Part 1 (75 points)

In this homework, you will apply federated learning to solve an image classification problem. The objectives for part 1 is as follows:

- Simulate a federated learning task where multiple clients collaboratively train a global machine learning model using their local data.
- Implement `FedAvg` algorithm and investigate the effect of different hyperparameters.
- Learn and use the Ray framework to run the client updates in parallel.

1. Implement `FedAvg` for Federated EMNIST dataset[1]

Download the data using this link or the link on Canvas. For this homework, you will use a dataset that consists of handwritten digits and letters from 100 users. There are 62 classes (10 digits, 26 lowercase, 26 uppercase). The images are 28 by 28 pixels. You can customize the preprocessing of data. Use accuracy as the evaluation metric for the image classification problem.

Each user is considered as a *client*. The samples from the same user are considered the local data for this client. Implement `FedAvg` algorithm to train a global model for the *server* to classify images into the 62 classes. The server has access to the number of samples at each client.

You should:

1. Describe the data distribution among classes of the overall dataset and 5 users of your choice.
2. Split each local dataset into 80% training and 20% validation data; plot the aggregated training and validation loss and accuracy versus the number of communication rounds.
3. In each communication round, randomly select $C = 10\%$ of all the clients to perform local updates.
4. Compare the training and validation accuracy for different values of your choice of C (no more than 10%) and training epochs E for one client update.
5. Pick good hyperparameters and evaluate the model trained with the chosen hyperparameters on the held-out test data. Report the test accuracy.

Your implementation should satisfy the requirements:

- Use a 2-layer fully connected Neural Network with 128 hidden units and the RELU activation function.
- Test data should not be used in any part of the training. Local data should not be shared among clients.
- You should comply with the homework policy. You should implement the algorithm by yourself using standard Python packages without using packages specialized for federated learning.

2. Parallel Clients

The updates of the client models are performed in parallel in many real world applications. Use Ray to run client updates in parallel. Each client used per round should be created as one instance of the `Actor` object. The `Actor` object should handle local model training. Your tasks are as follows:

1. Learn the basics of Ray, especially the use of `Actor`, from the documentation here or the lab materials.
2. Sample 4 clients per round. Allocate the available resources to run parallel updates for the selected clients.
3. Plot the aggregated training and validation loss and accuracy versus the number of communication rounds. Report the performance of the trained model on the test data.

Your code should satisfy the requirements:

- Use a 2-layer fully connected Neural Network with 128 hidden units and the RELU activation function.
- Each client should use no more than 1 CPU and 1 GPU, depending on the resources available.
- `Actor` has large memory overhead. You should create instances only for the selected clients within each round.
- You should only use functions of Ray Core for parallel computing without packages specialized for federated learning.

Submission

Submit a report of the required plots and brief answers to the questions in a .pdf file on Canvas. Submit the code to train your model and generate the results, and a README file of the instruction to run your code on GitHub.

Part 2 (25 points)

Part 2 will be released later.

References

[1] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.