AI4CI

Distributed and Federated Learning – TP1
Introduction to Flower Framework

Dmytro Rohovyi NTUU

23/05/2025

TP2: Understanding Data Heterogeneity and Client Drift

Ahmad Dabaja, Caina Figueiredo Pereira and Rachid Elazouzi

Introduction

In the first practical session (TP1), we implemented a customizable Federated Learning (FL) project based on the FedAvg scheme. This project allowed us to simulate federated training using various hyperparameters, including the Dirichlet distribution parameter α, which controls the degree of data heterogeneity among clients. Lower values of α result in non-identical (non-IID) distributions, where each client has access to a limited subset of classes, while higher values generate more balanced distributions across clients.

In this second session, we build upon the previously developed FL system to explore the implications of data heterogeneity, particularly its role in causing client drift — a phenomenon where local updates diverge from the global objective due to non-IID data. We will also introduce and implement two advanced FL algorithms, FedProx and SCAFFOLD, which are specifically designed to mitigate the negative impact of data heterogeneity and client drift.

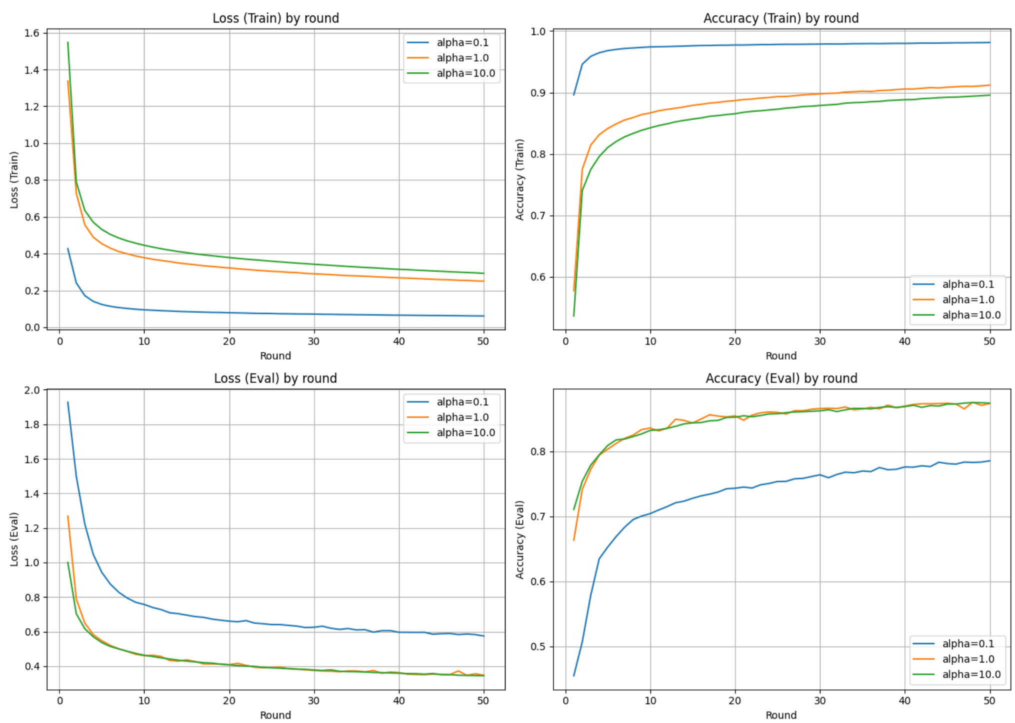
