Federated Learning: Adaptive Strategies and Performance Evaluation

Introduction to Automated Driving – Practical Project
Team Federated Learning

June 2025

1 Introduction

In the modern landscape of artificial intelligence and edge computing, Federated Learning (FL) has emerged as a pivotal technique to enable collaborative machine learning while preserving data privacy [RAH+21]. Traditional machine learning workflows often rely on centralized data collection, where all training data must be aggregated into a single location [MMR+17]. This approach, while effective in many domains, poses significant challenges in terms of privacy, data security, and regulatory compliance, especially when dealing with sensitive or distributed data sources such as mobile devices, autonomous vehicles, or healthcare systems [CWY+24].

Federated Learning redefines this paradigm by distributing the model training process across multiple devices or clients, each of which retains its local data. Rather than transmitting raw data to a central server, clients perform local training and share only model updates (e.g., gradients or weights). These updates are then aggregated by a central server to improve the global model. This decentralized approach not only reduces privacy risks but also minimizes communication costs and enables learning from previously inaccessible data [MMR+17].

This approach is particularly valuable in domains like automated driving, where vehicles generate massive volumes of sensor and image data that are both privacy-sensitive and bandwidth-intensive to transmit. In such scenarios, FL enables vehicles to learn collaboratively from diverse driving experiences across regions, conditions, and users—without ever sharing raw sensor data. This makes FL a powerful enabler for scalable and privacy-preserving advancements in perception, decision-making, and autonomous control systems [RAH+21].

1.1 Problem Statement

This project addresses two key challenges in federated learning:

1.1.1 Evaluating Base Federated Learning Strategies on Different Partitions

We aim to evaluate the performance of adaptive optimization techniques in federated learning settings. The core objective is to analyze how strategies such as FedAvg and Adaptive Federated Optimization (FedOpt) perform under different data distribution scenarios, particularly in non-IID environments. This comparison is essential to identify methods that generalize well while maintaining computational efficiency and privacy.

1.1.2 Minimizing Round Timeouts in Full Participation Settings

In the second experiment, we investigate the challenges associated with full client participation (`fraction_fit = 1.0`), where all clients take part in every training round. This configuration ensures maximal resource utilization and consistent participation, but it also introduces the risk of some slower clients dictating the round duration.

To better reflect real-world heterogeneity, we simulate scenarios where clients differ in computational resources (e.g. CPU vs. GPU) and local training effort. Clients train independently, and the server synchronizes updates after a fixed interval, simulating a semi-asynchronous training process. The objective here is to assess how adaptive optimization techniques can help mitigate the impact of slow clients and reduce timeout risks, ensuring efficient and stable convergence even in challenging deployment environments. This pipeline will be tested using the previously mentioned FedAvg and Adaptive Federated Optimization, but also using a novel federated aggregation technique (FedAvg-Weighted) which, in addition to aggregating the weights based on the number of samples each client had, it also takes into consideration the number of epochs did on that specific client.

2 Algorithms

2.1 FedAvg

Federated Averaging (FedAvg) is a widely-used baseline in FL. Each client performs local training on its data, and the server aggregates the model updates using a weighted average.

Pseudocode: FedAvg

1. Initialize global model weights w_0
2. For each round $t = 1, 2, \ldots, T$:
   (a) Server sends w_t to selected clients
   (b) Each client k updates w_t to w_{t+1}^k via local SGD
   (c) Server aggregates updates: $w_{t+1} = \sum_k \frac{n_k}{n} w_{t+1}^k$

2.2 Adaptive Federated Optimization (FedOpt)

We implemented the Adaptive Federated Optimization strategy proposed by Reddi et al. [RCZ+20], which extends FedAvg with adaptive optimization methods (AdaGrad, Adam, Yogi) at the server level. Adaptive Federated Optimization introduces federated versions of adaptive optimizers like Adagrad, Adam, and Yogi to address challenges in federated learning, particularly data heterogeneity and communication efficiency. By integrating these adaptive methods, the approach enhances convergence and performance over traditional methods like FedAvg, especially in non-convex settings with diverse client data. The **pseudocode** for this approach can be seen in Figure 1.

Algorithm 2 **FEDADAGRAD**, **FEDYOGI**, and **FEDADAM**

Initialization: $x_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ **(FEDADAGRAD)**
  $v_t = v_{t-1} - (1 - \beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$

Figure 1: Pseudocode for Adaptive Federated Optimization

2.3 Weighted-FedAvg

This algorithm has the same idea as FedAvg (Section 2.1), but all the clients are chosen for training (instead of few of them), the clients train for different number of epochs and, in the aggregation step, the weights are scaled based on both the number of samples and the number of epochs the local weights were trained on.

Pseudocode: Weighted-FedAvg

1. Initialize global model weights w_0
2. For each round $t = 1, 2, \ldots, T$:
   (a) Server sends w_t to **all** clients
   (b) Each client k updates w_t to w_{t+1}^k via local SGD on the local data (n_k samples) and on a various number of epochs (e_k).
   (c) Server aggregates updates: $w_{t+1} = \sum_k \frac{n_k \cdot e_k}{\sum_j n_j \cdot e_j} w_{t+1}^k$

3 Experimental Setup

3.1 Task and Dataset

We selected the CIFAR-10 dataset [KH+09] for our image classification task. CIFAR-10 is a benchmark dataset consisting of **60,000 32x32 color images** in **10 different classes**, with 6,000 images per class.

Figure 2 shows example images from each class.

To perform the classification task, we employed the **VGG-11** architecture [SZ14], a convolutional neural network (CNN) developed by the Visual Geometry Group at the University of Oxford.

Figure 2: Sample images from the CIFAR-10 dataset.

VGG-11 consists of 11 layers: 8 convolutional layers followed by 3 fully connected layers. It is known for its simplicity and effectiveness in image recognition tasks.

The model was trained under both centralized and federated learning setups. In both cases, we implemented the model using PyTorch, and for the federated learning setup, we used the Flower framework [BTM+20] to simulate multiple clients.

Figure 3 provides a high-level overview of the VGG-11 architecture.

3.2 Partitioning Strategies

To simulate a variety of realistic and synthetic data heterogeneity scenarios in federated learning, we implemented several data partitioning strategies. (examples in Figure 4)

- **IID** – uniform, randomly sampled data distribution.
- **Dirichlet** – label distributions drawn from a Dirichlet distribution with configurable concentration parameter.
- **Pathological** – each client receives data from a limited number of classes (e.g., two).
- **Shard** – data is sorted by label and divided into shards, with each client receiving a few shards.
- **Size** – clients are assigned partitions of configurable sizes.
- **Linear, Square, Exponential** – variations that assign partition sizes based on linear, squared, or exponential proportions.

To support flexible experimentation, we designed a partitioning manager that abstracts partitioner instantiation. This system allows for easy switching between partitioning strategies via configuration arguments, rather than code changes. Additional hyperparameters (e.g., Dirichlet α,

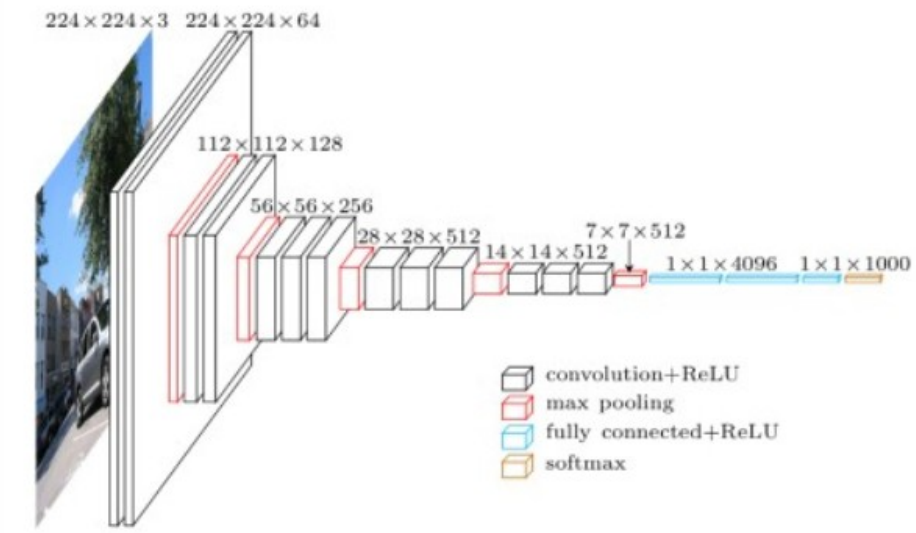


Figure 3: VGG-11 architecture used for image classification.

number of classes per partition) are passed as keyword arguments and handled internally by the manager.

