Advanced Topics in Machine Learning - PA4

Ahmad Ashraf [1] **Sarim Malik** [2]

Abstract

Federated Learning (FL) has emerged as a powerful paradigm for privacy-preserving machine learning by enabling decentralized training without sharing raw data. In this work, we revisit the theoretical foundations of FL through an empirical study of FedSGD, demonstrating its practical equivalence to centralized SGD under idealized conditions. Building on this foundation, we conduct a comprehensive analysis of FedAvg across a range of challenging scenarios. Our evaluation examines its sensitivity to key hyperparameters, including the number of local epochs and the client sampling fraction, as well as its behavior under varying degrees of data heterogeneity induced through Dirichlet partitions. Finally, we benchmark several state-of-the-art FL algorithms designed to mitigate client drift and handle non-IID data, including regularization-, control-variate-, gradient-harmonization-, and sharpness-aware methods. Through extensive experiments, we provide a comparative assessment of their convergence behavior, robustness, and practical trade-offs, offering insights into when and why each method excels.

Our codebase with complete experiments can be found here: https://github.com/s-malix21/atml-fall2025 and backup: https://github.com/ATML-AshrafxSarim/PA4.

1. Introduction

Modern machine learning systems rely heavily on large-scale datasets to achieve high predictive performance. However, in many real-world applications, data is inherently sensitive and cannot be freely shared due to privacy, legal, or ethical concerns. For instance, medical institutions such as hospitals often possess rich patient data, but privacy regulations and institutional policies prevent direct sharing of this information. This raises a critical question: how can multiple parties collaboratively leverage their data to train high-performing models without compromising privacy?

Federated Learning (FL) addresses this challenge by enabling decentralized model training. In FL, multiple clients collaboratively train a shared global model without exchanging raw data. Each client C_i retains its local dataset D_i and periodically communicates model updates with a central server. This approach preserves privacy while allowing knowledge to be aggregated across diverse data sources, making FL suitable for sensitive domains such as healthcare, finance, and edge computing.

The theoretical foundation of FL is closely linked to centralized training. In particular, FedSGD has been shown to be equivalent to centralized stochastic gradient descent under ideal conditions. This equivalence extends to FedAvg, which generalizes FedSGD by performing multiple local updates before aggregation. While FedAvg matches centralized training performance in IID settings, its behavior can vary under non-IID data distributions, motivating careful analysis and the development of specialized methods.

A wide variety of approaches have been proposed to improve FL in heterogeneous or challenging settings. These methods can be broadly categorized as follows:

- Regularization-based methods, such as FedProx, which add proximal terms to the local objective to reduce divergence from the global model.
- Control variate-based methods, such as SCAFFOLD, which leverage variance reduction techniques to correct for client drift.
- Gradient harmonization-based methods, such as FedGH, which adjust local gradients to improve consistency across heterogeneous clients.
- Sharpness-aware minimization, such as FedSAM, which optimizes for flatter minima to enhance generalization under non-IID distributions.

In this paper, we provide a systematic evaluation of FedAvg and its proposed improvements across a range of non-IID scenarios. Specifically, we aim to answer the following questions:

1. Does FedSGD behave equivalently to centralized SGD in practice, and what does this reveal about the theoretical foundations of FedAvg?

2. How does FedAvg perform under different non-IID settings?

3. How do hyperparameters, such as the number of local steps K and the fraction of participating clients n/N, impact the performance of traditional FL methods?

4. What are the relative improvements offered by specialized algorithms over FedAvg in terms of convergence, generalization, and inference performance?

Through this analysis, we provide a comprehensive analysis of the theoretical foundations of Federated Learning and a comparison of FL methods, highlighting their trade-offs and practical implications.

2. Methodology

2.1. Theoretical Equivalence of FedSGD and Centralized Training

2.1.1. Proof of Equivalence

Consider a global model with parameters θ_g. Let there be M clients, each with N_i samples. Denote the local model parameters at client i and training after local epoch of 1as $\theta_i^{(1)}$. We also express the global model at communication round t as θ_g^t.

After local training, server first receives all client updates and performs aggregation:

$$\theta_g^{t+1} = \sum_{i=1}^{M} \frac{N_i}{N} \theta_i^{(1)}, \tag{1}$$

where $N = \sum_{i=1}^{M} N_i$. This is equivalent to a global aggregated loss, defined as :

$$\mathcal{L}(\theta_g^t) = \sum_{i=1}^{M} \frac{N_i}{N} \mathcal{L}_i(\theta_i^{(1)}),$$

where $\mathcal{L}_i(\theta)$ is the local loss at client i. Then, the server update is equivalent to performing an SGD step on the global loss:

$$\theta_g^{t+1} = \theta_g^t - \eta \nabla \mathcal{L}(\theta_g^t).$$

$$\begin{aligned}
\theta_g^{t+1} &= \theta_g^t - \eta \nabla \mathcal{L}(\theta_g^t) \\
&= \sum_{i=1}^{M} \frac{N_i}{N} \theta_g^t - \eta \sum_{i=1}^{M} \frac{N_i}{N} \nabla \mathcal{L}_i(\theta_g^t) \\
&= \sum_{i=1}^{M} \frac{N_i}{N} \left(\theta_g^t - \eta \nabla \mathcal{L}_i(\theta_g^t) \right)
\end{aligned}$$

$$\theta_g^{t+1} = \sum_{i=1}^{M} \frac{N_i}{N} \theta_i^{(1)}. \tag{2}$$

Since both equations (1) and (2) are equivalent, FedSGD is theoretically equivalent to centralized SGD when all clients participate and compute full-batch gradients over their local datasets.

2.1.2. Experimental Setup

We illustrate this equivalence empirically using the CIFAR-10 dataset. For simplicity, we preprocess the dataset using standard normalization (mean and standard deviation) and select only 6000 samples. These samples are distributed across 6 clients in a nearly IID fashion, with each client receiving $1{,}000$ samples.

We utilize **Model 1** (see Table 6) to ensure deterministic outputs. To further guarantee reproducibility, we fix the random seed for all experiments and initialize both the centralized and FedSGD models with the same state dictionary, ensuring identical starting parameters for both paradigms.

For FedSGD, we perform a single local epoch of training per client and store the local gradients. At the end of each communication round, the server aggregates gradients across clients for each layer parameter and updates the model manually.

We also train a centralized model on all 6000 samples, performing manual gradient descent in the same manner. During training, we monitor the overall test accuracy, the loss pattern, and the average parameter differences per layer between the centralized model and the FedSGD model, defined as:

$$\Delta\theta = \|\theta_{\text{centralized}} - \theta_{\text{FedSGD}}\|_2$$

where $\theta_{(.)}$ denotes the parameters of a given model, and $\|\cdot\|_2$ is the Frobenius Euclidean norm.

