First Lab Report

Zeev Weizmann

October 2025

1 Introduction

In this lab, we studied the basics of federated learning and the FedAvg algorithm. The goal was to implement local training on clients, model aggregation, and analyze the effect of local epochs.

2 Exercise 1: UML Diagram

The first exercise required analyzing the provided codebase and producing a UML diagram of the classes and their relationships. Figure 1 shows the resulting UML diagram, which highlights the main components: `Train`, `Client`, `Learner`, `Aggregator`, and their interactions.

3 Exercise 2: Completing the Code

In the second exercise, the missing parts of the implementation were completed.

- The `Client` class: implemented `step()` for local training and `write_logs()` for reporting metrics.
- The `Aggregator` class: completed `mix()` to aggregate models and `update_clients()` to synchronize them.
- The training loop in `train.py`: finalized to perform multiple communication rounds.

This completed implementation allowed us to run experiments with FedAvg.

4 Exercise 3: Effect of Local Epochs

We analyzed the effect of the number of local epochs on the convergence and accuracy of the global model.

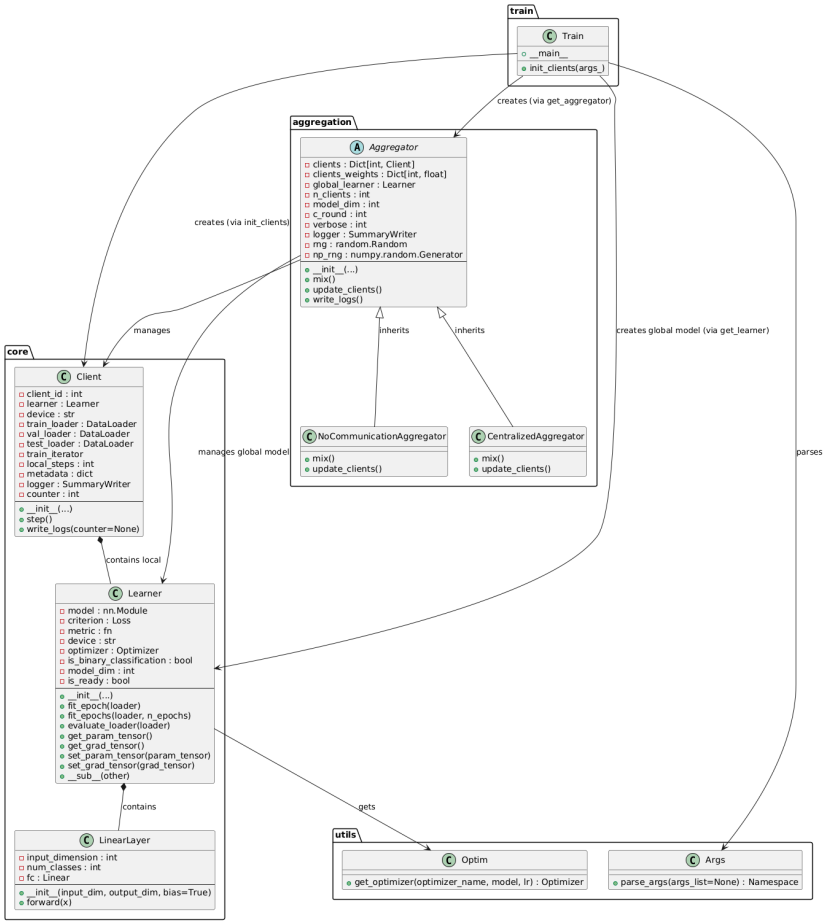


Figure 1: UML diagram of the codebase.

4.1 Experimental Setup

FedAvg was run on the MNIST dataset with different numbers of local steps:

$$1, 5, 10, 50, 100$$

Additionally, as a bonus experiment, we set the batch size equal to the dataset size and local steps = 1, which corresponds to FedSGD.

4.2 Results

Figure 2 shows the test accuracy for different local epochs. The final test accuracies are summarized in Table 1.