4 Experiments and Results

For all experiments, the dataset was first divided into training and test sets, with 10% of the data reserved for testing. From the training set, 20% was further allocated for validation.

4.1 Baseline Performance

To establish a centralized performance reference, we trained **VGG-11** in a nonfederated setting.

Training was carried out on a single device using the Adam optimizer with a learning rate of 0.001, a batch size of 64, and for a total of 50 epochs.

This centralized baseline serves as a reference point for evaluating the performance of federated learning approaches under identical model and different data conditions.

The final performance was measured using the accuracy of the classification and cross-entropy loss in the test set and a test loss of 1.07 and a test accuracy of 74.9% was obtained.

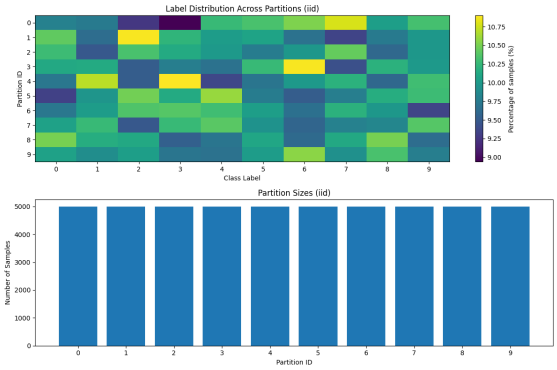
Other metrics are illustrated in Figure 5.

4.2 Experiment 1: Equal Hardware Experiment

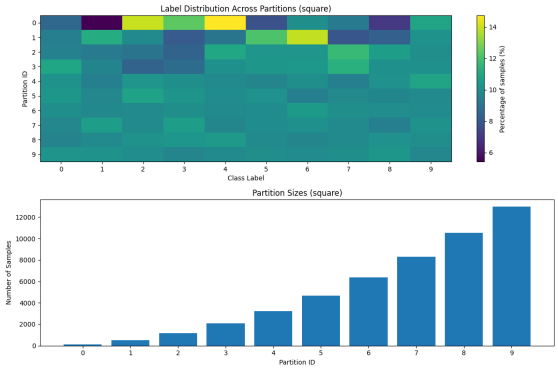
4.2.1 Hyperparameters

The following hyperparameters were used consistently across all configurations in Experiment 1, unless otherwise specified:

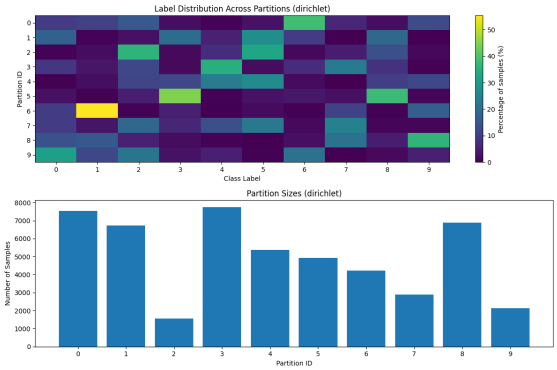
- **Number of clients**: 5
- **Number of server rounds**: 50
- **Fraction of clients per round**: 0.3



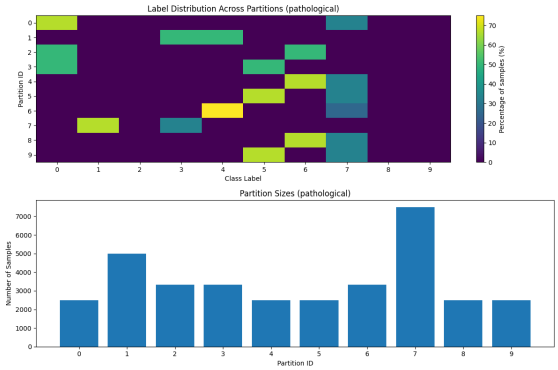
(a) IID Partitioning



(b) Square Partitioning



(c) Dirichlet partitioning



(d) Pathological partitioning

Figure 4: Visualization of various data partitioning strategies in federated learning

- **Server learning rate**: 0.01
- **Client learning rate**: 0.001
- **Local epochs**: 10
- **Batch size**: 64
- **Early stopping**: Enabled
- **Random seed**: 35

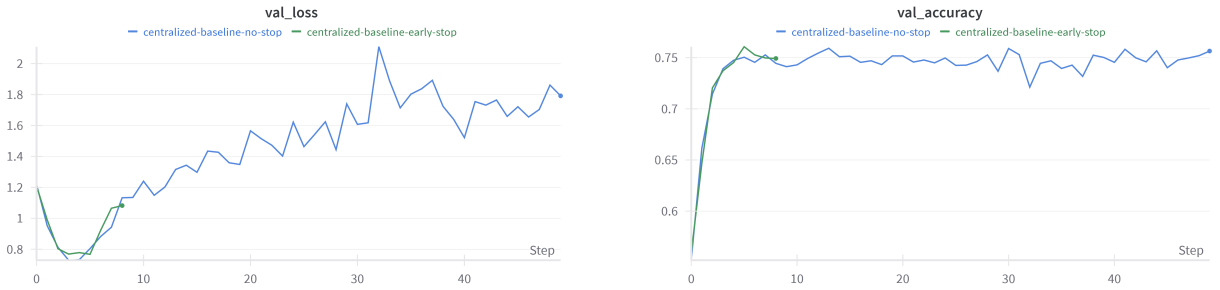
Optional parameters, such as the Dirichlet concentration parameter (`dirichlet-alpha = 0.5`), were used selectively depending on the scenario.

In all runs, we assume equally capable clients with uniform compute and communication capacity. Clients train locally for a fixed number of epochs before synchronizing their updates with the central server.

4.2.2 Partitioning Scenarios

To understand the impact of data heterogeneity, we simulate the following partitioning strategies:

- **IID**: Balanced and identically distributed data across clients.



(a) Baseline val loss

(b) Baseline val accuracy

Figure 5: Baseline performance metrics evolution

- **Square**: Imbalanced sample counts where the number of data points per client grows quadratically with the client index, creating significant data quantity heterogeneity.
- **Dirichlet**: Imbalanced data with heterogeneous class distributions, controlled by a Dirichlet prior.

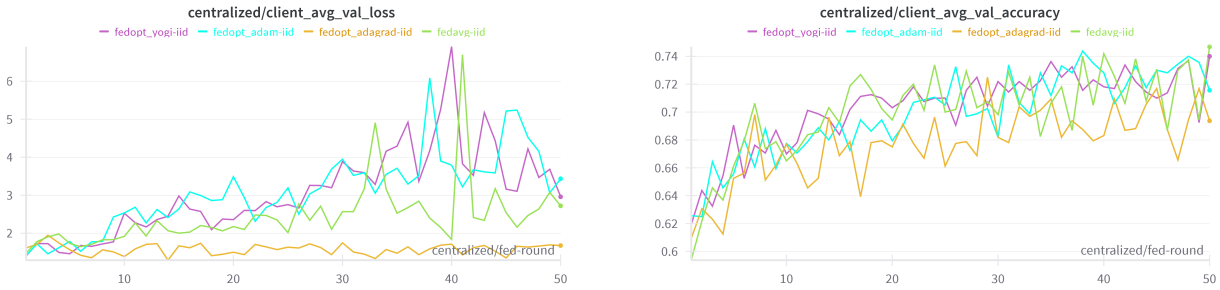
We now present results for each of these scenarios.

4.2.3 IID Scenario

In the IID setup, each client receives a uniform and balanced sample from the CIFAR-10 dataset, ensuring equal representation of all classes. This configuration serves as a performance baseline and illustrates the ideal case for federated training, as shown in Figure 4a.

Given the simulated nature of our federated learning environment, we can evaluate comprehensive metrics including average validation and test accuracy/loss across all clients—metrics that would not be accessible in real-world federated scenarios—as well as global model performance.

The experimental results demonstrate the comparative performance of different federated optimization strategies under IID conditions. Figure 6 shows the client-averaged validation metrics, while Figure 7 presents the global model performance on the test set.



