Assignment 4

Students' Names:

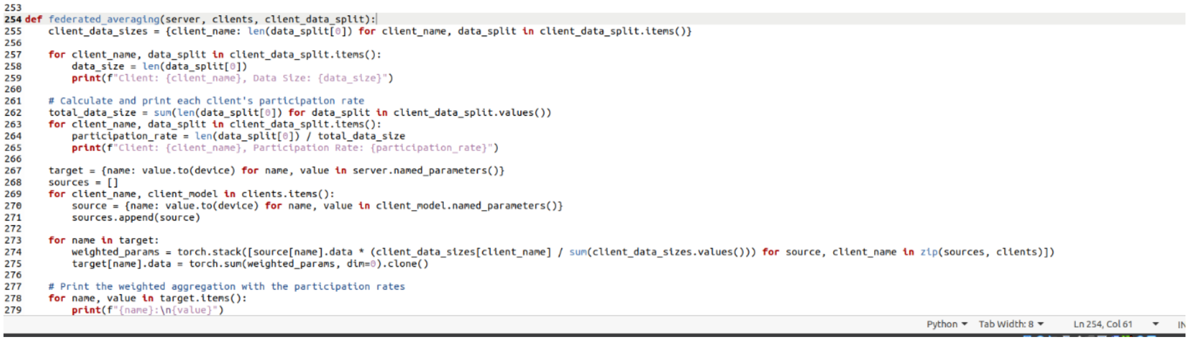
1-Amira Abu Issa
2-Heba Mostafa
3-Aya Metwally

Date: 7/10/2023

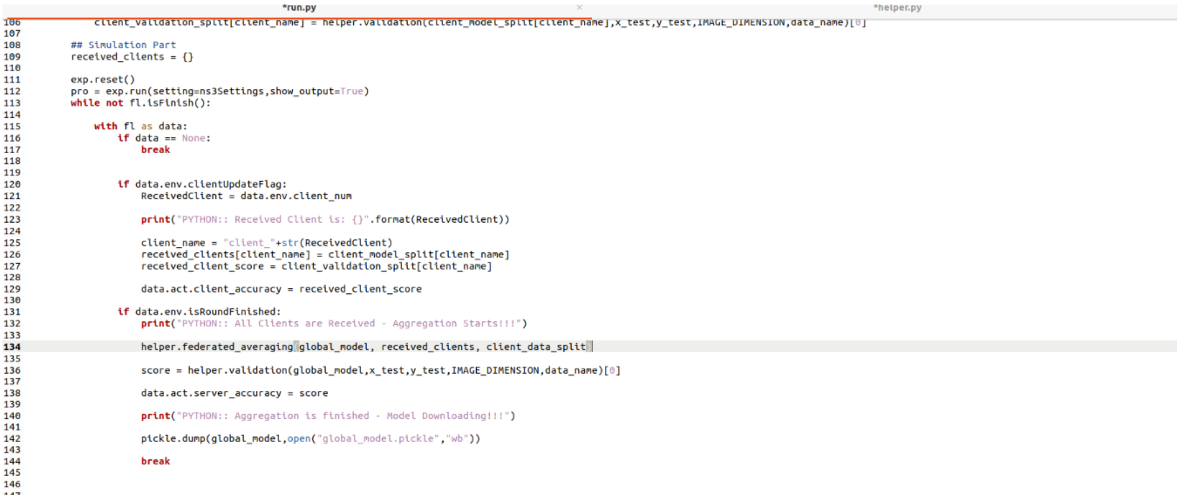
Task federated learning environment

1-1- Modifications in code

1. Apply weighted Averaging aggregation methods regarding their data size.

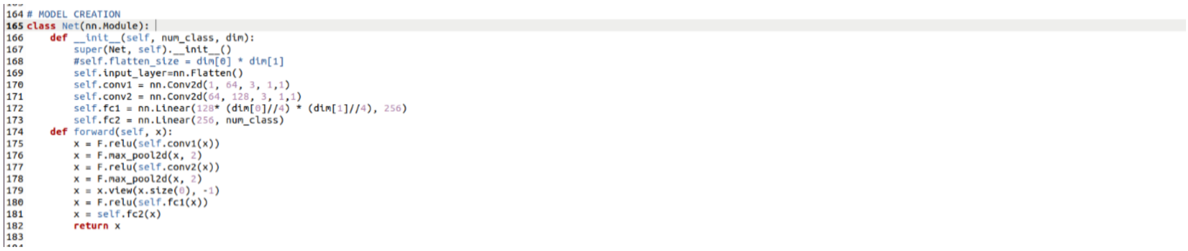


```
def federated_averaging(server, clients, client_data_split):
    client_data_sizes = {client_name: len(data_split[0]) for client_name, data_split in client_data_split.items()}

    for client_name, data_split in client_data_split.items():
        data_size = len(data_split[0])
        print(f"Client: {client_name}, Data Size: {data_size}")

    # Calculate and print each client's participation rate
    total_data_size = sum(len(data_split[0]) for data_split in client_data_split.values())
    for client_name, data_split in client_data_split.items():
        participation_rate = len(data_split[0]) / total_data_size
        print(f"Client: {client_name}, Participation Rate: {participation_rate}")

    target = {name: value.to(device) for name, value in server.named_parameters()}
    sources = []
    for client_name, client_model in clients.items():
        source = {name: value.to(device) for name, value in client_model.named_parameters()}
        sources.append(source)

    for name in target:
        weighted_params = torch.stack([source[name].data * (client_data_sizes[client_name] / sum(client_data_sizes.values())) for source, client_name in zip(sources, clients)])
        target[name].data = torch.sum(weighted_params, dim=0).clone()

    # Print the weighted aggregation with the participation rates
    for name, value in target.items():
        print(f"{name}:\n{value}")
```