Objectives

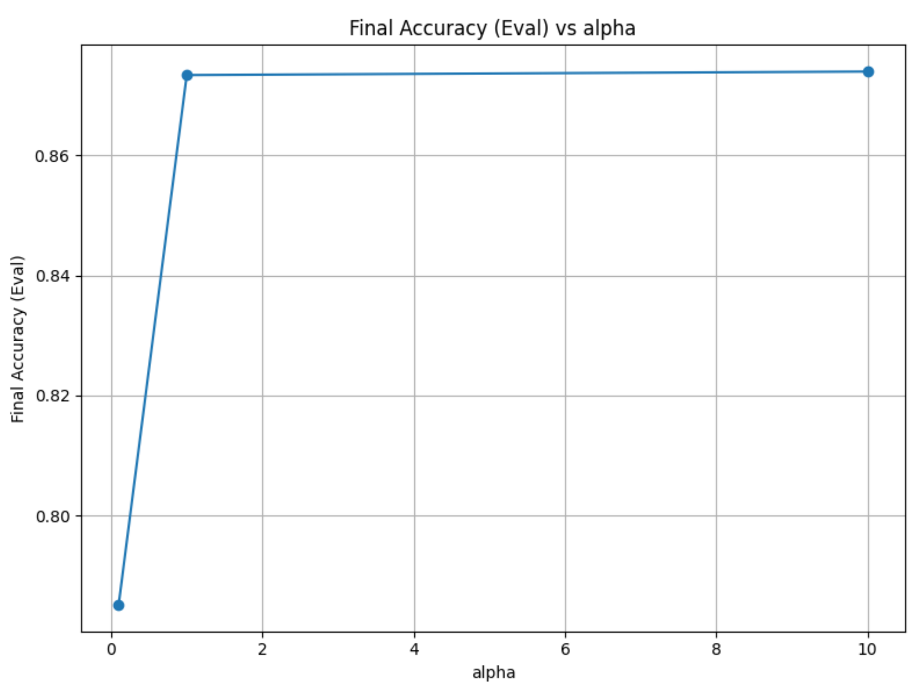
This practical session aims to deepen your understanding of the challenges introduced by data heterogeneity in Federated Learning and explore algorithmic solutions to address them. Specifically, you will:

1. Simulate multiple FedAvg training runs with varying levels of data heterogeneity.
2. Develop a conceptual understanding of data heterogeneity and client drift and how they affect model convergence and performance.
3. Explore and implement two Federated Learning algorithms: **FedProx** and **SCAFFOLD**.
4. Conduct simulations using FedProx and SCAFFOLD under different levels of data heterogeneity.
5. Evaluate and compare the performance of FedAvg, FedProx, and SCAFFOLD in terms of accuracy, loss, and convergence speed.
6. Analyze the effectiveness of each scheme in mitigating client drift and stabilizing training in heterogeneous FL settings.

Analyzing FedAvg under Different Levels of Data Heterogeneity

rounds	epoch	client	batch	lr	alpha	strategy	accuracy	loss
50	3	10	64	0.01	0.1	fedavg	0.7851549483505498	0.574692518639936
50	3	10	64	0.01	1.0	fedavg	0.8733860891295293	0.34783009097557277
50	3	10	64	0.01	10.0	fedavg	0.8739691795085381	0.34455077899391084