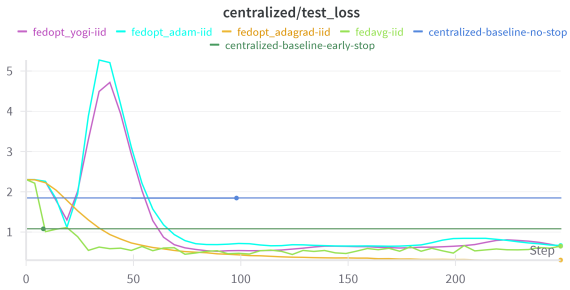
(a) Client average validation loss

(b) Client average validation accuracy

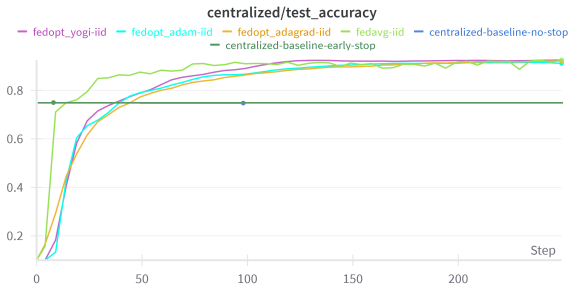
Figure 6: Client-side performance metrics evolution during training for IID partitioning

Performance Summary

Table 1 summarizes the final performance metrics for all tested federated optimization strategies under IID data distribution. The results reveal that FedOpt with Adam optimizer achieves the



(a) Global model test loss



(b) Global model test accuracy

Figure 7: Global model performance metrics evolution for IID partitioning

highest final accuracy 66%, while FedOpt with Yogi demonstrates the fastest convergence, requiring only 24 rounds to reach stable performance.

Table 1: Performance comparison of federated optimization strategies under IID partitioning

Strategy	**Final Accuracy**	**Final Loss**	**Rounds to Converge**
FedAvg	0.92	0.64	47
FedOpt (AdaGrad)	0.92	0.30	40
FedOpt (Adam)	0.92	0.66	40
FedOpt (Yogi)	0.91	0.64	24

The performance analysis indicates that while FedAvg provides a solid baseline with 64% accuracy, the choice of optimizer in FedOpt significantly impacts both final performance and convergence speed.

4.2.4 Square Partitioning Scenario

In this scenario, each client has an imbalanced distribution of samples, with the amount of samples varying with the square of the client index. This creates a highly heterogeneous data distribution where some clients have significantly more training data than others, as illustrated in Figure 4b.

The square partitioning scenario presents a more challenging federated learning environment compared to IID conditions, as the significant data imbalance between clients can lead to training instabilities and convergence issues. Figure 8 shows the client-averaged validation metrics, while Figure 9 presents the global model performance on the test set.

Performance Summary

Table 2 summarizes the final performance metrics for all tested federated optimization strategies under square partitioning. The results demonstrate that data imbalance significantly impacts algorithm performance, with `FedOpt` using `Adam` optimizer achieving the highest final accuracy (0.918), representing a notable improvement over the baseline `FedAvg` approach.

The performance analysis reveals interesting patterns under data imbalance conditions. While `FedAvg` shows strong baseline performance with 90.2% accuracy, it fails to converge within the observation period. `FedOpt` with `Adam` demonstrates the most robust performance, achieving 91.8% accuracy—the highest among all tested configurations. Both `Adam` and `Yogi` optimizers show enhanced performance compared to `FedAvg`, with `Adam` achieving the best accuracy and `Yogi` matching `AdaGrad` at 91.6%. Notably, `AdaGrad` achieves the lowest final loss (0.32) among the `FedOpt`

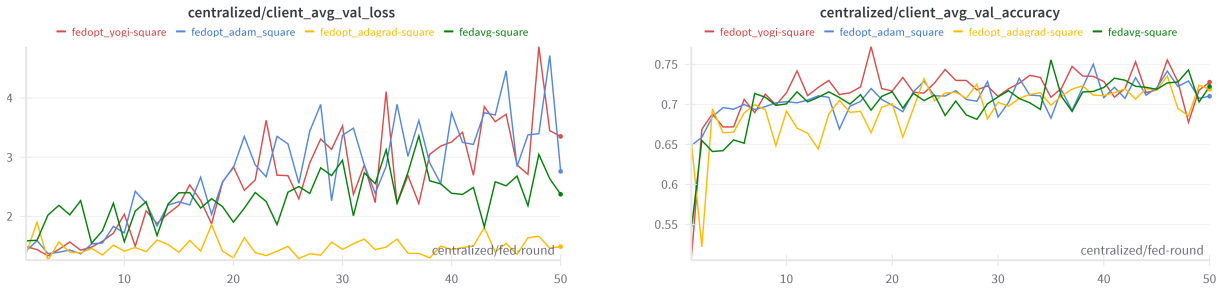


Figure 8: Client-side performance metrics: validation loss (left) and accuracy (right) evolution during training for square partitioning

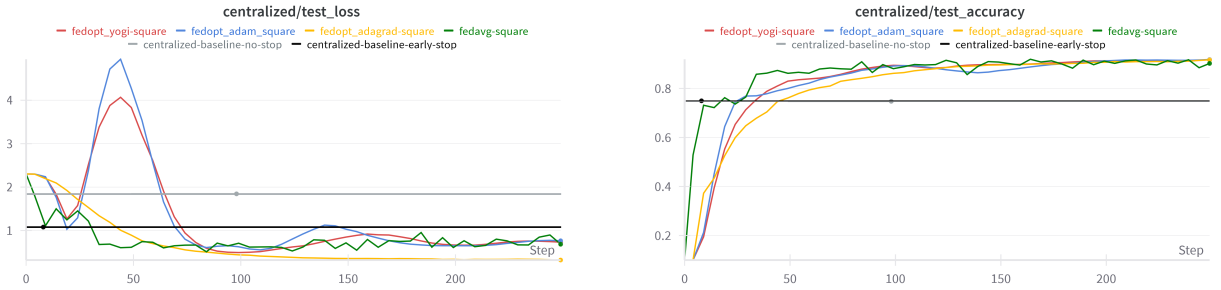


Figure 9: Global model performance metrics: test loss (left) and accuracy (right) evolution for square partitioning

variants, suggesting good convergence properties, while `Adam` has the highest loss (0.76) despite achieving the best accuracy. All `FedOpt` variants converge within 38–39 rounds, demonstrating efficient optimization in heterogeneous federated environments.

4.2.5 Dirichlet Partitioning Scenario

In this scenario, each client has an imbalanced distribution of both samples and classes, with the class distribution following a Dirichlet distribution. This creates the most challenging federated learning environment, where clients not only have varying amounts of data but also different class compositions, leading to significant statistical heterogeneity as shown in Figure 4c.

The Dirichlet partitioning represents the one of the more realistic federated learning scenario, where both data quantity and class distributions vary significantly across clients. This statistical heterogeneity poses substantial challenges for model convergence and generalization. Figure 10 shows the client-averaged validation metrics, while Figure 11 presents the global model performance on the test set.

Performance Summary

Table 3 summarizes the final performance metrics for all tested federated optimization strategies under Dirichlet partitioning. The results reveal the most dramatic performance variations among all tested scenarios, with some algorithms showing exceptional improvement while others struggle significantly under statistical heterogeneity.

The performance analysis for the Dirichlet partitioning scenario underscores the severe challenges posed by statistical heterogeneity. The baseline `FedAvg` strategy achieves the lowest accuracy (79.6%) and the highest final loss (1.84), failing to converge effectively within the training horizon.

Table 2: Performance comparison of federated optimization strategies under square partitioning

Strategy	Final Accuracy	Final Loss	Rounds to Converge
FedAvg	0.902	0.68	N/A
FedOpt (AdaGrad)	0.916	0.32	38
FedOpt (Adam)	0.918	0.76	39
FedOpt (Yogi)	0.916	0.73	38

Figure 10: Client-side performance metrics: validation loss (left) and accuracy (right) evolution during training for Dirichlet partitioning

Among the `FedOpt` variants, `AdaGrad` delivers the most robust results, achieving the highest final accuracy (87.9%) and the lowest loss (0.42), demonstrating strong generalization and convergence properties under non-IID conditions. `Adam` also performs well, reaching 86.5% accuracy but with a higher final loss (0.87), indicating less stable convergence. In contrast, `Yogi`, despite being effective in other scenarios, struggles under Dirichlet partitioning, matching `FedAvg` in accuracy (79.6%) and converging with a relatively high loss (1.41). These results highlight the sensitivity of optimization strategies to class imbalance and distributional shift, with `AdaGrad` emerging as the most effective optimizer in this heterogeneous setting.