We perform 10 communication rounds (or 10 parameter update steps) with a learning rate of 0.01 to demonstrate the equivalence empirically in a lightweight setup.

2.2. Federated Averaging (FedAvg)

Federated Averaging (**FedAvg**) extends the idea of Federated SGD by allowing clients to perform multiple local training epochs before aggregation. This reduces the frequency of communication between the server and clients, thereby improving communication efficiency. However, increasing the number of local updates introduces the risk of *client drift*, wherein individual clients may converge toward different local minima due to data heterogeneity or extended local optimization. We conduct experiments to analyze the behavior of FedAvg in different settings.

2.2.1. Experimental Setup

We employ the CIFAR-10 dataset for all FedAvg experiments. The dataset is preprocessed using standard normalization (mean and standard deviation), and a subset of $10{,}000$ samples is selected. These samples are distributed across 10 clients in an IID manner to ensure uniform data distribution and isolate the effects of local updates and sampling strategies.

For all experiments, we use **Model 2** (see Table 6) as the base neural architecture. The model is trained using the SGD optimizer with a learning rate of 0.01. Unless otherwise specified, all clients participate in every communication round, and the global model is initialized identically across experiments to ensure reproducibility. For all experiments unless otherwise stated, we perform FedAvg for a total of 20 communication rounds and then perform further analysis.

2.2.2. Effect of Local Epochs (K)

In the first experiment, we study the influence of the number of local training epochs (K) on global model performance and stability. Specifically, we vary the number of local epochs as $K \in \{1, 5, 10, 20\}$ while keeping all other hyperparameters constant.

Our objective is to examine how increasing K affects the performance of the global model (through the global model's accuracy on the test set) and the degree of the client drift observed across communication rounds, measured as the deviation of local parameters from the global model, defined as:

$$d_t^\theta = \frac{1}{M} \sum_{i=1}^{M} \left\| \theta_i^{(t,K)} - \theta_g^t \right\|_2 ,$$

where d_t^θ denotes the average distance between each client's model $\theta_i^{(t,K)}$ (after K local training epochs in round t) and the global model θ_g^t before aggregation. Larger values of d_t^θ indicate greater client drift during round t.

Since higher local updates allow clients to optimize more independently, we expect improved communication efficiency but also increased drift when K is large.

2.2.3. Effect of Client Sampling Ratio (n/N)

In the second experiment, we explore the impact of client participation ratio on model convergence and stability. Specifically, we vary the client sampling fraction $n/N \in \{1.0, 0.5, 0.2\}$, where n denotes the number of active clients per communication round, and N is the total number of clients. During a training round, the server randomly samples clients according to the client sampling fraction, and performs local training on these clients only. Similarly, the global model aggregation is also performed through the selected clients only.

This experiment investigates how partial client participation affects the overall convergence behavior of FedAvg, and the variance of updates and the resulting stability of global training. By reducing n/N, communication costs are further minimized; however, stochasticity in aggregation may slow convergence or introduce instability. Through this controlled setup, we empirically analyze the robustness of FedAvg under varying participation levels.

2.3. Exploring Data Heterogeneity Impact

2.3.1. Model Architecture and Dataset Setup

We employ a CNN architecture with six convolutional layers featuring 32, 32, 64, 64, 128, and 128 filters, each followed by batch normalization. Adaptive pooling is applied before three fully connected layers (256→128→10 outputs). Dropout regularization is used with rates of 0.5 and 0.3 in the final layers.

Experiments are conducted on the CIFAR-10 dataset using the standard train-test split. To control computational costs while maintaining data heterogeneity, only 10% of the training data is used and distributed among clients. Data preprocessing applies standard normalization with mean and standard deviation of (0.5, 0.5, 0.5). We simulate ten clients, with all clients participating in each communication round.

2.3.2. Dirichlet Partitioning for Heterogeneity Simulation

We simulate data heterogeneity using a Dirichlet distribution with concentration parameter $\alpha \in \{0.05, 0.1, 0.5, 1.0, 100.0\}$. For each class k, client proportions are drawn from $\mathrm{Dir}(\alpha)$ across M clients, and samples are distributed accordingly.

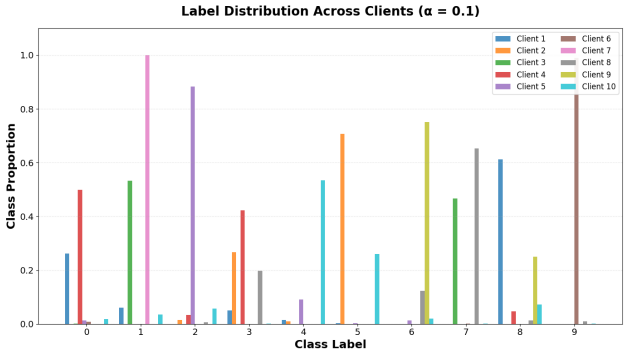

Figure 1. Visualization of label distribution across clients at Dirichlet setting α = 0.1, simulating high label skew in non iid setting

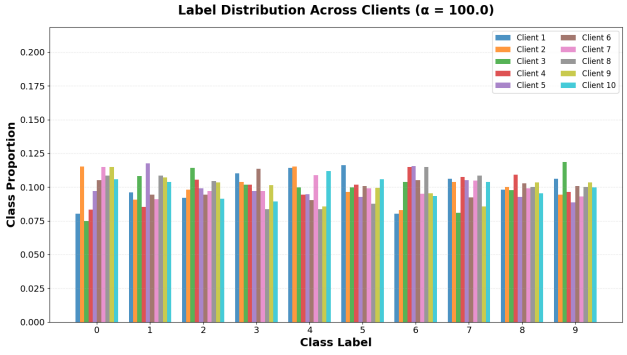

Figure 2. Visualization of label distribution across clients at Dirichlet setting α = 100.0, simulating iid setting

The concentration parameter α controls the level of heterogeneity. $\alpha = 100.0$ represents near-IID, uniform class allocation, $\alpha = 1.0$ shows moderate heterogeneity, whereas $\alpha = 0.1$ and $\alpha = 0.05$ are for highly skewed, non-IID distributions. Figures 1 and 2 show the resulting label distributions for $\alpha = 0.1$ and $\alpha = 100.0$, respectively, demonstrating the transition from non-IID to near-IID conditions.

2.3.3. Federated Training Protocol

Training was conducted over 50 communication rounds, with each client training locally for five epochs per round. We used SGD with a learning rate of 0.01, momentum of 0.9, and weight decay of 5×10^{-4}. Global aggregation followed the FedAvg rule, as defined above.

Adaptive batch sizing (minimum 2, maximum 64) was applied to handle clients with limited data and to prevent BatchNorm instability from small batches. After each aggregation round, the global model was evaluated on the standard CIFAR-10 test set.