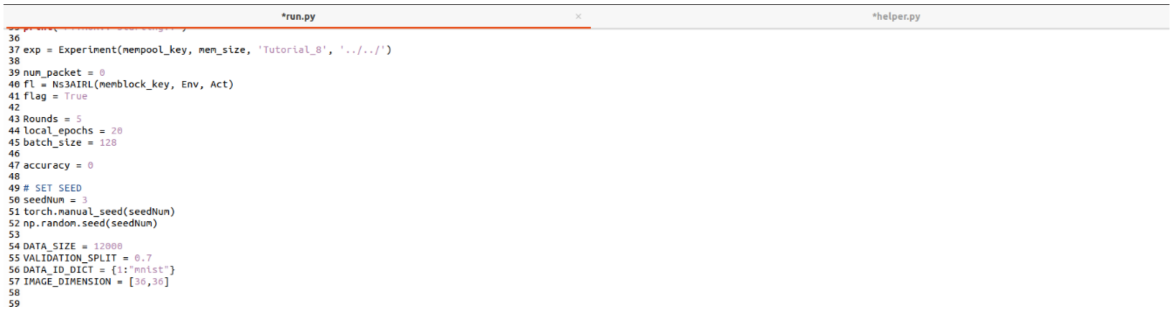
```
            client_validation_split[client_name] = helper.validation(client_model_split[client_name],x_test,y_test,IMAGE_DIMENSION,data_name)[0]

        ## Simulation Part
        received_clients = {}

        exp.reset()
        pro = exp.run(setting=ns3Settings,show_output=True)
        while not fl.isFinish():

            with fl as data:
                if data == None:
                    break


                if data.env.clientUpdateFlag:
                    ReceivedClient = data.env.client_num

                    print("PYTHON:: Received Client is: {}".format(ReceivedClient))

                    client_name = "client_"+str(ReceivedClient)
                    received_clients[client_name] = client_model_split[client_name]
                    received_client_score = client_validation_split[client_name]

                    data.act.client_accuracy = received_client_score

                if data.env.isRoundFinished:
                    print("PYTHON:: All Clients are Received - Aggregation Starts!!!")

                    helper.federated_averaging(global_model, received_clients, client_data_split)

                    score = helper.validation(global_model,x_test,y_test,IMAGE_DIMENSION,data_name)[0]

                    data.act.server_accuracy = score

                    print("PYTHON:: Aggregation is finished - Model Downloading!!!")

                    pickle.dump(global_model,open("global_model.pickle","wb"))

                    break
```

2. Creating Advance deep learning architecture CNN with 2 Convolutional and 2 dense layers. Unit size is up to you.



```
# MODEL CREATION
class Net(nn.Module):
    def __init__(self, num_class, dim):
        super(Net, self).__init__()
        #self.flatten_size = dim[0] * dim[1]
        self.input_layer=nn.Flatten()
        self.conv1 = nn.Conv2d(1, 64, 3, 1,1)
        self.conv2 = nn.Conv2d(64, 128, 3, 1,1)
        self.fc1 = nn.Linear(128* (dim[0]//4) * (dim[1]//4), 256)
        self.fc2 = nn.Linear(256, num_class)
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

3. Define each Client’s data size



```
exp = Experiment(mempool_key, mem_size, 'Tutorial_8', '../../')