4.3 Experiment 2: Full Participation Setting

The core idea of the experiment is to simulate federated learning with full client participation (`fraction_fit = 1.0`), while introducing variability in local training duration across clients to mimic real-world heterogeneity. Each client trains independently for a different number of epochs before synchronization, simulating a semi-asynchronous environment. This approach aims to explore how differing local compute capacities (e.g., CPU vs GPU) affect model convergence and system dynamics under full participation.

To simulate the idea of different devices training their data, we setup 3 different clients that will train their data per round as follows:

- the first client will train their samples for a number of epochs from the distribution $\mathcal{N}(5, 1)$ (integer approximation).
- the second client will train their samples for a number of epochs from the distribution $\mathcal{N}(10, 1)$ (integer approximation).
- the third client will train their samples for a number of epochs from the distribution $\mathcal{N}(15, 1)$ (integer approximation).

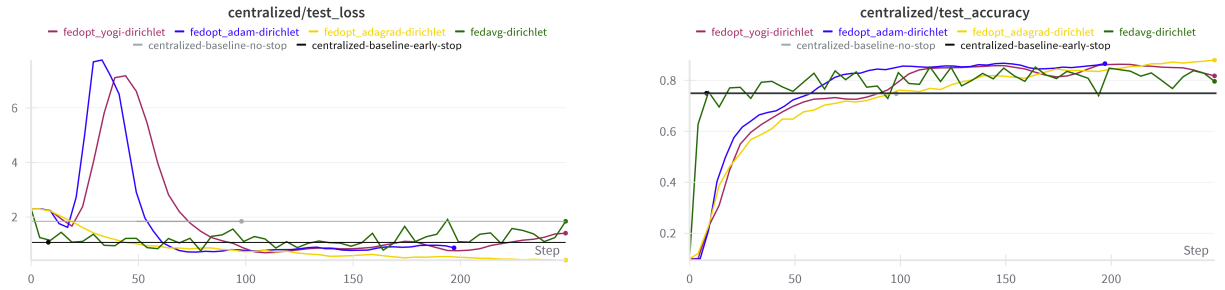


Figure 11: Global model performance metrics: test loss (left) and accuracy (right) evolution for Dirichlet partitioning

Table 3: Performance comparison of federated optimization strategies under Dirichlet partitioning

Strategy	**Final Accuracy**	**Final Loss**	**Rounds to Converge**
FedAvg	0.796	1.84	N/A
FedOpt (AdaGrad)	0.879	0.42	41
FedOpt (Adam)	0.865	0.87	40
FedOpt (Yogi)	0.796	1.41	43

Beside this setup, in our experiments we will variate the federated aggregation strategy (between FedAvg - Section 2.1, FedOpt with Adam - Section 2.2, and WeightedFedAvg - Section **??**) and the partitioning strategy (between IID, Dirichlet with $\alpha = 0.5$, Dirichlet with $\alpha = 0.1$, and Pathologic, all presented in Section 3.2). The fixed variables for our experiemnts are the following: `batch_size per client: 512`, `learning_rate per client: 0.001`, `optimizer per client: adam`, `fraction_fit: 1`, `num_server_rounds: 15`, `learning_rate server: 0.01`.

4.3.1 Centralized Results vs Individual Clients Results across Experiments

In this subsection of the results (Figures 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23), we plot how the losses and the accuracies evolve over time between each client and the server in each federated learning setup. We compare those with the baseline result (from Section 4.1). In each figure, there are 3 plots:

- in the first plot (left) the training loss of each client is compared to the centralized test loss,
- in the second plot (center) the test loss of each client is compared to the centralized test loss,
- in the third plot (right) the test accuracy of each client is compared to the centralized accuracy.

4.3.2 Results Comparison between Aggregation Strategies across Partition Strategies

In this subsection of the results (Figures 24, 25, 26, 27), we plot how different the test losses and the test accuracies of each aggregation strategy in comparison with the test loss and test accuracy from the baseline, based on various partition strategies. In each figure, there are 2 plots:

- the first plot (left) showcases the test loss across rounds for each aggregation strategy compared to the baseline,

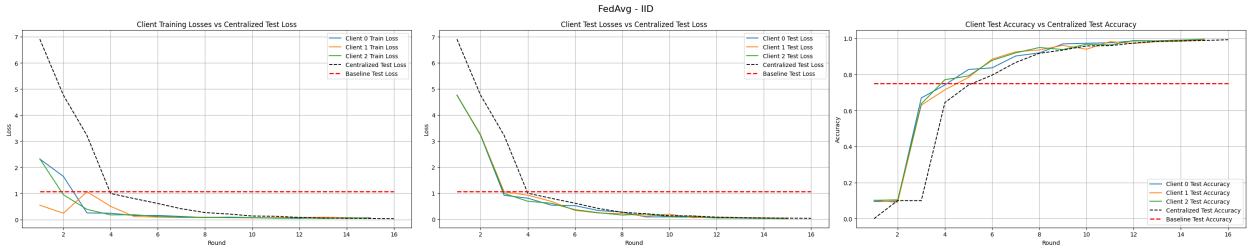


Figure 12: Results for FedAvg aggregating strategy and IID partitioning strategy.

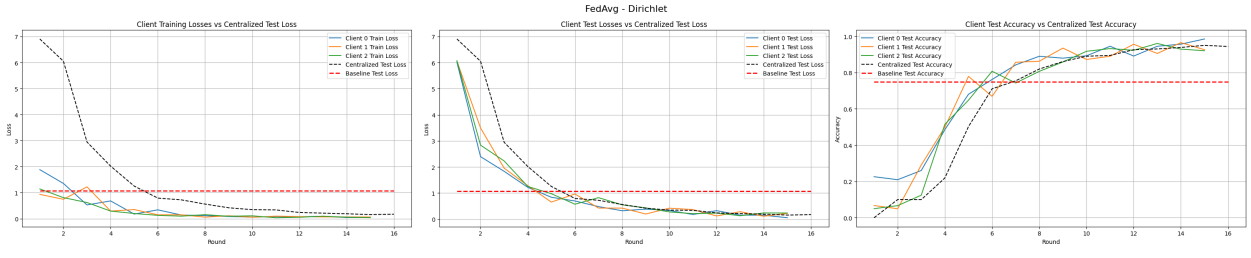


Figure 13: Results for FedAvg aggregating strategy and Dirichlet partitioning strategy.

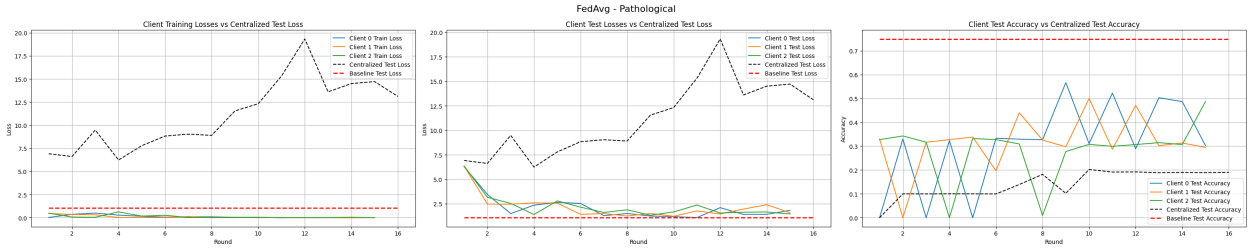


Figure 14: Results for FedAvg aggregating strategy and Pathological partitioning strategy.

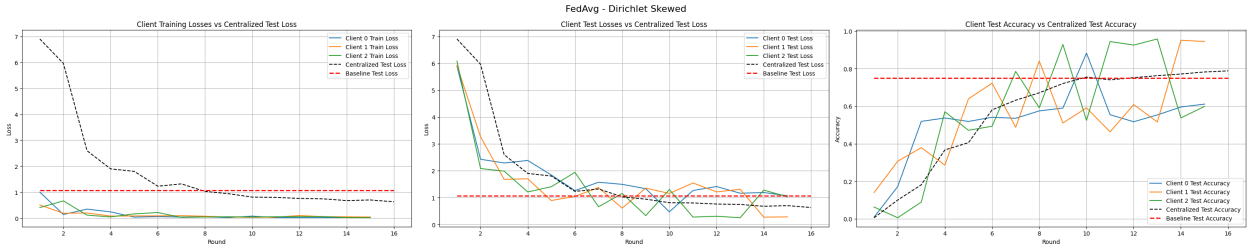


Figure 15: Results for FedAvg aggregating strategy and Dirichlet Skewed partitioning strategy.