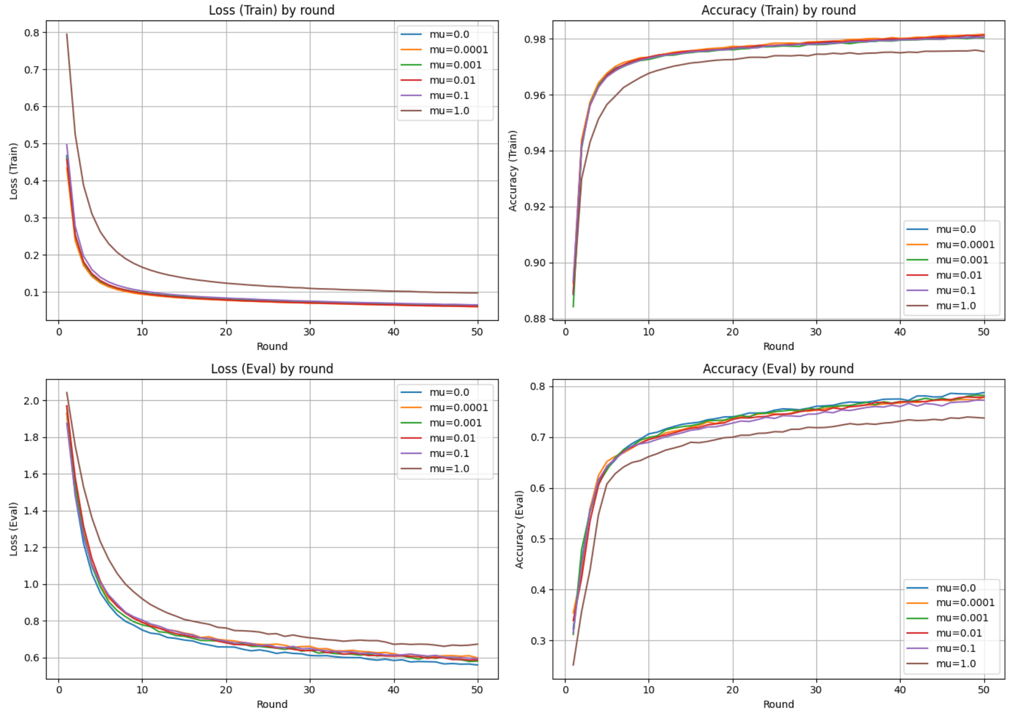




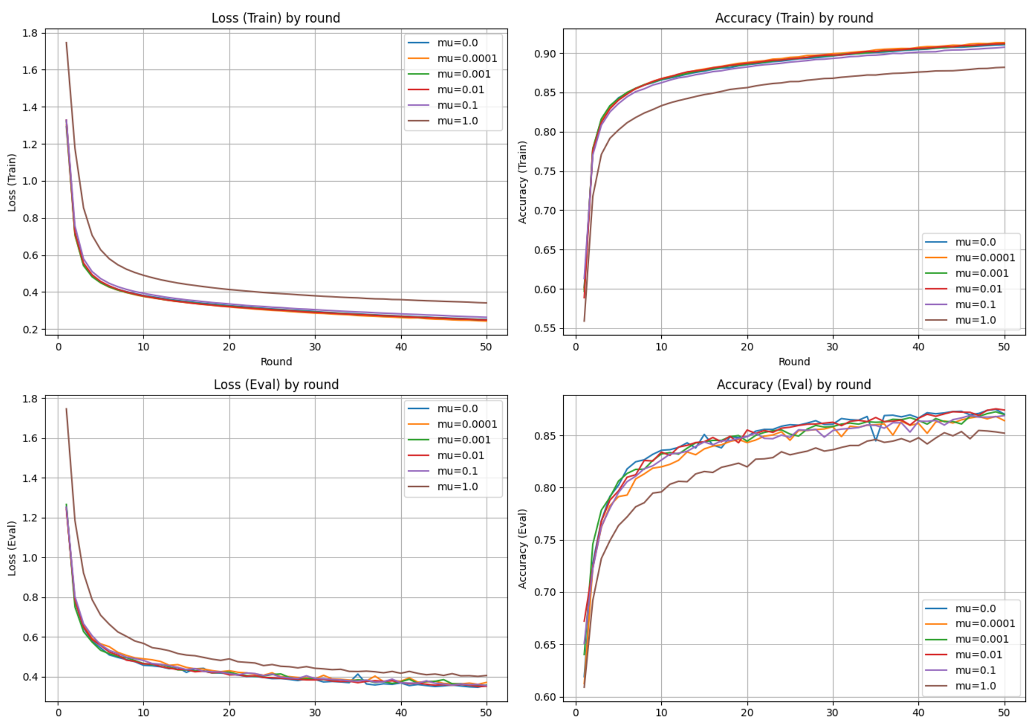
Firstly, it's clearly visible that FedAvg performs better on IID data, which was expected. What's interesting is training curves – here a=0.1(highly non-IID data) shows lowest loss curve and highest accuracy curve, which caused by quick overfitting of clients, leading to poor evaluation results.

Second interesting observation is evaluation curves for a=1 is the most unstable. a=1 means that data is already not skewed enough to prevent global model's overfitting, but at the same time it's still not IID enough to result in good and stable generalization.

