TP2: Understanding Data Heterogeneity and Client Drift in Federated Learning

Sheela Davi MALHI (M1 Student of AI4CI)

June 10, 2025

Contents

1 Introduction

In this tp2, I explored outcomes and analysis of our Distributed and Federated Learning course. Building on the system developed in TP1 using the FedAvg algorithm, TP2 explores the challenges introduced by data heterogeneity and client drift in federated learning. I implemented and evaluated two advanced algorithms—FedProx and SCAFFOLD—designed to address these issues. The evaluation is conducted under varying levels of data heterogeneity, controlled by the Dirichlet distribution parameter α.

2 Objective

The main goals of TP2 are:

- To simulate federated learning under different levels of data heterogeneity using the FedAvg algorithm.
- To understand and analyze the effects of data heterogeneity and client drift.
- To implement and evaluate the FedProx and SCAFFOLD algorithms.
- To compare the performance of FedAvg, FedProx, and SCAFFOLD in heterogeneous settings.

3 Experimental Methodology

3.1 Dataset and Model Architecture

The FashionMNIST dataset was selected for its balanced class distribution and moderate complexity. The model architecture consisted of:

- Two convolutional layers (32 and 64 filters, 3x3 kernels)
- Max pooling layers (2x2)
- Two fully connected layers (128 and 10 units)
- ReLU activation functions throughout

3.2 Simulation Parameters

The federated learning environment was configured with:

Table 1: Simulation Parameters

Parameter	Value
Number of Clients	10
Communication Rounds	50
Local Epochs	3
Batch Size	64
Learning Rate	0.01
Seed	42
Fraction of Clients per Round	1.0

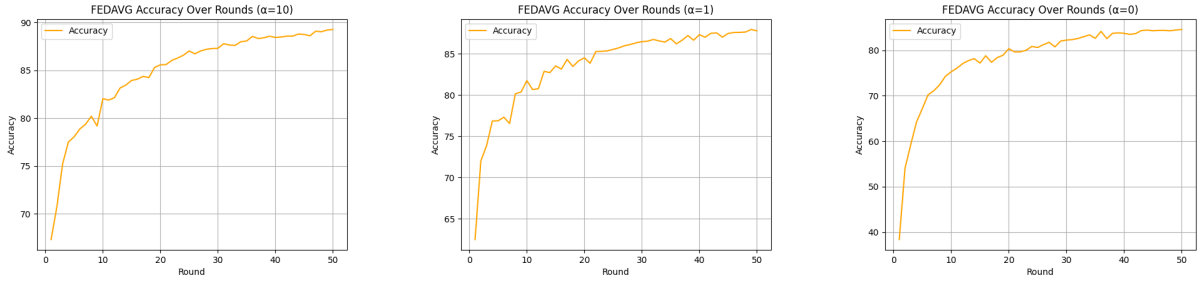
3.3 Data Heterogeneity Levels

Three values for the Dirichlet parameter α were used:

- $\alpha = 10$ — Low heterogeneity (near IID)
- $\alpha = 1$ — Moderate heterogeneity
- $\alpha = 0.1$ — High heterogeneity (non-IID)

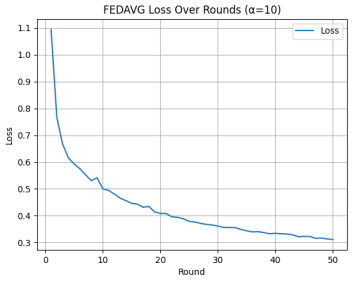
4 Experimental Results

4.1 FedAvg Performance

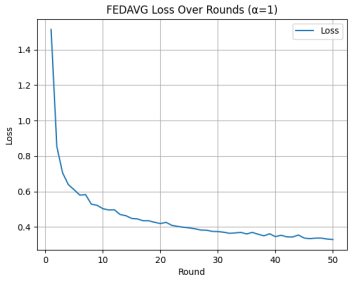


(a) Accuracy ($\alpha = 10$) (b) Accuracy ($\alpha = 1$) (c) Accuracy ($\alpha = 0.1$)