- the second plot (right) showcases the test accuracies across rounds for each aggregation strategy compared to the baseline.

4.4 Results Comparison between Partition Strategies across Aggregation Strategies

In this subsection of the results (Figures 28, 29, 30), we plot how different the test losses and the test accuracies of each parition strategy in comparison with the test loss and test accuracy from the baseline, based on various aggregation strategies. In each figure, there are 2 plots:

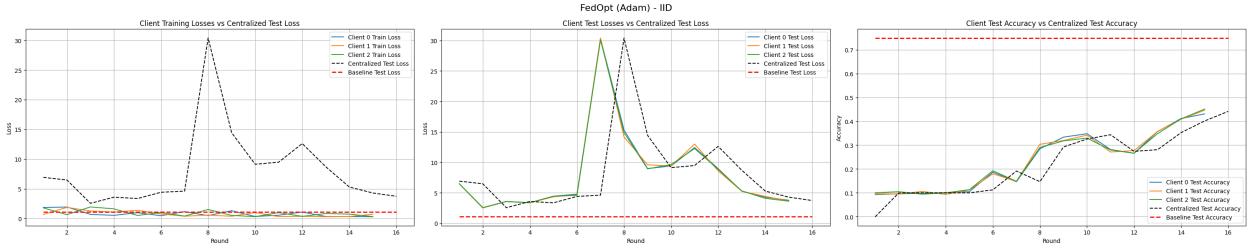


Figure 16: Results for FedOpt (Adam) aggregating strategy and IID partitioning strategy.

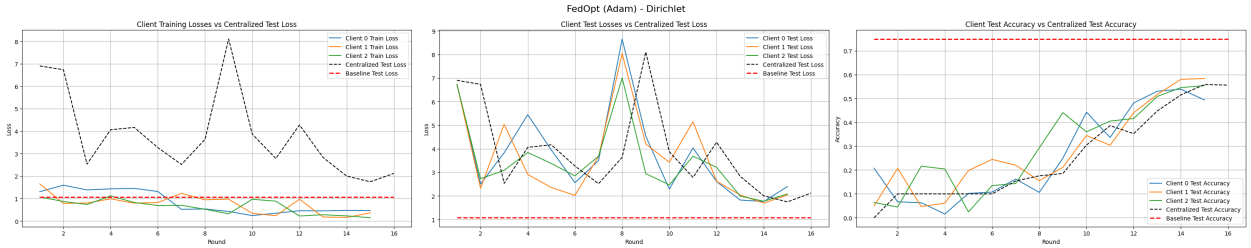


Figure 17: Results for FedOpt (Adam) aggregating strategy and Dirichlet partitioning strategy.

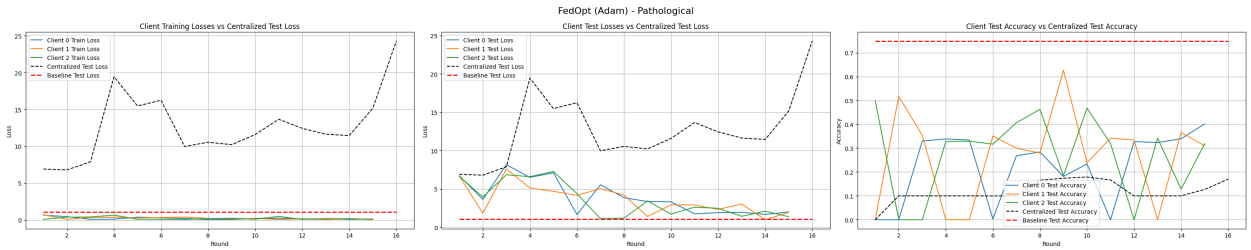


Figure 18: Results for FedOpt (Adam) aggregating strategy and Pathological partitioning strategy.

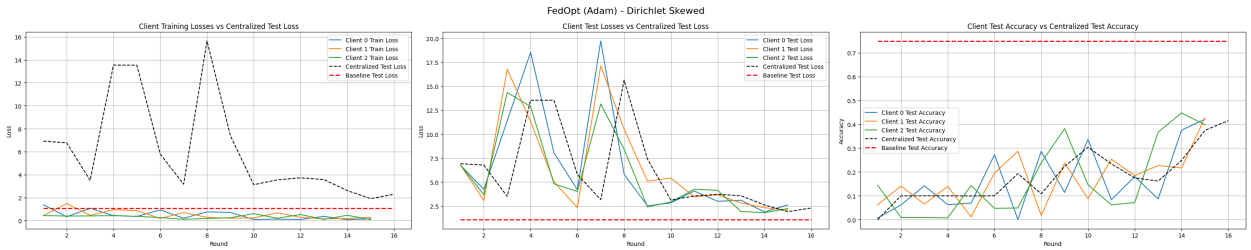


Figure 19: Results for FedOpt (Adam) aggregating strategy and Dirichlet Skewed partitioning strategy.

- the first plot (left) showcases the test loss across rounds for each partition strategy compared to the baseline,
- the second plot (right) showcases the test accuracies across rounds for each partition strategy compared to the baseline.

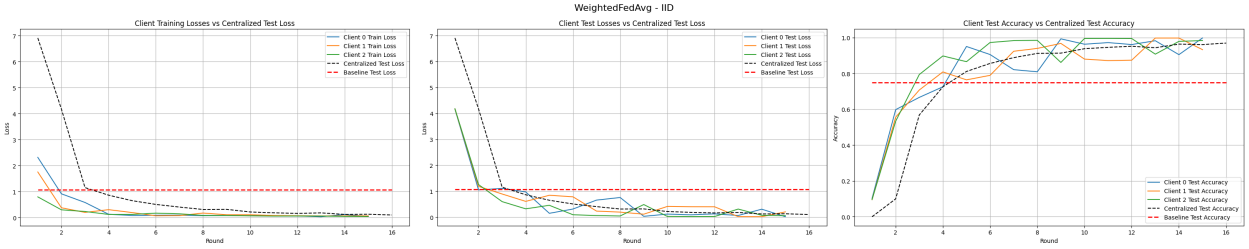


Figure 20: Results for Weighted-FedAvg aggregating strategy and IID partitioning strategy.

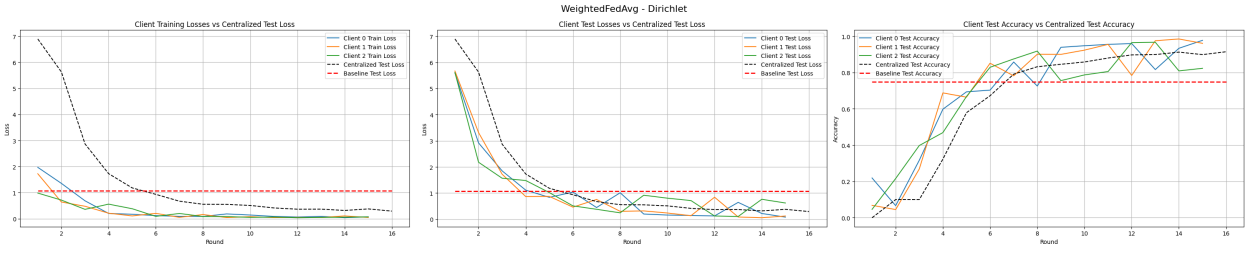


Figure 21: Results for Weighted-FedAvg aggregating strategy and Dirichlet partitioning strategy.

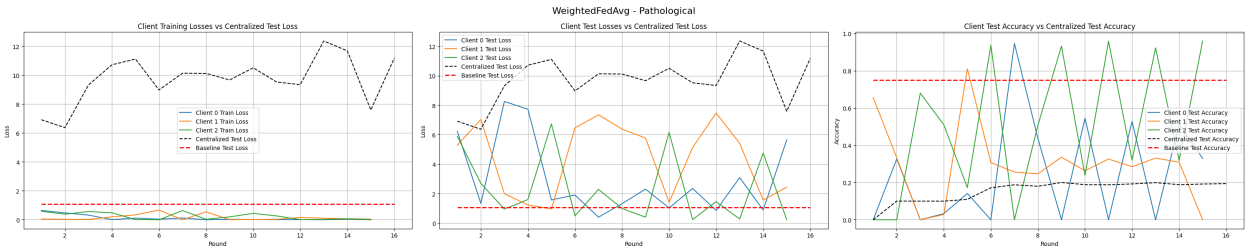


Figure 22: Results for Weighted-FedAvg aggregating strategy and Pathological partitioning strategy.