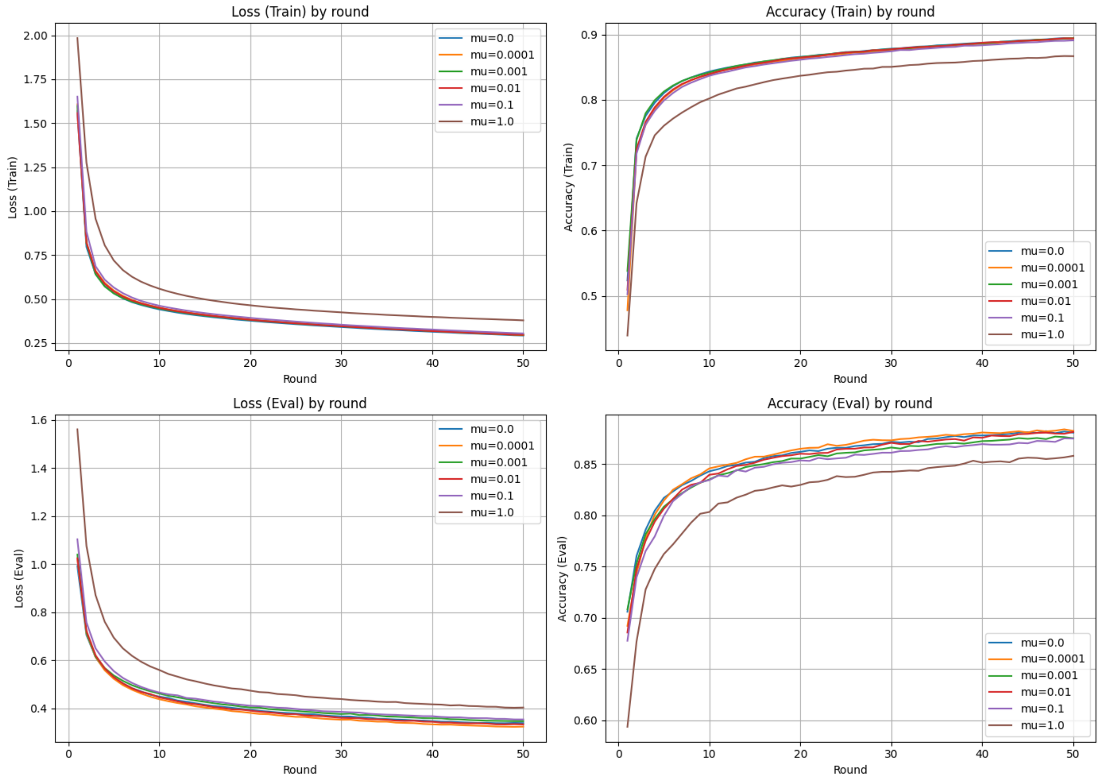
Implementing FedProx: Mitigating Client Drift through Proximal Regularization