2.3.4. Performance and Drift Metrics

Model behavior under heterogeneous conditions was evaluated using several metrics. Test accuracy measured the proportion of correct predictions on unseen data. Client divergence captured the distance between local and global model parameters, while the generalization gap reflected the difference between clients training accuracy and overall test accuracy. Gradient diversity indicated the consistency of client updates across rounds. Finally, label skew was estimated using the Jensen–Shannon divergence $\mathrm{JS}(P\|U)$ between each clients label distribution P and a uniform distribution U, quantifying the degree of non-IID data imbalance.

2.4. Mitigating Heterogeneity with Extensions of FedAvg

2.4.1. Experimental Setup and Model Architecture

We used a basic CNN model with two convolutional layers, each containing 64 filters of size 5×5, followed by max pooling. The network ends with three fully connected layers with 384, 192, and 10 units for CIFAR-10 classification. This was done on purpose; methods such as SCAFFOLD are designed to handle drift not through architectural tricks (also some methods are sensitive to BN, LN), therefore to demonstrate the nuance between all methods, we chose a really simple backbone CNN. Our experiments used a 25% subset of CIFAR-10, divided among 10 clients. The data were made non-IID using a Dirichlet distribution with concentration parameter $\alpha = 0.1$, which creates uneven class distributions across clients. Training ran for 50 communication rounds, each with 5 local epochs. Clients used SGD with learning rate 0.01, momentum 0.5, weight decay 10^{-3}, and batch size 64. The choice of $\alpha = 0.1$ provides a strongly heterogeneous setting that helps reveal how each method handles client drift during training.

2.4.2. Federated Learning Algorithms Implementation

FedAvg served as the baseline method, where clients perform local SGD and the server aggregates models by weighted averaging. FedProx added a small proximal term to each client's loss to limit deviations from the global model; we used $\mu = 0.01$ to balance stability and flexibility. SCAFFOLD used control variates, where the server keeps a global control vector and each client updates its own control after local training. FedGH adjusted conflicting client gradients at the server by orthogonalizing updates that point in opposite directions. FedSAM incorporated sharpness-aware updates by slightly perturbing the weights before computing gradients, using a perturbation radius of $\rho = 0.05$. **More details related to the algorithms can be**

found in Appendix A.3.

2.4.3. Evaluation Metrics and Measurements

We evaluated global performance using test accuracy, and test loss measured with cross-entropy. Convergence was monitored through training accuracy, the generalization gap (difference between client training accuracy and global test accuracy), and the number of rounds required to reach a certain test accuracy. To study heterogeneity effects, we measured client model divergence using the distance $\|\theta_i - \theta_{\text{global}}\|_2$, and also tracked gradient norms and cosine similarity between client updates. Computation cost differed slightly by method, with FedProx, SCAFFOLD, FedGH, and FedSAM requiring modestly more local processing.

2.4.4. Hyperparameter Sensitivity Experiments

To study FedProx behavior, we tested regularization values $\mu \in \{0.001, 0.01, 0.1\}$ and observed how different strengths of the proximal term affected stability and progress. For FedSAM, we evaluated perturbation radii $\rho \in \{0.01, 0.05, 0.1\}$ to understand how the sharpness-aware step size influences generalization. All methods were run under identical conditions, using the same data partitions, initialization seeds, and training settings to ensure a fair comparison (although number of rounds were reduced from 50 to 20 just for the hyperparam sensitivity analysis between these two methods).

3. Results

3.1. FedSGD vs Centralized SGD

Figure 3 illustrates the empirical equivalence between FedSGD and centralized SGD. The loss and accuracy trajectories for both methods overlap almost perfectly, indicating that their optimization dynamics are effectively identical. Towards the later stage of the communication rounds, we observe a small increase in the average parameter difference between the two models. However, this deviation is attributable to floating-point precision effects rather than algorithmic disagreement. Overall, the experiments confirm that FedSGD converges to the same solution as centralized SGD when trained under identical conditions.

3.2. Federated Averaging (FedAvg)

3.2.1. Effect of Local Epochs (K)

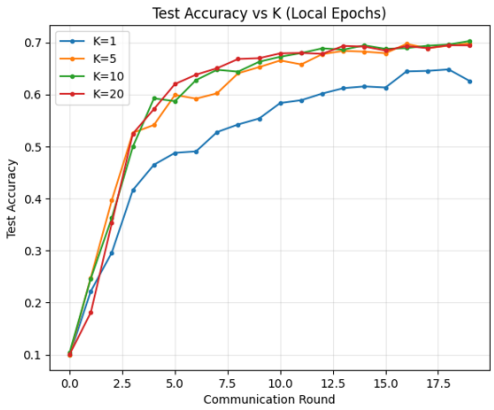

Figure 4. Loss trajectories for different values of local epochs $K \in \{1, 5, 10, 20\}$.

Figure 4 shows the effect of varying the number of local epochs. The results indicate that K=1 achieves noticeably lower accuracy compared to $K \in \{5, 10, 20\}$. However, the gains diminish once $K \geq 5$: the final accuracies for $K = 5$, $K = 10$, and $K = 20$ fall within a very similar range, as can be seen by Table 1. $K = 10$ achieves the highest accuracy but these values suggests that increasing the number of local epochs beyond 5 offers limited additional benefit for this setting, making $K = 5$ a reasonable and efficient choice for FedAvg.

Table 1. Effect of local epochs K in FedAvg on final accuracy and client drift.

K	Final Accuracy	Final Drift
1	0.6263	19.52
5	0.6984	44.89
10	0.7029	45.05
20	0.6947	53.83

The client drift results show a different trend. Here, $K = 1$ produces the lowest overall drift. Ideally, drift would remain close to zero for the $K = 1$ setting, but the presence of layers such as BatchNorm layers and Dropout introduces stochasticity into the local training dynamics, which increases variability regardless of the chosen K.

The results in Table 1 show that client drift increases consistently as the number of local epochs K grows. However, the rate of increase is relatively moderate: although $K = 20$ has the highest drift, the progression from $K = 5$ to $K = 10$ to $K = 20$ is gradual rather than explosive. A reason for this behavior can be that the model updates eventually satu-

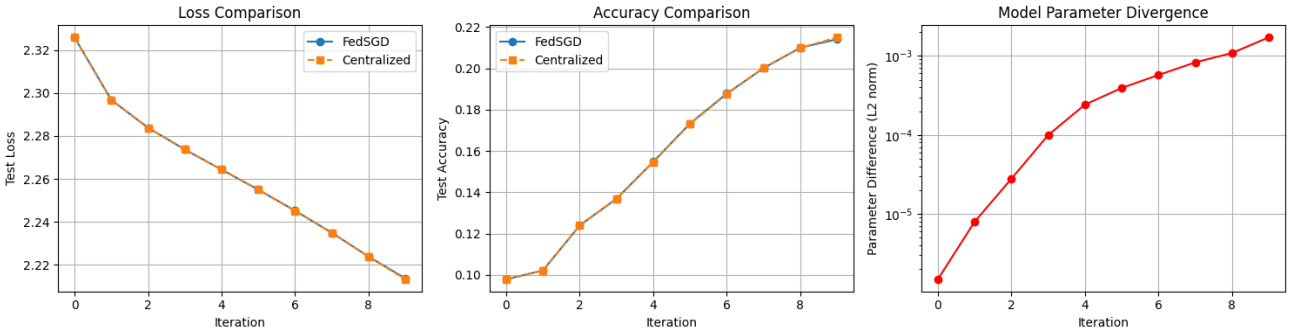

Figure 3. Comparison between FedSGD and centralized SGD, showing loss curves, accuracy curves, and average model-parameter differences across communication rounds.

rate: after several local steps, gradients become smaller and clients' movements slowly converge toward flatter regions of the loss landscape. As a result, while higher K naturally induces more divergence, the magnitude grows at a diminishing rate because clients' parameter updates become less aggressive in later local iterations.