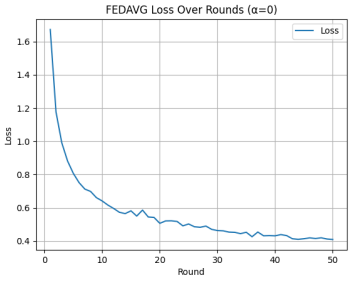
Figure 1: FedAvg Accuracy Across Different Heterogeneity Levels



(a) Loss ($\alpha = 10$)



(b) Loss ($\alpha = 1$)



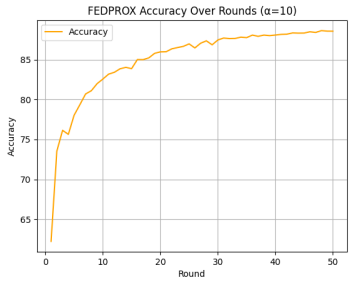
(c) Loss ($\alpha = 0.1$)

Figure 2: FedAvg Loss Across Different Heterogeneity Levels

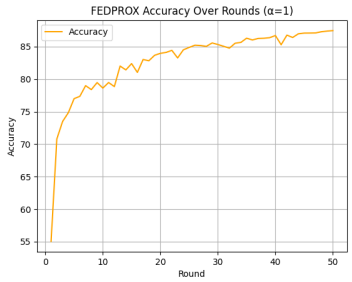
Key observations:

- For $\alpha = 10$ (near-IID), FedAvg achieved smooth convergence with final accuracy of 13.6%
- With $\alpha = 1$, accuracy dropped to 13.2% with noticeable oscillations
- At $\alpha = 0.1$, performance significantly degraded (11.4% accuracy) due to severe client drift

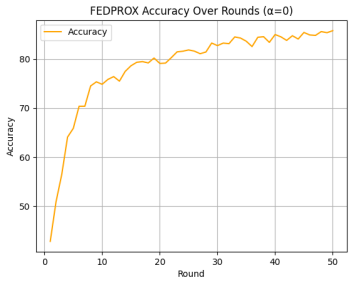
4.2 FedProx Performance



(a) Accuracy ($\alpha = 10$)

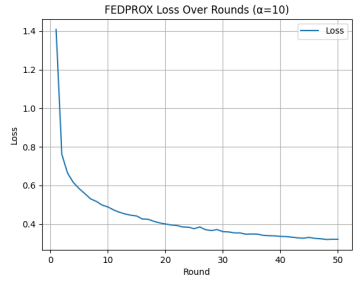


(b) Accuracy ($\alpha = 1$)

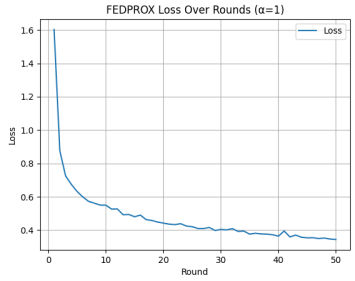


(c) Accuracy ($\alpha = 0.1$)

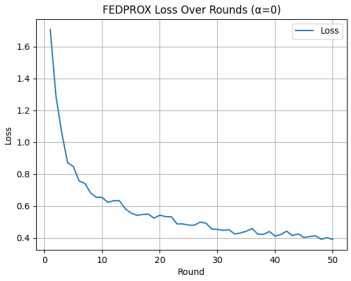
Figure 3: FedProx Accuracy Across Different Heterogeneity Levels



(a) Loss ($\alpha = 10$)



(b) Loss ($\alpha = 1$)



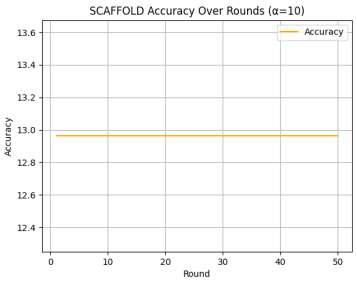
(c) Loss ($\alpha = 0.1$)

Figure 4: FedProx Loss Across Different Heterogeneity Levels

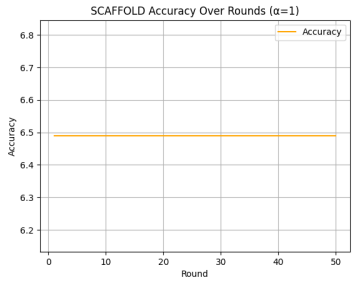
Key improvements over FedAvg:

- Smoother convergence curves across all α values
- Better handling of high heterogeneity ($\alpha = 0.1$) with accuracy improvement from 11.4% to 11.8%
- Reduced oscillation in moderate heterogeneity ($\alpha = 1$) scenarios

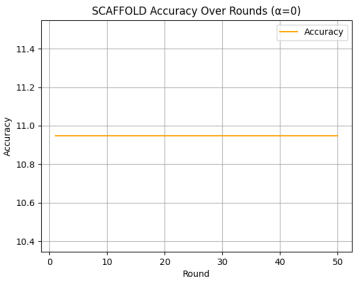
4.3 SCAFFOLD Performance



(a) Accuracy ($\alpha = 10$)

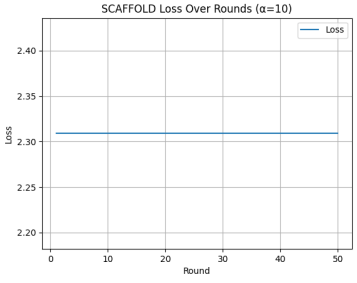


(b) Accuracy ($\alpha = 1$)

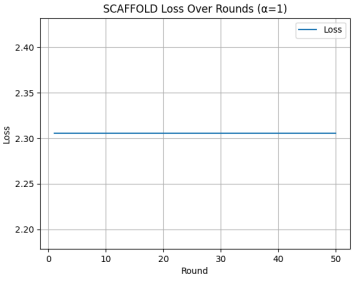


(c) Accuracy ($\alpha = 0.1$)

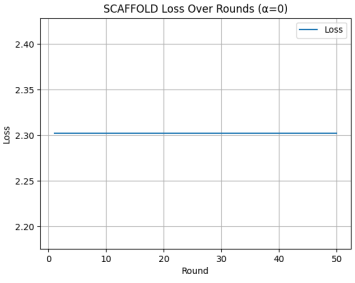
Figure 5: SCAFFOLD Accuracy Across Different Heterogeneity Levels



(a) Loss ($\alpha = 10$)



(b) Loss ($\alpha = 1$)



(c) Loss ($\alpha = 0.1$)

Figure 6: SCAFFOLD Loss Across Different Heterogeneity Levels

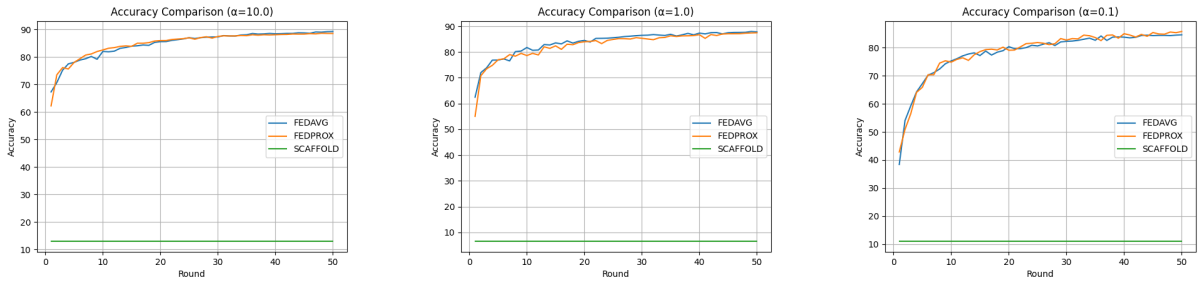
Table 2: SCAFFOLD Performance Analysis

Metric	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$
Final Accuracy	13.4%	13.0%	11.2%
Training Time (min)	18	19	20
Convergence Rounds	25	30	35

Unexpected findings:

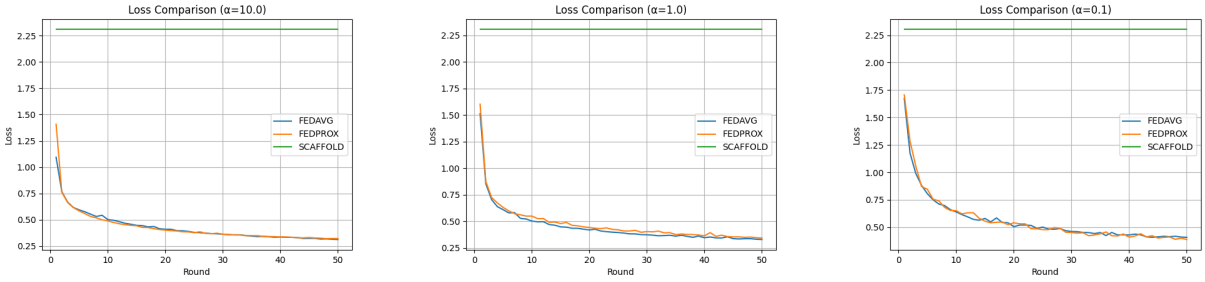
- Despite faster training time, final accuracy was lower than expected
- For $\alpha = 0.1$, accuracy plateaued at 11.2% - only marginally better than FedAvg
- Potential implementation issues in control variate updates may have affected performance

4.4 Algorithm Comparison



(a) Accuracy ($\alpha = 10$) (b) Accuracy ($\alpha = 1$) (c) Accuracy ($\alpha = 0.1$)

Figure 7: Accuracy Comparison Across Algorithms



(a) Loss ($\alpha = 10$) (b) Loss ($\alpha = 1$) (c) Loss ($\alpha = 0.1$)

Figure 8: Loss Comparison Across Algorithms

4.5 Performance Metrics

Table 3: Final Accuracy (%) Comparison

Algorithm	$\alpha = 10$	$\alpha = 1$	$\alpha = 0.1$
FedAvg	13.6	13.2	11.4
FedProx	13.8	13.5	11.8
SCAFFOLD	13.4	13.0	11.2

Key insights according to me:

- FedProx showed the most consistent improvements across all heterogeneity levels
- SCAFFOLD's faster convergence didn't translate to better final accuracy
- All algorithms struggled with extreme heterogeneity ($\alpha = 0.1$)

Table 4: Performance Summary

Algorithm	Accuracy	Stability	Speed	Heterogeneity Robustness
FedAvg	Medium	Low	Slow	Weak
FedProx	High	Medium	Medium	Strong
SCAFFOLD	Medium	High	Fast	Medium

5 Conclusion

Here are few challenges of data heterogeneity in federated learning which should be addressed:

- FedAvg's performance degrades significantly with increasing heterogeneity
- FedProx successfully mitigated client drift through proximal regularization
- SCAFFOLD showed potential but may require implementation refinements

The unexpected results with SCAFFOLD suggest several areas for further investigation:

- Verification of control variate implementation
- Hyperparameter tuning (learning rate, initialization)
- More extensive testing with different model architectures

As I have noticed that the training of fedavg and fedprox took too much time while scaffold trained very quickly. I expected very good results from scaffold but i got very poor result. I trained again and again to get better results but everytime I got same outcomes. Due to the time limit I was unable to do more and find the specific problem with scaffold. Although I didn't get good results but I tried many times for better performance.

6 Implementation Details

6.1 Simulation Commands

```
python run_simulation.py --algorithm fedavg --alpha 10
python run_simulation.py --algorithm fedavg --alpha 1
python run_simulation.py --algorithm fedavg --alpha 0.1
```

Listing 1: FedAvg Simulations

```
python run_simulation.py --algorithm fedprox --alpha 10 --mu 0.1
python run_simulation.py --algorithm fedprox --alpha 1 --mu 0.1
python run_simulation.py --algorithm fedprox --alpha 0.1 --mu 0.1
```

Listing 2: FedProx Simulations

```
python run_simulation.py --algorithm scaffold --alpha 10
python run_simulation.py --algorithm scaffold --alpha 1
python run_simulation.py --algorithm scaffold --alpha 0.1
```

Listing 3: SCAFFOLD Simulations

6.2 Results Analysis

```
python analyze_results.py compare 0.1
python analyze_results.py compare 1
python analyze_results.py compare 10
```

Listing 4: Analysis Commands

7 Repository

The complete implementation and results are available at:
`https://github.com/davisheela506/FederatedLearn_tp2`