a=0.1



a=1

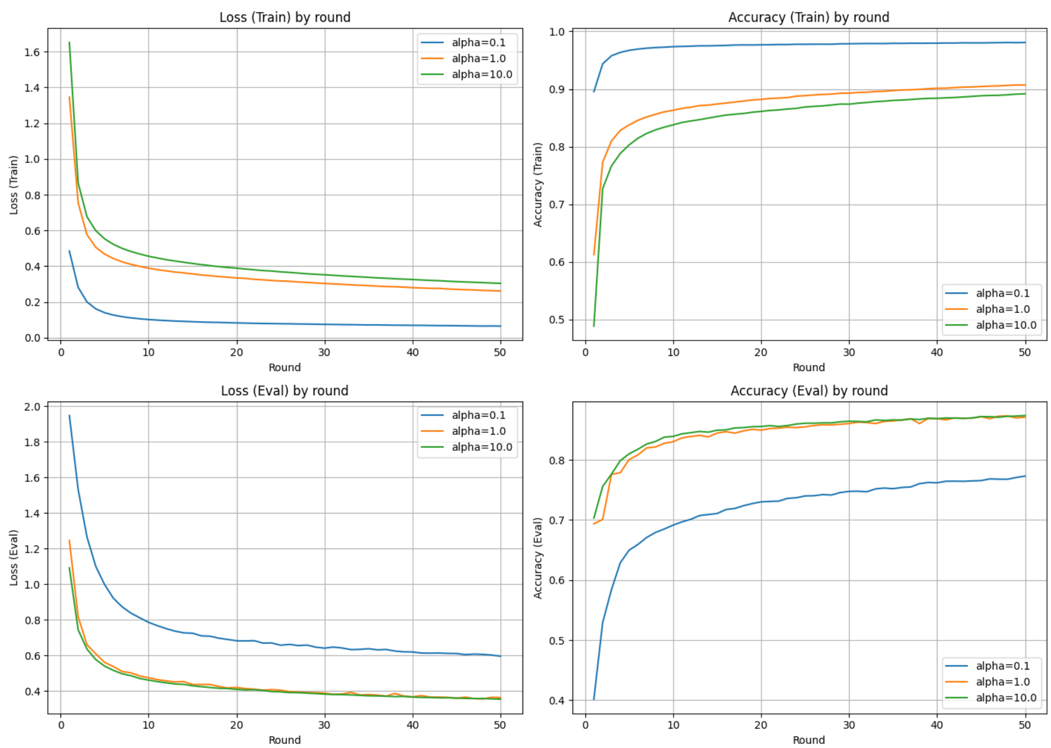


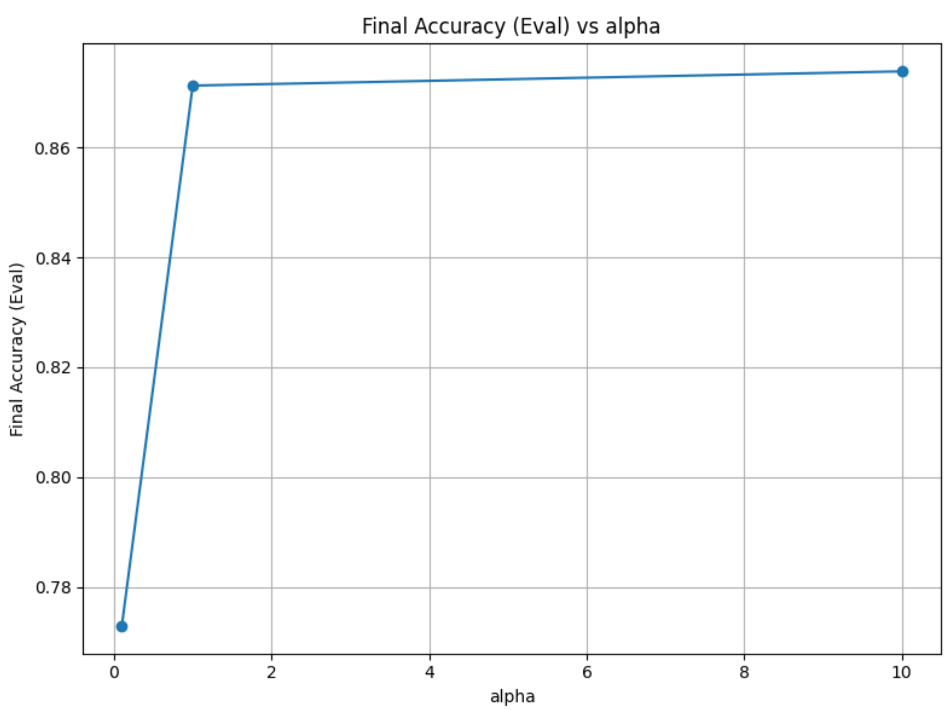
a=10

Tests for all 3 values of parameter alpha showed that “mu” don’t have significant impact on training process, unless it’s extremely high, like we see in our case where mu=1 shows lower performance. Due to this, for all future tests with FedProx strategy default mu=0.1 will be used.

FedProx normal test

rounds	epoch	client	batch	lr	alpha	strategy	mu	accuracy	loss
50	3	10	64	0.01	0.1	fedprox	0.1	0.7729923358880373	0.5963089995220259
50	3	10	64	0.01	1.0	fedprox	0.1	0.8712203248646397	0.36345692414236486
50	3	10	64	0.01	10.0	fedprox	0.1	0.8738025822573927	0.354971316562307