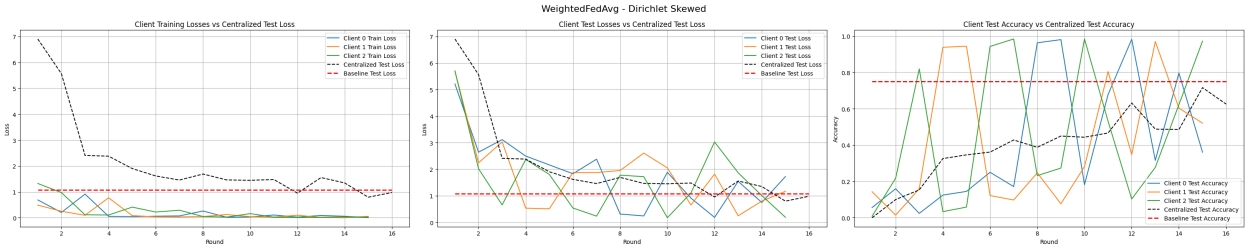


Figure 23: Results for Weighted-FedAvg aggregating strategy and Dirichlet Skewed partitioning strategy.

4.4.1 Summary Table

In the Table 4, the results after the training for each simulation are shown. The rounds to converge was computed as the approximate number of steps to reach a plateau in the test accuracies.

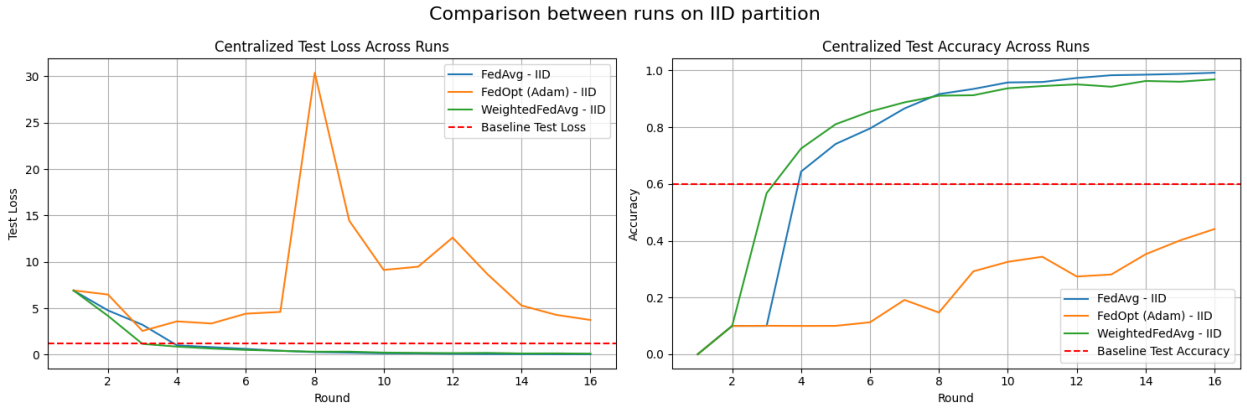


Figure 24: Comparison between various aggregation strategy on IID partition strategy.

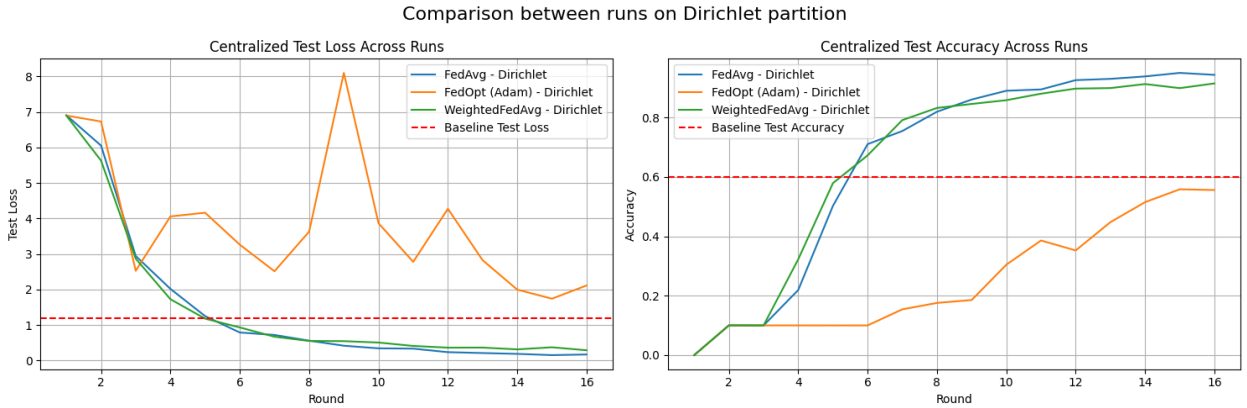


Figure 25: Comparison between various aggregation strategy on Dirichlet partition strategy.

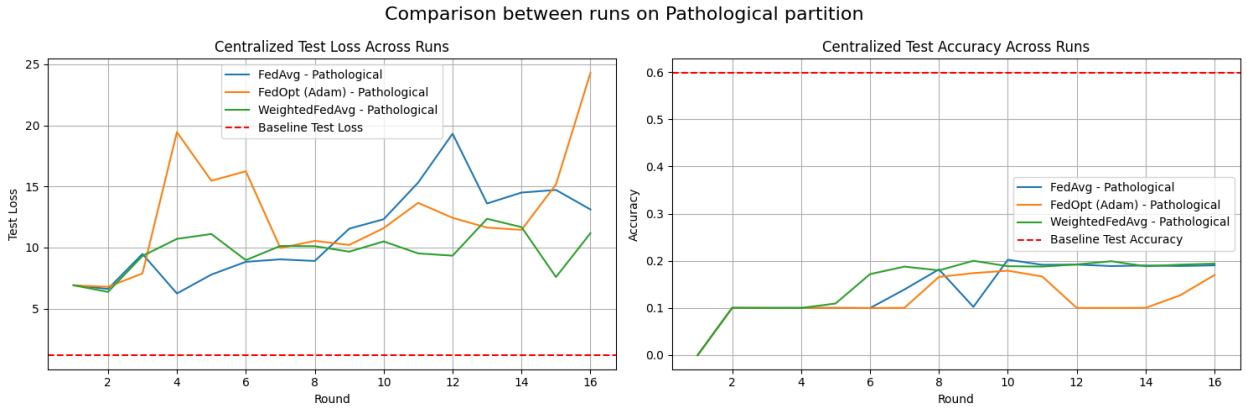


Figure 26: Comparison between various aggregation strategy on Pathological partition strategy.

5 Discussion

This section discusses the key findings from both experiments, emphasizing the implications of federated optimization strategies and data partitioning schemes on convergence behavior and model performance.

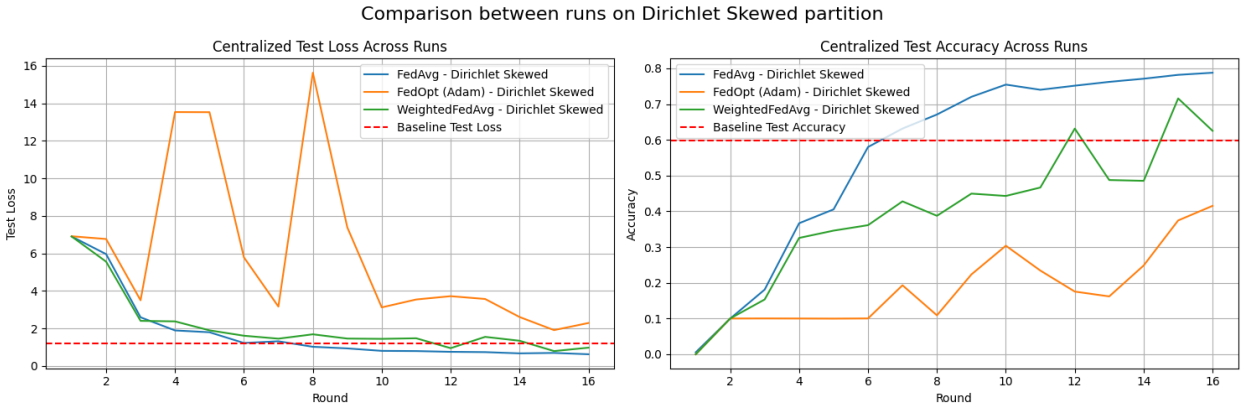


Figure 27: Comparison between various aggregation strategy on Dirichlet Skewed partition strategy.