3.2.2. Effect of Client Sampling Fraction (n/N)

Table 2. Effect of client sampling fraction on final accuracy, drift, and communication cost.

Sampling Ratio	$n/N = 1.0$	$n/N = 0.5$	$n/N = 0.2$
Final Accuracy	0.6984	0.6857	0.6729
Final Drift	44.89	47.87	38.38
Comm. Cost	200	100	40

The results in Table 2 show a clear trend: the final accuracy decreases as the client sampling fraction n/N is reduced. At the same time, the communication cost decreases proportionally with smaller values of n/N, since fewer clients exchange updates with the server each round. Using all clients in every round ($n/N = 1.0$) produces the highest accuracy because the aggregated update is computed from the complete client population. When only half the clients participate ($n/N = 0.5$), the accuracy decreases slightly, and the drop becomes more noticeable for $n/N = 0.2$, where only two clients contribute per round. Despite this reduction, the accuracy at $n/N = 0.2$ remains reasonably close to the full-participation case, while offering a substantially lower communication cost. This makes the $n/N = 0.2$ setting a practical and efficient choice in scenarios where communication bandwidth or computational resources are constrained.

Lower sampling fractions lead to substantially more variability in model updates across rounds.For example, when only $n/N = 0.2$ of clients participate, each round is influenced by a very small and potentially unrepresentative subset of data. As a result, the aggregated update can move in widely varying directions depending on which clients are selected. This produces the irregular and noisy drift trajectories observed for $n/N = 0.2$, as seen in Table 13. In contrast, using all clients ($n/N = 1.0$) yields more stable trajectories because the update reflects the full data distribution at each step.

From a drift perspective, L2-based drift naturally becomes more variable when fewer clients participate. Since drift is computed as the average L2 distance between the participating clients' local models and the global model, a small participant set means the estimate depends heavily on whichever clients happen to be sampled. With only 2 clients per round, this estimate fluctuates significantly, even if the underlying model behavior is consistent.

3.3. Impact of Data Heterogeneity on FedAvg

Table 3. Effect of data heterogeneity (α) on test accuracy, loss, divergence, and generalization.

α	Acc. (%)	Loss	Div.	Gen. Gap (%)
0.05	41.04	1.7066	0.0041	47.55
0.10	39.36	1.7794	0.0041	51.12
0.50	54.28	1.7496	0.0039	41.28
1.00	55.89	1.7537	0.0032	42.34
100.00	64.09	1.3970	0.0026	35.87

3.3.1. Impact of Data Heterogeneity on Global Model Performance

As shown by Table 3 The experiments reveal a clear negative relationship between data heterogeneity and global performance. Under the near IID setting $\alpha = 100$, FedAvg reaches a test accuracy of 64.09%, whereas the most heterogeneous configuration $\alpha = 0.05$ attains only 41.04%, creating a gap of more than 23 pecrcent.

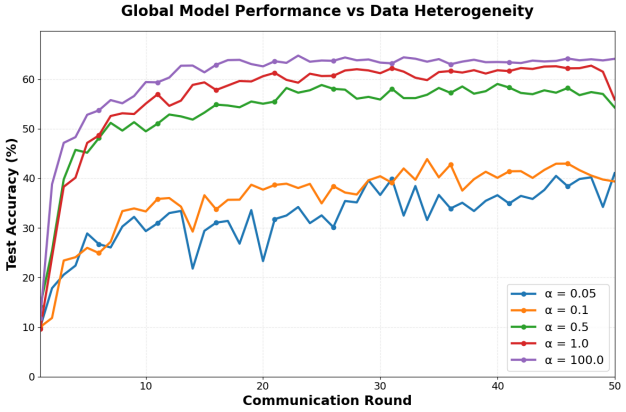

Figure 5. Visualization of global model performance vs data heterogeneity levels α across communication rounds

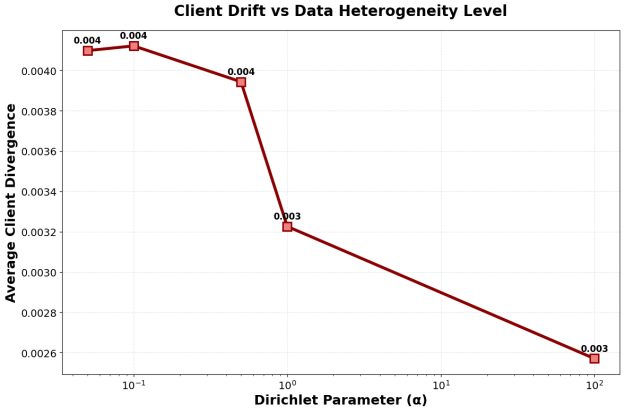

Figure 6. Visualization of client drift across different Dirichlet parameters α

Convergence behavior follows this pattern: the IID case improves smoothly and stabilizes early, while highly heterogeneous settings exhibit greater fluctuataios with accuracy between 20% and 45%. Moderate heterogeneity $\alpha = 0.5, 1.0$ yields more stable convergence but remains below the IID baseline. Test loss trends align with these observations, decreasing steadily as α increases and confirming the sensitivity of FedAvg to non IID data. Figure 5 demonstrates this relationship.

3.3.2. Client Level Analysis and Model Drift

As shown by Figure 6, client side diagnostics indicate that performance degradation is closely tied to divergence in local updates. Average model drift rises from 0.0026 under IID data to 0.0041 in the most heterogeneous setting, reflecting stronger discrepancies in the directions of local optimization. As data heterogeneity grows, the clients updates drifts further apart in similar patterns that were observed. Local training results further illustrate imbalance: at $\alpha = 0.05$ some clients achieve over 80% accuracy while others remain below 40%, corresponding directly to the skewness of their label distributions.

3.3.3. Data Distribution Characterization and Generalization Analysis

The Dirichlet sampling procedure produces the anticipated spectrum of heterogeneity. For $\alpha = 0.1$ most clients obtain highly concentrated distributions dominated by one or two classes, while $\alpha = 100$ yields nearly uniform allocations.