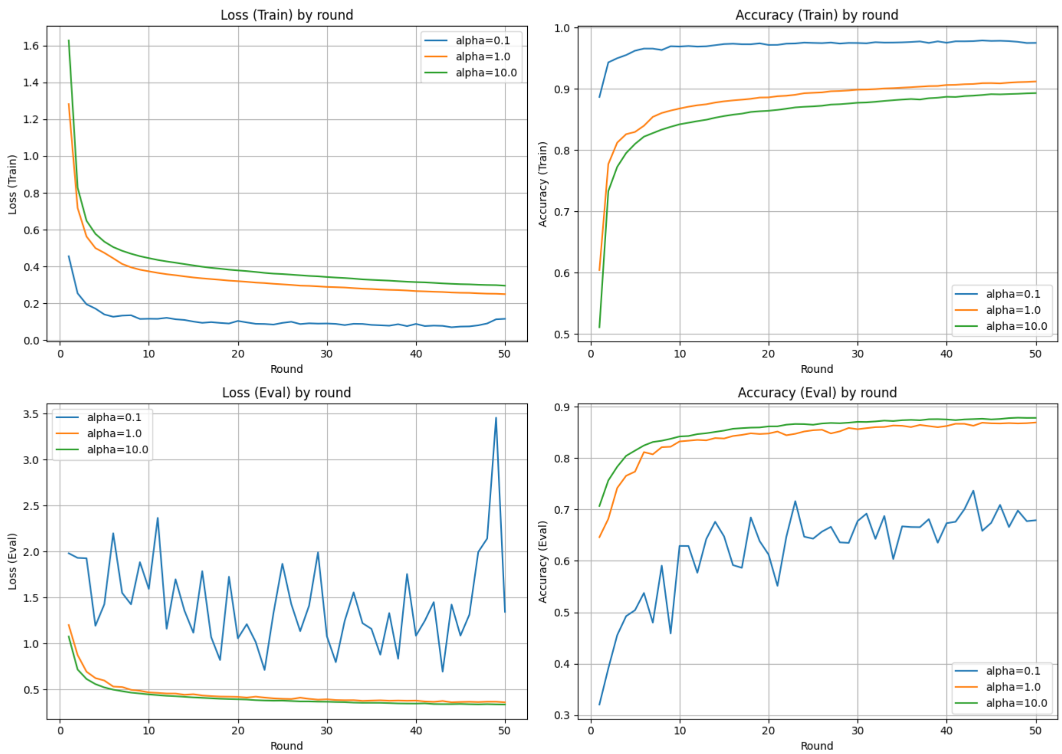


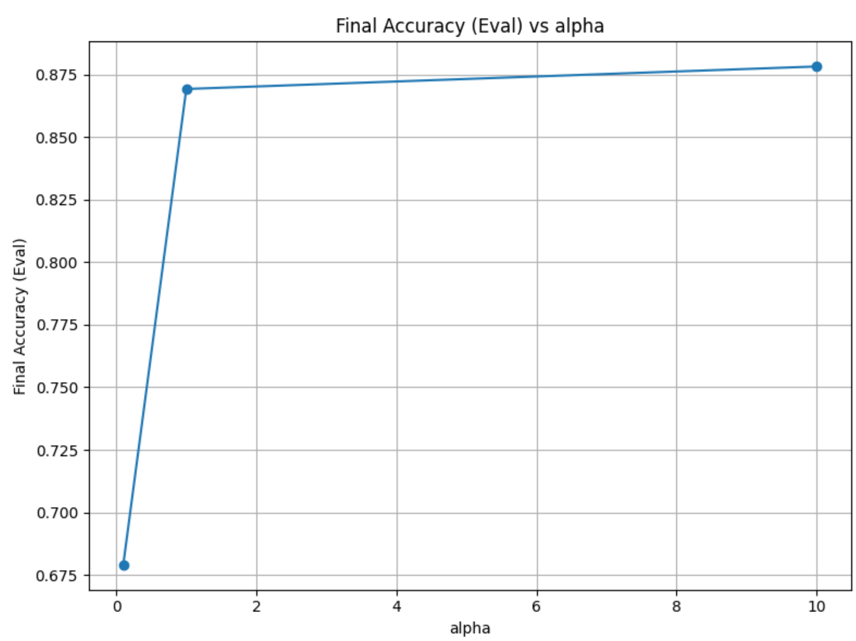
Besides expected fact that higher alpha will yield higher result we can make an interesting observation. Evaluation curves for alpha = 1 are the most unstable, which is result of data being not homogeneous enough to provide all clients with reasonable class distribution and ensure smooth local updates and high overall performance, while, at the same time, being not heterogeneous enough to lead to overfitting of all clients, lowering overall performance and leading to smooth, but poor local updates. This can be further proved by training curves.

Another interesting observation is training curves. Alpha=0.1 produces best training results(highest accuracy and lowest loss), but it performs the worst on evaluation. This signals severe overfitting of local clients, and means that each client quickly learns patterns of their local data, returning high training results, but global model fails to generalize, due to said overfitting.

Implementing SCAFFOLD: Mitigating Client Drift with Control Variates

rounds	epoch	client	batch	lr	alpha	strategy	accuracy	loss
50	3	10	64	0.01	0.1	scaffold	0.6790236587804065	1.3447741155103916
50	3	10	64	0.01	1.0	scaffold	0.8693044564764681	0.36080283325495993
50	3	10	64	0.01	10.0	scaffold	0.8783007080383174	0.33679702558998065