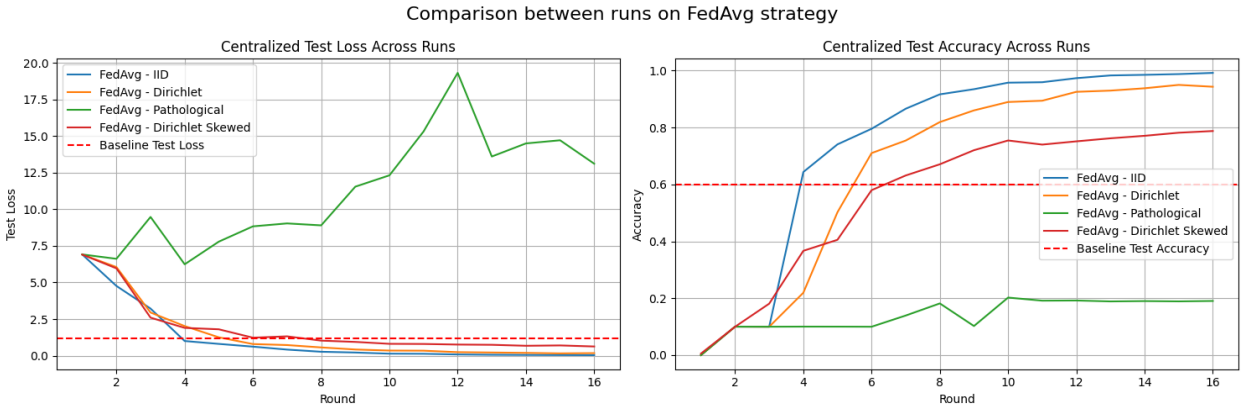


Figure 28: Comparison between various partition strategy on FedAvg aggregation strategy.

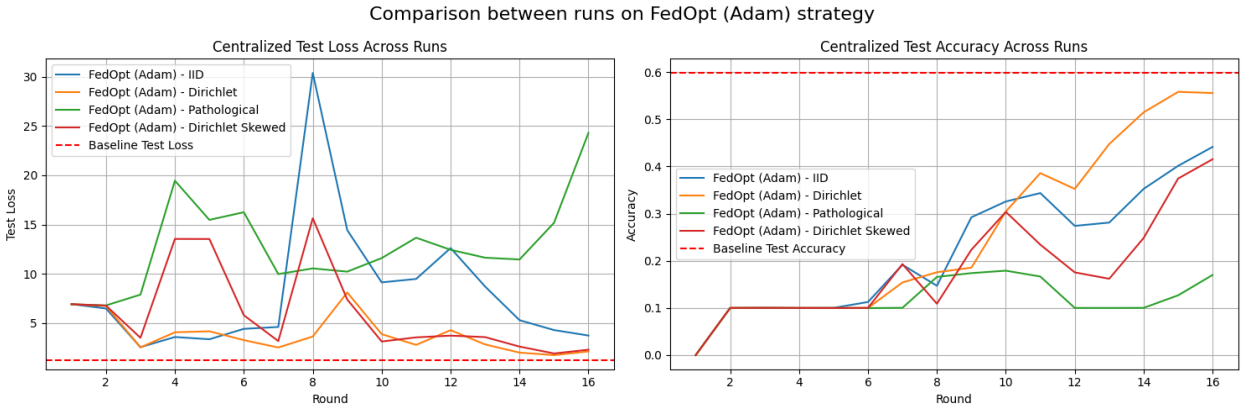


Figure 29: Comparison between various partition strategy on FedOpt (Adam) aggregation strategy.

5.1 Effect of Optimization Strategies in Experiment 1

In the Equal Hardware Experiment, all clients possessed identical computational capacity and participated at a fixed fraction (0.3) each round. Under this controlled setting, we compared FedAvg against several FedOpt variants (AdaGrad, Adam, Yogi) across three partitioning scenarios (IID,

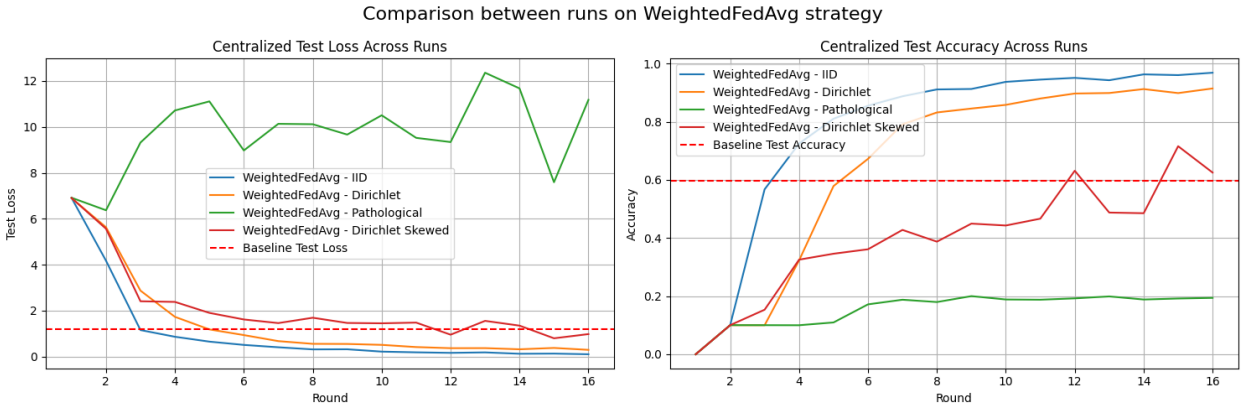


Figure 30: Comparison between various partition strategy on Weighted-FedAvg aggregation strategy.

Table 4: Test Results for Different Models and Data Partitions in Experiment 2

Model	**Partition**	**Test Loss**	**Test Accuracy**	**Rounds to Converge**
FedAvg	IID	0.0348	99.22%	10
FedOpt (Adam)	IID	3.7302	44.14%	-
Weighted FedAvg	IID	0.9913	96.88%	12
FedAvg	Dirichlet (alpha=0.5)	0.1540	95.00%	14
FedOpt (Adam)	Dirichlet (alpha=0.5)	1.7433	55.82%	-
Weighted FedAvg	Dirichlet (alpha=0.5)	0.2904	91.46%	11
FedAvg	Pathological	6.2423	20.22%	11
FedOpt (Adam)	Pathological	6.7886	16.99%	12
Weighted FedAvg	Pathological	6.3691	19.40%	6
FedAvg	Dirichlet (alpha=0.1)	0.6258	78.78%	14
FedOpt (Adam)	Dirichlet (alpha=0.1)	1.9128	41.53%	-
Weighted FedAvg	Dirichlet (alpha=0.1)	0.7943	71.58%	-

Square, Dirichlet). The main observations are:

- Under IID Partitioning (Section 4.2.3), all strategies eventually converged to very similar final accuracy (-0.92), but they differed in convergence speed:
  - FedOpt with Yogi converged fastest, reaching a stable plateau by round 24, whereas FedAvg required 47 rounds and both AdaGrad/Adam variants needed 40 rounds (Table 1).
  - Although final accuracies were nearly identical, FedOpt (Yogi) exhibited smoother validation-loss curves with fewer oscillations, suggesting that its adaptivity helps dampen fluctuations when data are identically distributed.
- In the Square Partitioning setup (Section 4.2.4), where sample sizes per client grew quadratically, we observed that:
  - FedAvg itself achieved a respectable 90.2% final accuracy but failed to formally "converge" within the 50-round window (denoted as N/A in rounds to converge).

  - Among FedOpt variants, Adam yielded the highest final accuracy (91.8%) in 39 rounds. Both AdaGrad and Yogi matched each other at 91.6% and converged in 38 rounds (Table 2).
  - The fact that all FedOpt variants outperformed FedAvg under strong data-quantity imbalance (Square) indicates that server-side adaptivity can compensate for uneven local contributions, but Adam's slightly higher accuracy suggests it better balances large- versus small-client updates in this scenario.