Jensen Shannon divergence captures this effect quantitatively, exceeding 0.4 in highly skewed settings and remaining below 0.1 in the IID configuration. These values correlate strongly with final performance. Generalization analysis from Table 3 shows that the gap between mean client

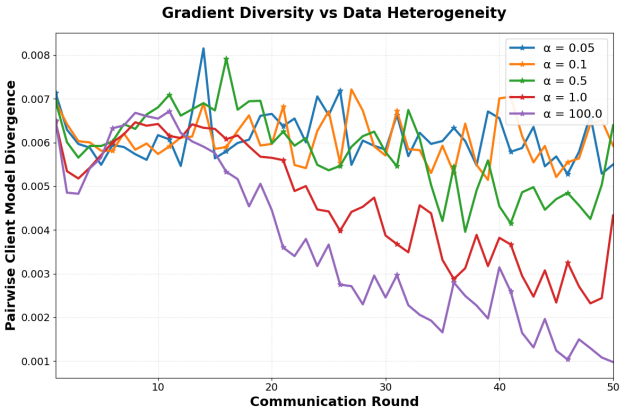

Figure 7. Visualization of Gradient Diversity vs Data Heterogeneity

training accuracy and global test accuracy grows notably with heterogeneity: from 35.87% under IID conditions to beyond 47–51% when data is highly skewed. This indicates that clients overfit their restricted local distributions while the aggregated model fails to generalize. Weight norm dynamics and gradient diversity patterns (see Figure 7) further reflect this instability, displaying higher and more variable values under severe non IID conditions.

Table 4. Comprehensive Performance Summary of Federated Learning Methods.

Method	Test Acc (%)	Test Loss	Train Acc (%)	Gen Gap (%)	Avg Div.	Early Acc (%)	Late Acc (%)
FedAvg	40.55	1.8834	94.58	54.03	0.00145	23.72	41.67
FedProx	44.01	1.8138	95.66	51.65	0.00140	23.81	42.60
SCAFFOLD	27.44	2.0251	69.26	41.82	0.00042	15.12	26.96
FedGH	39.90	2.5014	97.71	57.81	0.00154	18.02	40.89
FedSAM	46.04	1.6211	94.04	48.00	0.00146	23.35	45.12

3.4. Mitigating Heterogeneity – Comparison of FedAvg Extensions

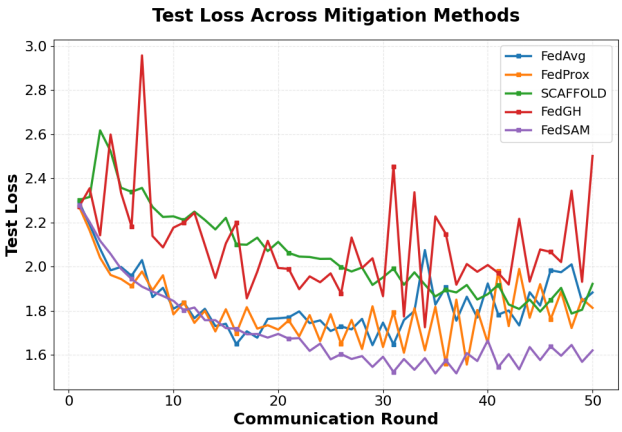

Figure 8. Test Loss across FL methods

3.4.1. Comparative Performance of Heterogeneity Mitigation Methods

As shown by Table 4, across all methods, the regularization based approaches delivered the strongest improvements under $\alpha = 0.1$. FedSAM reached the highest test accuracy at 46.04%, with FedProx close behind at 44.01%, both noticeably above the FedAvg baseline of 40.55%. SCAFFOLD remained significantly lower at 27.44%, and FedGH performed similarly to FedAvg at 39.90%. Test loss curves from Figure 8) support these findings: FedSAM converged most smoothly with the lowest final loss, whereas FedGH showed persistent oscillations and the highest loss values. The generalization gap was smallest for FedSAM at 48.00%, indicating better transfer to the global test distribution, while FedGH exhibited the largest gap at 57.81%. Early round performance was similar across methods, but clear separation emerged in the later stages, with FedSAM and FedProx retaining consistent gains.

3.4.2. Client Dynamics and Gradient Analysis

Client level behavior provides additional insight into the performance differences (see Table 4). SCAFFOLD achieved the lowest average model divergence, indicating effective suppression of client drift through its control variates, though this stability coincided with lower accuracy. FedProx achieved moderate divergence reduction while maintaining strong performance, which showed a more balanced trade-off between stability and flexibility. FedSAM displayed stable gradient norms throughout training, consistent with its sharpness aware updates that seek flatter minima and may explain its superior generalization. Cosine similarity analysis from Figure 9, shows that FedGH reduced gradient conflicts as intended, though its high loss volatility suggests that the orth. step may discard useful update components. Methods with stronger outcomes, particularly FedSAM and FedProx, demonstrated more coherent update alignment over time, whereas less stable methods showed more frequent directional shifts.

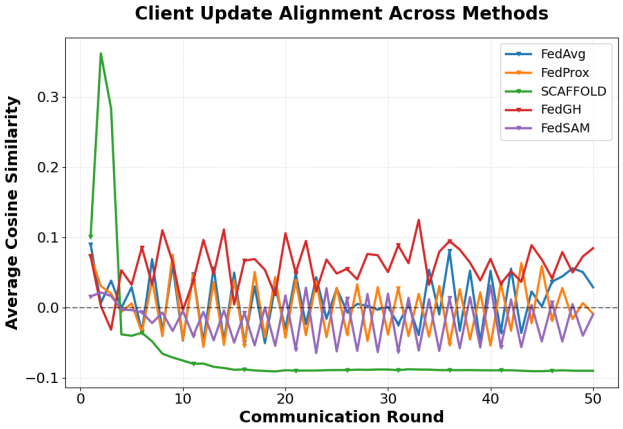

Figure 9. Client update alignment across methods uisng cosine similarity

3.4.3. Analysis of Additional Observations

Table 5 contains hyperparameter studies that indicate that both FedProx and FedSAM benefit from stronger regularization under non IID conditions. For FedProx, $\mu = 0.1$ produced the best accuracy among tested configurations, while for FedSAM the larger perturbation radius $\rho = 0.1$ outperformed smaller values. These results suggest that both approaches require careful tuning when data is highly

skewed. Although we did not perform an explicit computation or communication cost analysis, prior studies report that FedSAM roughly doubles local computation because each update requires gradients at both the original and perturbed weights (Forét et al., 2021; Qu et al., 2022), while SCAFFOLD introduces additional communication due to control variates (Karimireddy et al., 2020). Existing literature also notes that these overheads can influence practical deployment decisions (Kairouz et al., 2021). Nevertheless, reported findings in similar settings show that performance oriented methods such as FedSAM often justify their added cost when accuracy is the primary metric. The sensitivity of some methods, particularly SCAFFOLD, to optimization hyperparameters further highlights the need for method specific tuning in heterogeneous federated environments.