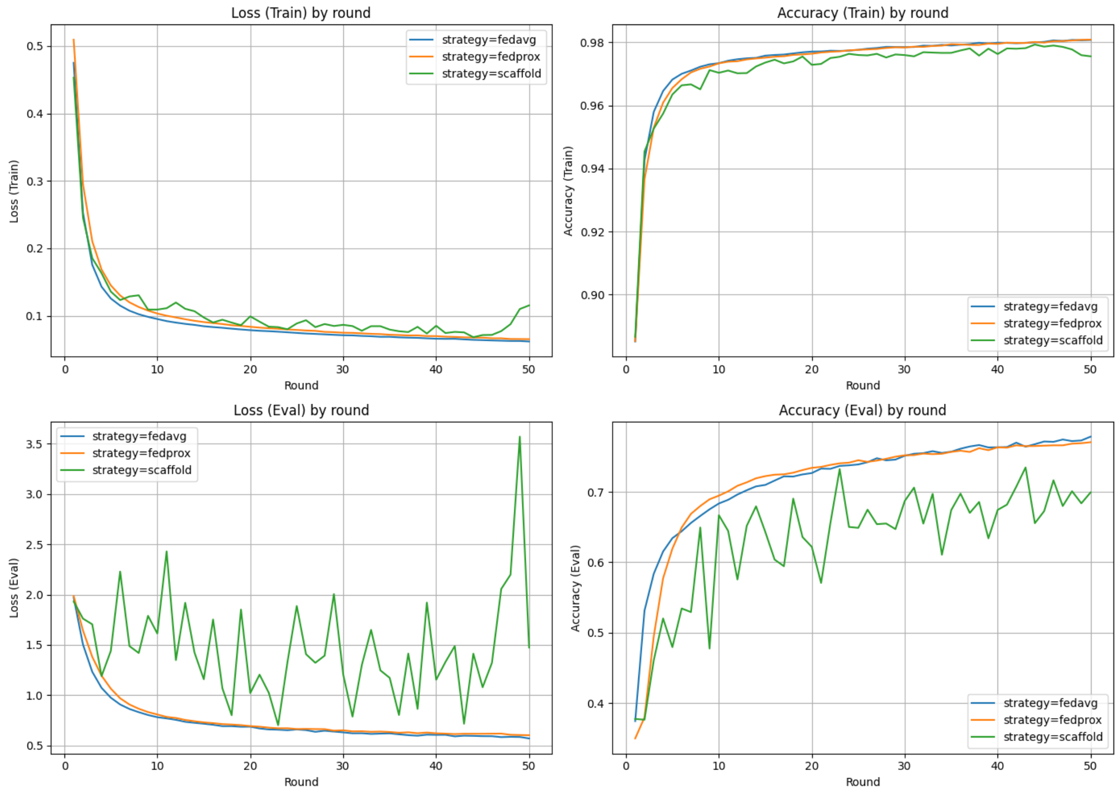




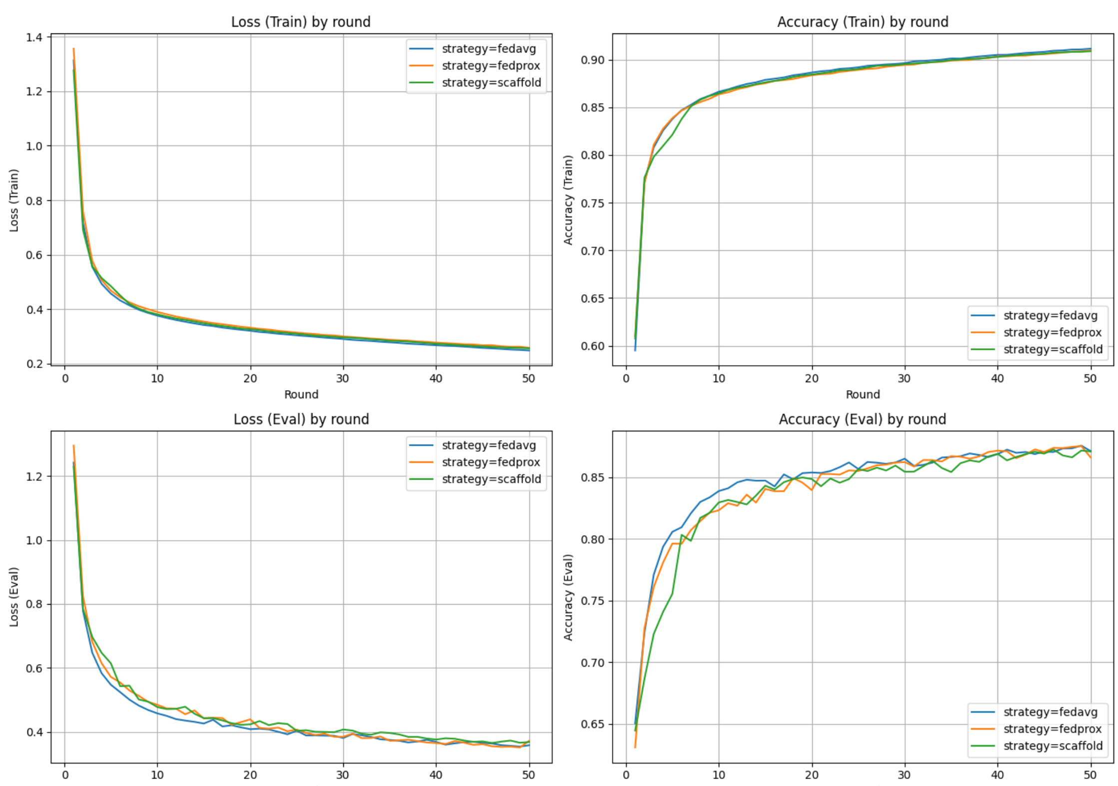
First, and most important thing visible from this data is alpha severely impacting stability of training. Alpha=10 yields smooth curves, meaning stable training, while alpha=0.1 leads to serious oscillations of evaluation curves, while providing smooth training curves. This shows main flaw of SCAFFOLD – it does not smooth deviation of global model, caused by client drift, it actively tries to correct it. For highly heterogeneous data this only makes things worse, leading to extremely unstable training and lower performance for lower values of alpha.