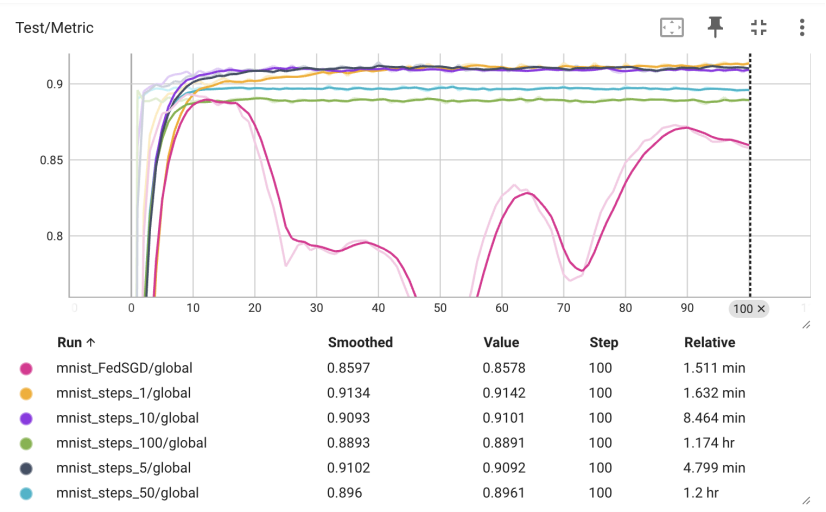


Figure 2: Effect of local epochs on test accuracy.

Setting	Final Test Accuracy
FedSGD (bz = dataset, steps=1)	0.858
Local steps = 1	0.914
Local steps = 5	0.910
Local steps = 10	0.910
Local steps = 50	0.896
Local steps = 100	0.889

Table 1: Comparison of test accuracies with different local epochs.

4.3 Discussion

We observe that:

- Accuracy is highest for 1 local step (0.914).
- Increasing the number of local epochs (50, 100) leads to lower accuracy, due to clients overfitting their local data before aggregation.
- FedSGD achieves lower accuracy compared to FedAvg with 1 epoch, since it reduces communication variance but sacrifices local adaptation (being more sensitive to random initialization, as it performs only one global update per full batch and thus lacks the adaptation provided by multiple local epochs).

Thus, choosing a moderate number of local epochs (e.g., 1–10) balances local learning and global model performance.

5 Bonus Exercise: Complex Dataset & Model

5.1 Objective

The goal of the bonus exercise was to challenge ourselves by integrating a more complex dataset and a deep learning model into the federated learning framework. Specifically:

- Integrate the CIFAR-10 dataset into the learning process.
- Implement the MobileNet model using the PyTorch framework.

5.2 Implementation

For the MNIST experiments, training on CPU was sufficient and could be executed locally without major issues. However, when moving to the more complex CIFAR-10 dataset with MobileNet, each training round on my local computer required several hours due to limited computational resources.

Initially, I used `weights=None` (training the MobileNet from scratch). However, I quickly realized that reaching competitive accuracy would require an impractically large number of communication rounds, far beyond my available resources. To address this, I switched to using pretrained weights `MobileNet_V2_Weights.DEFAULT`, which allowed me to leverage transfer learning. With this setup, training became much more efficient, and I was able to obtain meaningful results even with a limited number of rounds.

To run the experiments more effectively, I moved to Google Colab, which provides GPU acceleration and enabled me to execute the code faster. Still, Colab imposes strict usage limits, so I was able to perform only a subset of the planned runs.