Note on SCAFFOLD's performance: Although SCAFFOLD is theoretically designed to reduce client drift, its practical performance depends strongly on the choice of optimization hyperparameters (Karimireddy et al., 2020). Our results showed that SCAFFOLD underperformed compared to FedAvg, FedProx, and FedSAM, which is consistent with prior reports showing significant sensitivity to learning rate schedules, momentum values, and the number of local updates (Reddi, 2021).

Several studies note that SCAFFOLD requires careful method-specific tuning to achieve its intended benefits. In heterogeneous federated settings, the control variate corrections can introduce instability when tuning is not properly matched to the data distribution or client sampling strategy (Karimireddy et al., 2020). This helps explain why our default FedAvg-optimized hyperparameters resulted in degraded accuracy for SCAFFOLD.

Table 5. Hyperparameter Sensitivity Analysis for FedProx (μ) and FedSAM (ρ). Final test accuracy is reported at Round 20/20 for each setting.

Method	Hyperparameter Value	Final Test Acc. (%)
FedProx	$\mu = 0.001$	37.81
	$\mu = 0.010$	36.48
	$\mu = 0.100$	39.12
FedSAM	$\rho = 0.010$	36.38
	$\rho = 0.050$	35.21
	$\rho = 0.100$	37.47

4. Discussion

4.1. FedSGD vs. Centralized SGD

Figure 3 demonstrates that the training trajectories of FedSGD and centralized SGD almost perfectly overlap, highlighting that the two methods behave equivalently when run under identical conditions. However, this equivalence critically depends on the training process being fully deterministic. In realistic federated environments, architectural components such as batch normalization and dropout introduce client-specific randomness, which can generate drift even when the underlying optimization algorithms are theoretically identical, as discussed in Section 3.2.1. Consequently, establishing strict equivalence between federated and centralized SGD is highly dependent on the model architecture, often requiring modifications such as those made when transitioning from Model 2 to Model 1 in Table 6.

4.2. Impact of Data Heterogeneity on FedAvg

The degradation we observe as data heterogeneity increases is consistent with well-established findings in federated learning. When clients possess highly skewed label distributions, their local optimization paths diverge from the global objective, causing aggregated updates to drift away from a shared optimum. This behavior, studied extensively in (Li et al., 2020) and (McMahan et al., 2017), explains both the rise in client–global divergence and the instability that emerges at low values of α. Our experiments reflect these effects clearly: as label skew intensifies, local gradients pull the global model in conflicting directions, producing oscillatory convergence patterns that simple averaging cannot correct.

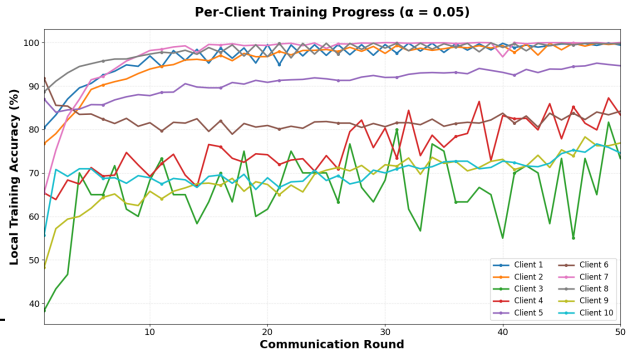

Figure 10. Per Client Training Progress under fixed Heterogeneity of $\alpha = 0.05$

The widening generalization gap further illustrates how client-specific specialization undermines global performance. While many clients achieve strong local accuracy under highly skewed settings as shown by Figure 10, the aggregated model performs poorly because each client optimizes for a narrow class subset rather than the full distribution, a phenomenon also highlighted in analyses of heterogeneity-induced overfitting (Zhao et al., 2018). The strong correlation between label skew and reduced test accuracy, supported by rising Jensen–Shannon divergence given

in Table 3, reinforces the view that global generalization fails when client optima differ too sharply.

Finally, our gradient diversity measurements show that client updates in heterogeneous regimes also differ in direction. This aligns with theoretical work in (Li et al., 2019; Zhao et al., 2018), which argues that biased client gradients require correction mechanisms to prevent systematic drift. Together, these observations clarify why FedAvg struggles under non-IID conditions: the core problem lies not in less amount of data, but in the conflict between localized training dynamics and global model generalization.

4.3. Understanding Mitigation Strategies in Heterogeneous Federated Learning

The differences observed across methods reflect the fundamental problems introduced by data heterogeneity in federated learning. FedSAM's stronger performance and smaller generalization gap are consistent with work showing that sharpness-aware updates guide models toward flatter minima that generalize better when clients train on biased or limited datasets (Qu et al., 2022). FedProx also performs reliably, and its proximal regularization appears to mitigate client drift in proportion to the severity of data skew, echoing earlier findings that stronger regularization becomes more important as heterogeneity increases (Li et al., 2020).

In contrast, SCAFFOLD's weak performance highlights the gap between its theoretical guarantees and its practical behavior. Although its control variates are designed to correct local drift, recent studies have noted that the method is highly sensitive to learning rates, local epochs, and heterogeneity levels; our results support this view, showing that overly strong corrections can suppress useful local adaptation. FedGH occupies a middle ground: while gradient harmonization reduces conflicts between client updates, its orthog. step may inadvertently remove information that contributes to convergence, which would explain the unstable loss dynamics observed in practice (Yuan et al., 2023).

These results suggest that choosing an appropriate method requires balancing accuracy goals against computational and communication constraints. FedSAM's higher computational cost may be justified in highly heterogeneous settings where accuracy is the priority, whereas FedProx remains an efficient and robust default when resources are limited. The strong hyperparameter sensitivity observed across several methods reinforces that careful, method-specific tuning is essential for reliable performance in heterogeneous federated environments. For the scope of this work, the results were also highly dependent on the degree of the Dirichlet parameter as well as the backbone CNN architecture itself, for instance we observed that adding excessive dropouts and batch normalization layers to the architecture would negatively impact methods sensitive to architectural differences such as SCAFFOLD and FedGH.

5. Conclusion

This work provides a comprehensive evaluation of federated learning methods under varying levels of data heterogeneity. Our experiments confirm that while FedSGD closely mirrors centralized SGD in theory, practical implementations reveal challenges introduced by architectural choices and client sampling.

The analysis of FedAvg highlights trade-offs between local computation and client drift, showing that performance can degrade significantly as data distributions become more skewed. Our experiments illustrate that heterogeneity leads to unstable convergence and reduced global accuracy, emphasizing the inherent difficulties in non-IID federated learning scenarios.