SCAFFOLD, due to the way it tries correcting client drift using control values, instead of lowering its impact on global model, is sensible to training hyperparameters, and requires careful tuning, which wasn’t done in our case.

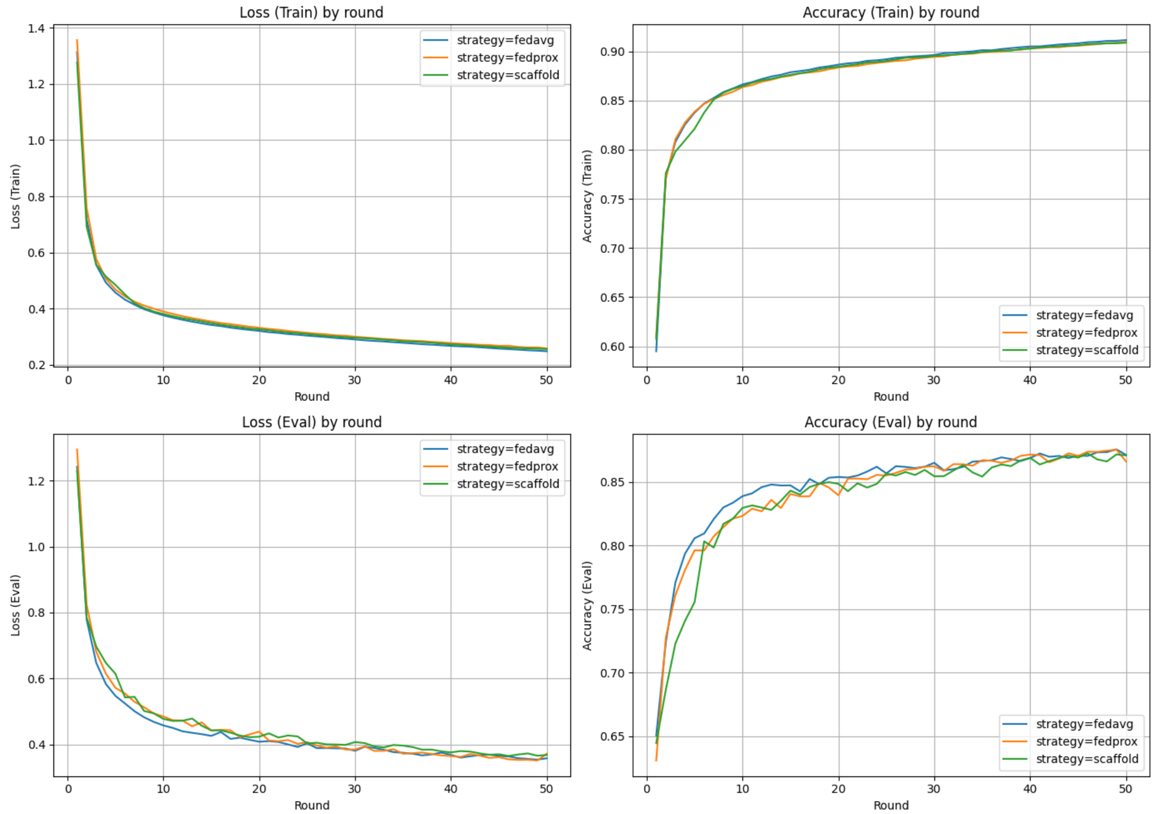
Summary



Alpha = 0.1



Alpha = 1



Alpha=10

Based on all previous observation and training/evaluation curves for both 3 algorithms we can clearly say that in our case and with our training parameters FedAvg performed best. SCAFFOLD showed worst results due to it’s reliance on carefully fine-tuned hyperparameters. FedProx performed similarly to FedAvg and provided more stable evaluation curves.