- Dirichlet Partitioning (Section 4.2.5) introduced both sample-size and label-distribution heterogeneity. Here:
  - FedAvg struggled, achieving only 79.6% accuracy with a high final loss (1.84) and no clear convergence.
  - FedOpt (AdaGrad) outperformed all, reaching 87.9% accuracy with a relatively low final loss (0.42) after 41 rounds. FedOpt (Adam) trailed slightly at 86.5% with a higher loss (0.87) in 40 rounds, and FedOpt (Yogi) matched FedAvg's 79.6% but required even more rounds (43).
  - These results highlight that adaptive methods are critical under statistical heterogeneity. AdaGrad's per-parameter learning-rate scaling seems particularly robust to non-IID label skews, whereas Yogi's performance fell off sharply—suggesting its momentum or correction terms may be ill-suited to extreme class-imbalance in this setting.

5.2 Impact of Partitioning Strategies in Experiment 1

By holding optimizer choice constant and varying data partitions, Experiment 1 clarifies how data heterogeneity alone influences federated training when hardware is equal:

- IID Partitioning represents the "best-case" scenario, where all clients see identically distributed samples. Under IID, all methods (FedAvg and FedOpt variants) converged to high accuracy (-0.92), with minimal variance in validation trajectories. This confirms that when data are balanced, federated gaps versus centralized training are small.

- Square Partitioning, which creates extreme sample-count heterogeneity, exposed FedAvg's slower convergence—in spite of identical local compute—because small-sample clients contributed much noisier gradients. FedOpt variants (especially Adam) mitigated this by reweighting updates so that large-sample clients do not overwhelm smaller ones entirely. Hence, while FedAvg attained strong baseline accuracy (90.2%), FedOpt's adaptivity delivered a modest -1.6% boost (91.8%).

- Dirichlet Partitioning—our most realistic simulation of non-IID label distributions—had the largest overall performance drop. FedAvg collapsed to under 80% accuracy, indicating that naive averaging across clients with divergent class labels causes severe model drift. In contrast, FedOpt (AdaGrad) recovered nearly 88%, showing that adaptive learning rates per update effectively correct for client-level class skews. This suggests that in any real-world FL deployment where clients naturally observe different label frequencies, adaptive strategies are essential just to approach baseline accuracy.

Taken together, Experiment 1 underscores that partitioning strategy alone can shift final test accuracy by over 15 percentage points (from 92% down to 79%), and that adaptive optimizers can recover much of this loss.

5.3 Effect of Optimization Strategies in Experiment 2

From the first experiments, the choice of optimization strategy had a substantial impact on model performance and convergence speed.

FedAvg consistently provided a reliable baseline across all partitioning schemes. It showed strong performance under IID conditions and was more stable than FedOpt in the full participation setup. However, its convergence was relatively slower, particularly in non-IID scenarios like Dirichlet and Pathological.

FedOpt exhibited a high degree of sensitivity to data heterogeneity. In this experiment, it often failed to converge or converged to poor solutions, especially under full participation with heterogeneous clients. This may be due to choosing only the Adam variant of this strategy, but it can also suggest that while adaptive optimizers can accelerate training, they are also more prone to instability when exposed to system-level variability or extreme data imbalance.

Weighted FedAvg, which incorporates both the number of samples and the number of epochs trained per client, proved effective in full-participation scenarios. It consistently achieved near-baseline accuracy, especially in IID and Dirichlet setups, and demonstrated the fastest convergence in the Pathological scenario. These results suggest that weighting updates not just by data volume but also by compute effort helps stabilize training in heterogeneous federated environments.

5.4 Impact of Partitioning Strategies in Experiment 2

The partitioning strategy had a significant influence on learning dynamics:

- **IID partitions** enabled the fastest and most stable convergence across all algorithms. Models reached near-centralized performance levels with minimal rounds, especially with FedAvg and Weighted FedAvg.
- **Dirichlet partitions ($\alpha = 0.5$ and 0.1)** introduced class imbalance and led to slower convergence and reduced accuracy. Still, Weighted FedAvg mitigated this effect better than FedAvg or FedOpt, suggesting that accounting for local effort in aggregation can improve robustness to statistical heterogeneity.
- **Pathological partitions**, where clients had access to only a subset of classes, posed the greatest challenge. All algorithms experienced sharp drops in accuracy and required more rounds to converge, with FedOpt (Adam) often underperforming due to poor generalization.

5.5 Convergence Behavior and System Implications in Experiment 2

We observed notable differences in how quickly and reliably each strategy converged under various conditions. FedAvg and Weighted FedAvg generally converged within 10–15 rounds in stable scenarios. In contrast, FedOpt showed delayed or unstable convergence, particularly when exposed to asynchronous or variable client training durations in Experiment 2.

The full participation setup revealed that while involving all clients each round maximizes resource utilization, it also increases the risk of synchronization issues due to heterogeneity in local computation time. Our simulation using different epoch distributions ($\mathcal{N}(5,1)$, $\mathcal{N}(10,1)$, and $\mathcal{N}(15,1)$) effectively demonstrated the semi-asynchronous behavior of real-world federated systems.

5.5.1 Recommandations

Based on our two experiments:

- **Assess Data Heterogeneity Early**: Profile client data distributions (sample counts, class frequencies) before choosing an algorithm. If non-IID effects are minor, FedOpt (AdaGrad or Yogi) can accelerate training. If severe, consider robust weighting schemes.
- **Choose Adaptive Federated Optimization Sparingly**: Use FedOpt only when clients synchronize on a fixed local-epoch schedule. If clients cannot be forced to train the same number of epochs, adaptive optimizers like Adam may introduce instability.
- **Implement Effort-Aware Aggregation for Full Participation**: In any setting where clients train asynchronously or at variable speeds, track both sample count and local-training effort. Weighted averaging based on (samples $\times$ epochs) helps mitigate stale or overly aggressive updates.
- **Balance Participation Fraction**: Requiring every client every round maximizes data usage but introduces straggler risk. A moderate fraction (e.g., 0.3 as in Experiment 1) can yield near-optimal accuracy while avoiding severe synchronization stalls.

6 Conclusions & Future Work

Our two experiments revealed that data heterogeneity is the primary driver of performance loss in federated learning. Under equal-hardware partial participation (Section 4.2), adaptive optimizers (AdaGrad, Adam, Yogi) outperformed FedAvg when data were imbalanced or labels skewed, with AdaGrad most robust under extreme Dirichlet splits and Adam excelling under sample-count imbalance. However, when all clients participated with variable local epochs (Section 4.3), these adaptive methods (especially Adam) became unstable, often failing to converge. Instead, Weighted FedAvg, which scales updates by (samples $\times$ epochs), offered the best trade-off of convergence speed and final accuracy in semi-asynchronous settings. In both regimes, no single method was universally optimal.

Some future work points would be:

- **Dynamic Client Scheduling**. Investigate participation schemes that adaptively include or exclude clients based on data skew, resource constraints, or recent contributions to mitigate straggler delays.
- **Joint Feature – Label Heterogeneity**. Extend simulations to incorporate both feature and label non-IID splits, assessing whether new correction terms or optimizer tweaks can better handle complex skews.
- **Extended Training for FedOpt in Experiment 2**: Increase the number of global rounds and local-epoch budgets when using adaptive optimizers (e.g., Adam, AdaGrad) under full participation to determine if longer training stabilizes their moment estimates and closes the accuracy gap observed in Experiment 1.
- **Hyperparameter Tuning under Variable Epochs**: Systematically explore learning-rate schedules, momentum decay, and adaptive-optimizer coefficients when clients train for different epoch counts, to identify configurations that mitigate instability in semi-asynchronous settings.
- **Hybrid Aggregation Strategies**: Combine elements of FedOpt (e.g., per-parameter learning-rate adjustments) with (samples $\times$ epochs) weighting to see if a mixed approach recovers

FedOpt's convergence speed from Experiment 1 while retaining the stability of Weighted FedAvg in Experiment 2.

By addressing these areas, future federated-learning systems can better balance accuracy, communication efficiency, and robustness to real-world heterogeneity.

References

[BTM+20] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[CWY+24] Di Chai, Leye Wang, Liu Yang, Junxue Zhang, Kai Chen, and Qiang Yang. A survey for federated learning evaluations: Goals and measures. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images., 2009.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[RAH+21] K M Jawadur Rahman, Nazma Akther, Mohammad Hasan, Ruhul Amin, Kazi Ehsan Aziz, A.K.M. Islam, Saddam Mukta, and A.K.M. Islam. Challenges, applications and design aspects of federated learning: A survey. *IEEE Access*, PP:1–1, 09 2021.

[RCZ+20] Sashank J. Reddi, Zachary B. Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecný, Sanjiv Kumar, and H. B. McMahan. Adaptive federated optimization. *ArXiv*, abs/2003.00295, 2020.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.