Among the mitigation strategies, FedSAM achieved the best performance through sharpness-aware optimization, while FedProx offered a practical balance between computational efficiency and accuracy gains. SCAFFOLD, despite its theoretical advantages, did not yield improved results under our settings, underscoring the need for careful hyperparameter tuning.

Overall, these findings emphasize that data heterogeneity is a core challenge in federated learning, and method selection should consider both the expected distribution skew and available computational resources. Future work should aim to develop approaches that remain robust across diverse heterogeneity levels while minimizing reliance on extensive hyperparameter optimization.

References

Forét, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2021. URL https://arxiv.org/abs/2010.01412.

Kairouz, P., McMahan, H. B., and et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021. URL https://arxiv.org/abs/1912.04977.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of Machine Learning Research*, volume 119 of *PMLR*, 2020. URL https://proceedings.mlr.press/v119/karimireddy20a.html.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Machine Learning and Systems (MLSys)*, 2020.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019. URL https://arxiv.org/abs/1907.02189.

McMahan, B., Moore, E., Ramage, D., and Hampson, S. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Qu, Z., Li, X., Duan, R., Liu, Y., Tang, B., and Lu, Z. Generalized federated learning via sharpness aware minimization. In *Proceedings of Machine Learning Research*, volume 162 of *PMLR*, 2022. URL https://proceedings.mlr.press/v162/qu22a.html.

Reddi, S. e. a. Adaptive federated optimization. In *ICLR*, 2021.

Yuan, H., Wu, J., and Zhang, Y. Federated gradient harmonization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, J., and Chandra, V. Federated learning with non-iid data. In *arXiv preprint arXiv:1806.00582*, 2018.

A. Appendix

A.1. Supplementary Material - FedSGD vs Centralized

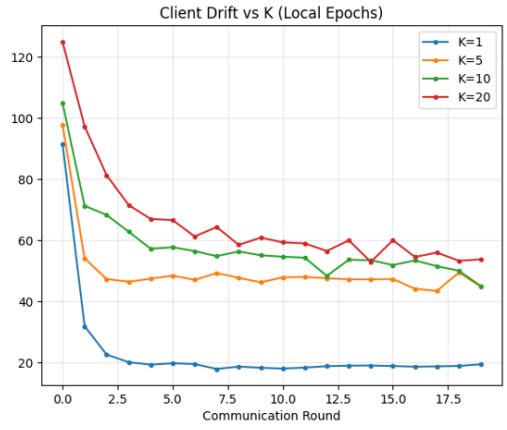

Figure 11. Client drift trajectories for different values of $K \in \{1, 5, 10, 20\}$.

Figure 12. Accuracy for different values of $n/N \in \{1, 0.5, 0.2\}$.

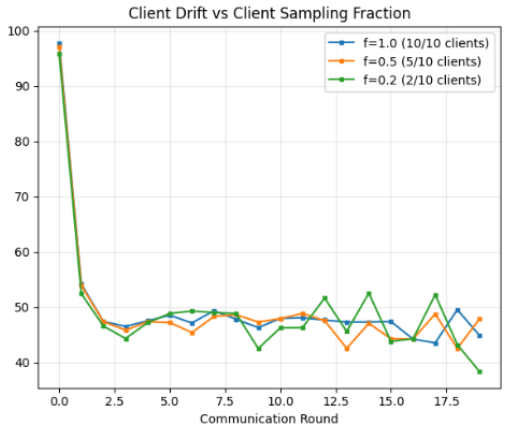

Figure 13. Client drift trajectories for different values of $n/N \in \{1, 0.5, 0.2\}$.

A.2. Supplementary Material - Exploring Data Heterogeneity Impact

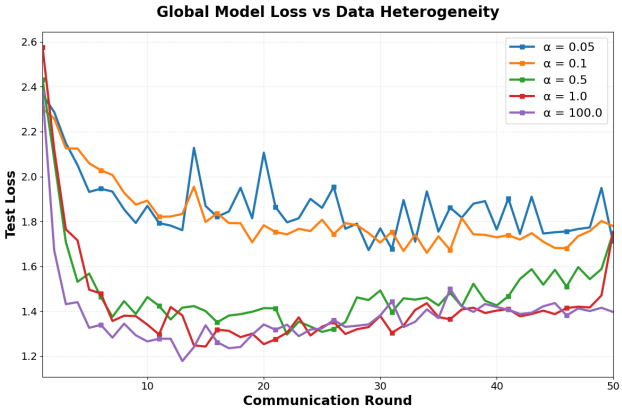

Figure 14. Visualization of global model accuracy vs data heterogeneity levels α across communication rounds

Table 6. Detailed structure of model architectures used for our experiments. Models are identical except for the use of Group Normalization and Batch Normalization in Model 1 and Model 2 respectively. This was done to ensure deterministic performance in Task 1.

Layer	Model 1. CNN with GroupNorm	Model 2. CNN with BatchNorm	Dropout (GN / BN)
Input	–	–	– / –
Conv1	Conv2d(3, 32, 3, pad=1)	Conv2d(3, 32, 3, pad=1)	– / –
Conv2	Conv2d(32, 32, 3, pad=1)	Conv2d(32, 32, 3, pad=1)	– / –
Norm1	GroupNorm(4,32)	BatchNorm2d(32)	– / –
Pool1	MaxPool2d(2,2)	MaxPool2d(2,2)	– / –
Conv3	Conv2d(32, 64, 3, pad=1)	Conv2d(32, 64, 3, pad=1)	– / –
Conv4	Conv2d(64, 64, 3, pad=1)	Conv2d(64, 64, 3, pad=1)	– / –
Norm2	GroupNorm(8,64)	BatchNorm2d(64)	– / –
Pool2	MaxPool2d(2,2)	MaxPool2d(2,2)	– / –
Conv5	Conv2d(64, 128, 3, pad=1)	Conv2d(64, 128, 3, pad=1)	– / –
Conv6	Conv2d(128, 128, 3, pad=1)	Conv2d(128, 128, 3, pad=1)	– / –
Norm3	GroupNorm(8,128)	BatchNorm2d(128)	– / –
Pool3	MaxPool2d(2,2)	MaxPool2d(2,2)	– / –
Adaptive Pool	AdaptiveAvgPool2d(2,2)	AdaptiveAvgPool2d(2,2)	– / –
FC1	Linear(512,256)+GN(16,256)	Linear(512,256)+BN1d(256)	– / 0.5
FC2	Linear(256,128)+GN(8,128)	Linear(256,128)+BN1d(128)	– / 0.3
FC3	Linear(128,10)	Linear(128,10)	– / –