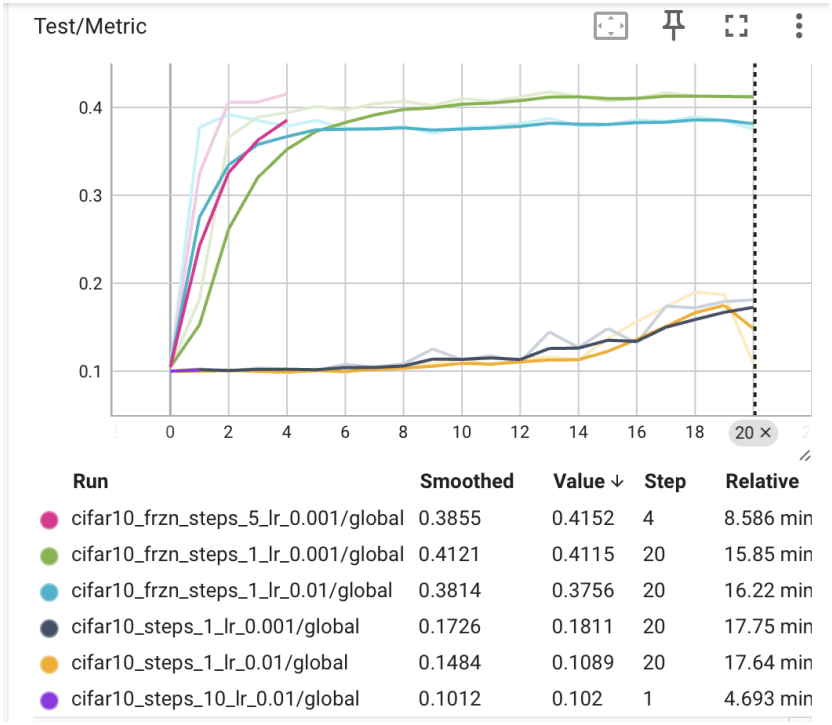
5.3 Training & Evaluation

The model was trained using the FedAvg algorithm and evaluated on CIFAR-10. I tested different hyperparameter setups, including various numbers of local steps and learning rates. The training curves are shown in Figure 3, where each run corresponds to a different configuration.

5.4 Discussion

From the experiments, we can already observe several trends:

- Freezing the feature extractor and fine-tuning only the classification head leads to faster convergence and higher accuracy (around 40%).
- A smaller learning rate ($lr = 0.001$) consistently performs better than a larger one ($lr = 0.01$), which often destabilizes training.



Run	Smoothed	Value ↓	Step	Relative
cifar10_frzn_steps_5_lr_0.001/global	0.3855	0.4152	4	8.586 min
cifar10_frzn_steps_1_lr_0.001/global	0.4121	0.4115	20	15.85 min
cifar10_frzn_steps_1_lr_0.01/global	0.3814	0.3756	20	16.22 min
cifar10_steps_1_lr_0.001/global	0.1726	0.1811	20	17.75 min
cifar10_steps_1_lr_0.01/global	0.1484	0.1089	20	17.64 min
cifar10_steps_10_lr_0.01/global	0.1012	0.102	1	4.693 min

Figure 3: Test accuracy on CIFAR-10 with MobileNet under different hyperparameter settings.

- Increasing the number of local steps helps the model reach good performance more quickly, but requires careful balance with the communication rounds to avoid overfitting.

These findings confirm that careful hyperparameter tuning and model freezing strategies are critical when moving to more complex datasets and architectures in a federated learning setup.

6 Conclusion

In this lab, we successfully implemented FedAvg, analyzed the effect of local epochs, and compared FedAvg with FedSGD on MNIST. For MNIST, we observed that frequent communication (a small number of local steps) generally leads to better convergence and accuracy.

In the bonus CIFAR-10 experiment with MobileNet, however, the situation was different. Here, transfer learning (using pretrained weights) proved essential, and the trade-off between local steps, learning rate, and resource constraints played a much stronger role. Increasing the number of local steps could accel-

erate convergence, but only when combined with a low learning rate and frozen feature extractor.

Thus, while small local steps are beneficial in simple setups such as MNIST, in more complex settings (CIFAR-10 + MobileNet) careful balancing of local training, learning rate, and model initialization is crucial for achieving meaningful results.

Code

The code for this lab is available on GitHub:

SSH: `git@github.com:ZeevWeizmann/fedcourse24_lab1.git`

HTTPS: `https://github.com/ZeevWeizmann/fedcourse24_lab1.git`

The repository contains:

TP1: Implementation for the MNIST experiments (Exercises 1–3).

TP1_bonus: Implementation for the CIFAR-10 experiments (Bonus Exercise: Complex Dataset & Model).