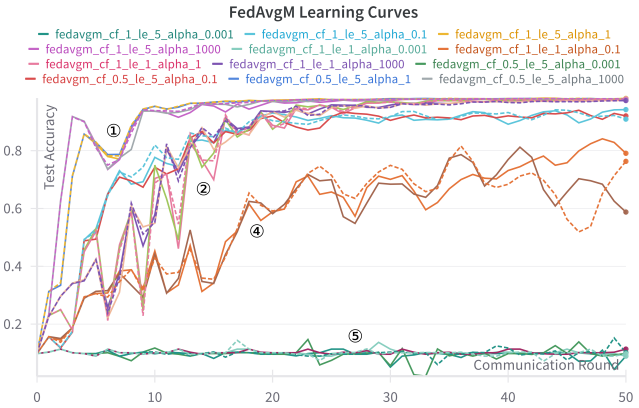
Fig. 3: Test accuracy heat-maps for Experiment III-A ($E = 1$ left, $E = 5$ right).



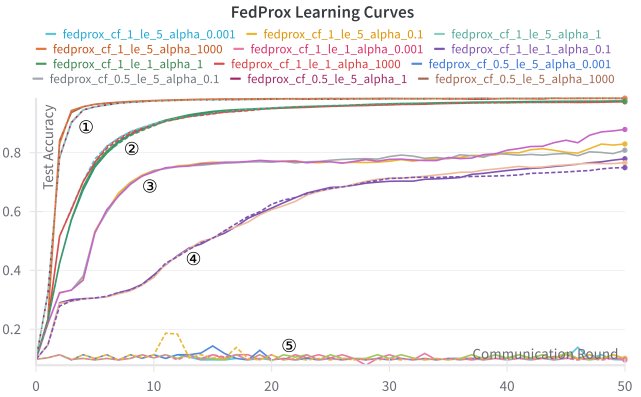
(a) ① Fast and stable convergence: IID or mild skew, high client fraction, moderate E.



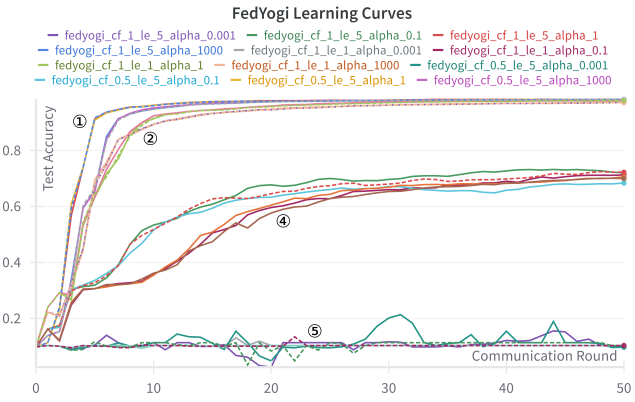
(b) ② Gradual but consistent convergence: slight non-IID, steady climb to 85–90%.



(c) ③ Slow or partial convergence: moderate heterogeneity, plateau ∼75–85%.



(d) ④ Poor convergence / drift: high skew or extended E, accuracy ∼60–75%.



(e) ⑤ Failure or divergence: extreme heterogeneity or bad hyperparams, chance accuracy.

Fig. 4: Test-accuracy (and one test-loss) learning curves for the federated learning algorithms. Each ①–⑤ marks a convergence group: ① fast, stable convergence; ② gradual but consistent convergence; ③ slow or partial convergence; ④ poor convergence with drift; ⑤ outright failure or divergence.