Assignment 2

Big Data Processing Assignment: Implementing Federated Averaging (FedAvg)

Dataset: MNIST

The dataset for this assignment is the **MNIST dataset**, a widely used benchmark in machine learning. MNIST consists of **60,000 training images** and **10,000 testing images** of handwritten digits (0-9), each represented as a 28x28 grayscale image. It is ideal for evaluating image classification models due to its simplicity and accessibility, making it a perfect choice for this assignment.

Predefined Code: Local Client Distribution and Testing

1. For accurate evaluation, code for distributing the MNIST dataset to **local clients (or local devices)** has already been provided. Each client will have its own local subset of the dataset to simulate federated learning.
2. Additionally, a **testing function** to evaluate the global model at each training round is also included. This ensures that the accuracy of the model can be monitored as training progresses.

Important Note:
The code for **data distribution to local clients** and the **testing function** must **not** be modified under any circumstances. These components are critical for the consistency and fairness of the evaluation process. Any alterations will invalidate the results and will not be accepted.

Tasks

Task 1:

- Implement the following two functions to complete the FedAvg framework:
    1. **average_weights(selected_models)**: A function to aggregate the model weights from selected clients during federated learning.
    2. **federated_training(num_rounds, num_clients, client_fraction, local_epochs, train_loaders, test_loader, lr=0.001)**: A function to perform federated training by coordinating clients, updating the global model, and testing it after each round.
- After completing the implementation, document the training results for the federated learning process.

Task 2:

- Set the number of participants (**num_clients**) to **60**, local epochs to **1**, and the learning rate to **0.001**.
- Train the model for **20, 30, and 40 rounds**, and plot the **training results** as a graph with:
    - **x-axis**: Number of rounds
    - **y-axis**: Accuracy
- Discuss the outcomes and explain why these results were observed.

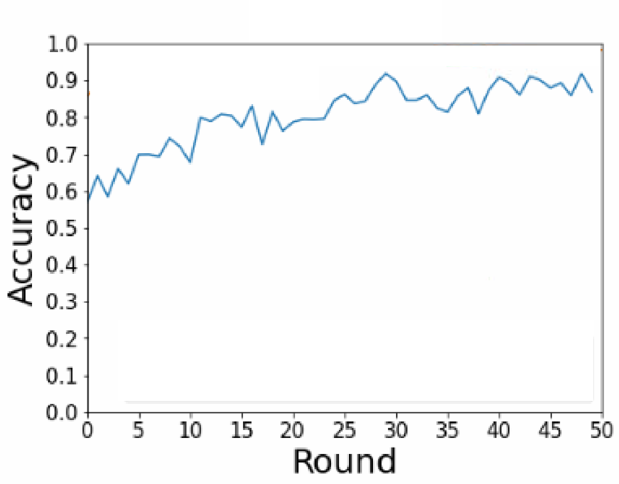
Task 3:

- Set the number of participants (**num_clients**) to **60**, learning rate to **0.001**, and the number of rounds to **20**.
- Train the model with **local epochs** set to **1, 5, and 10**, and plot the **training results** as a graph with:
    - **x-axis**: Number of rounds
    - **y-axis**: Accuracy
- Discuss the outcomes and explain the effect of changing the number of local epochs on the learning process.

Task 4:

- Set the number of participants (**num_clients**) to **100**, learning rate to **0.001**, number of rounds to **20**, and local epochs to **1**.
- Train the model with **client_fraction** set to **0.01, 0.05, and 0.1**, and plot the **training results** as a graph with:
    - **x-axis**: Number of rounds
    - **y-axis**: Accuracy
- Discuss the outcomes and explain how varying the fraction of participating clients affects the learning process.

An Example of Round-Accuracy Graph



This assignment provides hands-on experience in implementing federated learning using the FedAvg algorithm. Make sure to follow the guidelines strictly, and ensure your submission includes:

1. **Complete implementation of the functions** (fully functional code will be tested to verify correctness).
2. **Graphs for each task as specified**.
3. A **PDF report (or MS Word, *.docx)** containing the graphs and a detailed discussion and analysis of the results for each task.

Important Note: Submissions with non-functional code or incorrect file formats (e.g., not in PDF) will result in a deduction of points.