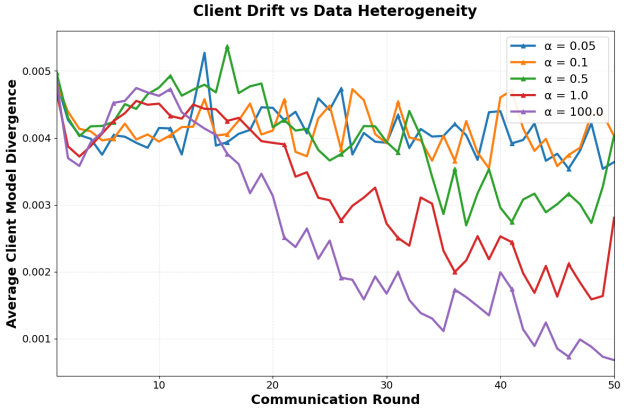

Figure 15. Visualization of client drift vs data heterogeneity levels α across communication rounds

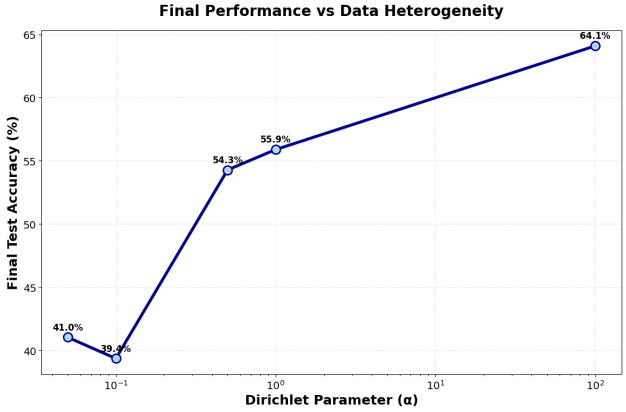

Figure 16. Visualization of final accuracies across different Dirichlet parameter settings

A.3. Supplementary Material - Mitigating Heterogeneity with Extensions of FedAvg

A.3.1. FedProx: Proximal Regularization

FedProx extends FedAvg by modifying each client's local optimization problem to restrict updates from drifting too far from the global model received at the beginning of the round. The local objective for client i becomes

$$\min_{\theta} \; F_i(\theta) \; + \; \frac{\mu}{2} \left\| \theta - \theta^t_{\text{global}} \right\|_2^2 ,$$

where F_i is the empirical risk on client i and θ^t_{global} is the global model at round t.

During training, the combined loss is computed as the sum of the standard cross-entropy term and the proximal penalty. The gradient with respect to the proximal term is

$$\nabla_\theta \left(\frac{\mu}{2} \left\| \theta - \theta^t_{\text{global}} \right\|_2^2 \right) = \mu \left(\theta - \theta^t_{\text{global}} \right),$$

which gently pulls the client parameters toward the global model. This reduces excessive divergence while allowing some degree of local adaptation.

A.3.2. SCAFFOLD: Control Variates for Drift Reduction

SCAFFOLD reduces client drift through control variates that correct biased local gradients. The server maintains a global control vector c, and each client maintains its own control vector c_i.

At each local update step, the corrected gradient is computed as

$$g_{\text{corr}} = g_{\text{local}} + (c - c_i),$$

which compensates for the mismatch between global and local gradient directions created by heterogeneous data.

After K local epochs with learning rate η, each client updates its control variate according to

$$c_i^+ = c + \frac{\theta^t_{\text{global}} - \theta^t_i}{K\eta},$$

capturing the deviation between the expected global update and the client's actual trajectory.

The server aggregates both model updates and control variates by weighted averaging, ensuring that c remains an accurate global correction term.

A.3.3. FedGH: Gradient Harmonization

FedGH addresses conflicting client updates by harmonizing gradients at the server. For each pair of participating clients (i, j), we compute the dot product

$$\langle g_i, g_j \rangle,$$

where negative values indicate conflicting gradient directions that would partially cancel each other during aggregation.

When a conflict is detected, FedGH removes the conflicting component by projecting one gradient onto the orthogonal complement of the other:

$$g_i' = g_i - \frac{\langle g_i, g_j \rangle}{\|g_j\|_2^2} g_j.$$

This procedure is applied symmetrically for both gradients and repeated for all $\binom{M}{2}$ client pairs. The computational complexity is therefore $O(M^2)$ per round, where M is the number of participating clients.

A.3.4. FedSAM: Sharpness-Aware Minimization

FedSAM incorporates the SAM optimization principle into the federated setting. Each client performs a two-step update. First, the client computes the gradient of the local loss $L(\theta)$ and constructs an adversarial weight perturbation:

$$\theta_{\text{adv}} = \theta + \rho \frac{\nabla L(\theta)}{\|\nabla L(\theta)\|_2},$$

where ρ controls the perturbation radius.

The client then evaluates the loss at θ_{adv} and computes the gradient

$$\nabla L(\theta_{\text{adv}}),$$

which emphasizes flatter regions of the loss landscape. After restoring the original weights, this SAM-modified gradient is applied using standard SGD. This process encourages solutions that generalize better across heterogeneous client distributions.

A.3.5. Additional Ablations and Results

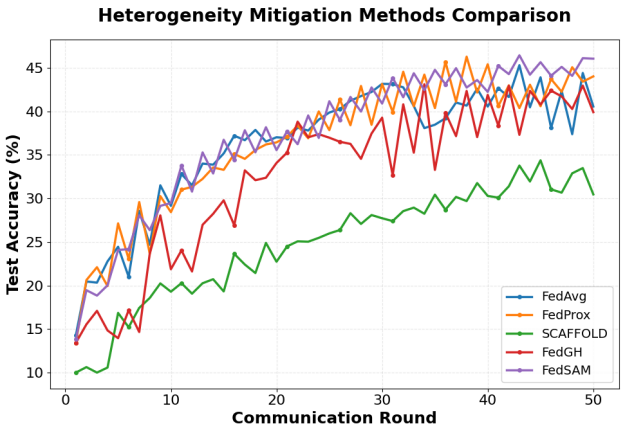

Figure 17. Heterogeneity Mitigation Methods Comparison

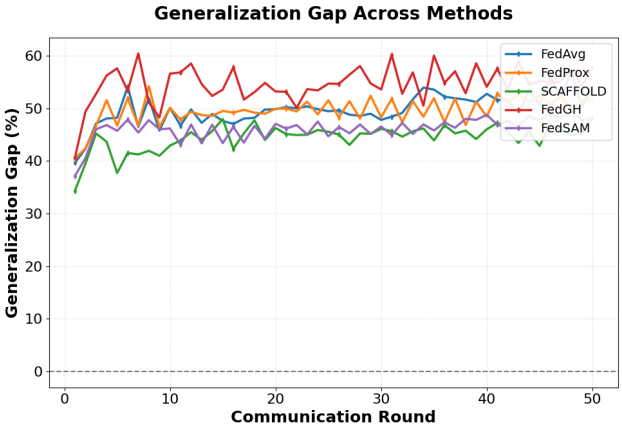

Figure 18. Generalization gap across methods