num_packet = 0
fl = Ns3AIRL(memblock_key, Env, Act)
flag = True

Rounds = 5
local_epochs = 20
batch_size = 128

accuracy = 0

# SET SEED
seedNum = 3
torch.manual_seed(seedNum)
np.random.seed(seedNum)

DATA_SIZE = 12000
VALIDATION_SPLIT = 0.7
DATA_ID_DICT = {1:"mnist"}
IMAGE_DIMENSION = [36,36]


```

1-2- Report

Our code implements the basic functionality for training and aggregating models in a federated learning setting using different datasets. It allows for distributed training and model aggregation across multiple clients, which is useful for implementing machine learning algorithms in smart city environments.

Steps of code:

1- Data Preparation

2- Model Creation:

- The code defines a neural network model using the Net class. It consists of convolutional and fully connected layers.
- The model is created with a specified number of output classes and input dimensions.

3- Training:

  - The train_local function performs local training on a client's data using the specified model, data, labels, client name, number of epochs, and batch size.
  - It uses the Adam optimizer and cross-entropy loss for training.
  - The training process is done in batches, and the model parameters are updated using backpropagation.

4- Validation:

The validation function evaluates the trained model on test data.

It reshapes the test data, transforms and normalizes it.

The model's outputs are calculated for the test data, and the overall accuracy is computed.

5- Synchronization & Aggregation:

- The syncronize_with_server function updates the client's model parameters with the server's parameters.
- The federated_averaging function performs federated averaging to aggregate the model parameters from multiple clients.
- It calculates the participation rate of each client based on the size of its data.
- The model parameters are averaged based on the participation rates and assigned to the server's model.

Our code also integrates AI training and simulation components to implement federated learning in a smart city environment using ns3. It trains client models locally, performs federated averaging for model aggregation, and communicates with the ns3 simulation for synchronization and result retrieval.

- Importing Dependencies:
  - The code imports necessary libraries and modules, including the py_interface module for communication between Python and ns3, the ctypes module for working with C data types, and other custom helper modules.
- Environment and Action Structures:
  - The code defines two structures: Env and Act.
  - The Env structure contains information about the environment, such as the number of clients, update flags, and round status.

  - The Act structure represents the actions taken in response to the environment, including client and server accuracy.
- Initialization and Setup:
  - The code initializes the experiment and sets up the necessary parameters, such as memory pool and block keys.
  - It creates an instance of the Experiment class and sets the memory pool and block keys.
- Data and Model Setup:
  - The code defines settings related to the dataset and model.
  - It specifies the dataset name, number of clients, validation split, data size, and image dimensions.
  - The code prepares the dataset and splits it into client-specific data using the split_data function from the helper module.
  - It creates client models and a global server model using the Net class from the helper module.
- Training and Simulation Loop:
  - The code runs a loop for a specified number of rounds.
  - Inside each round, it performs the following steps:

  1- AI Part:

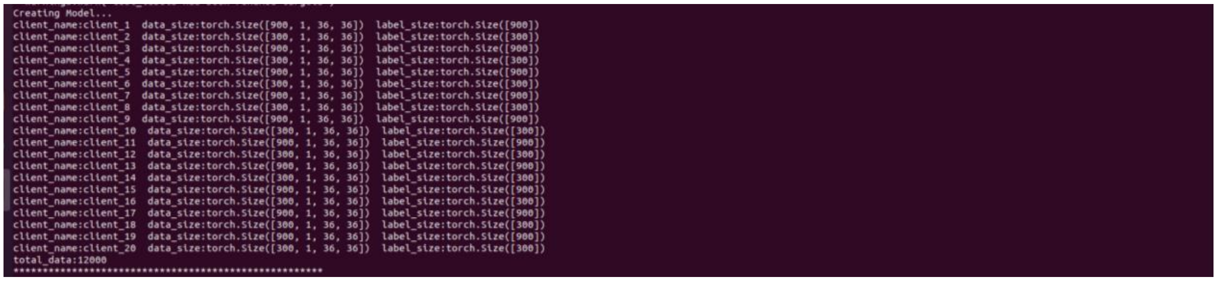
  - The client models are synchronized with the global server model using the syncronize_with_server function from the helper module.
  - Local training is performed on each client's data using the train_local function from the helper module.
  - The accuracy of each client model on the validation data is calculated using the validation function from the helper module.

  2- Simulation Part:

  - The simulation is reset using the reset method of the experiment.
  - The simulation is run with the specified ns3 settings.
  - The received client updates and round status are processed.
  - The federated averaging is performed to aggregate the client models using the federated_averaging function from the helper module.
  - The server model's accuracy is calculated on the validation data using the validation function.
  - The global model is saved to a pickle file for downloading.

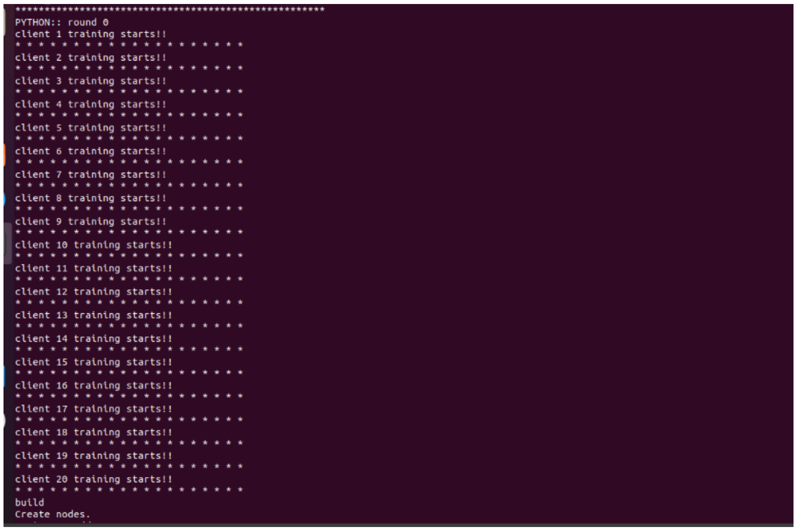
1-3- Results

- Finds each Client’s data size and print them for proper client

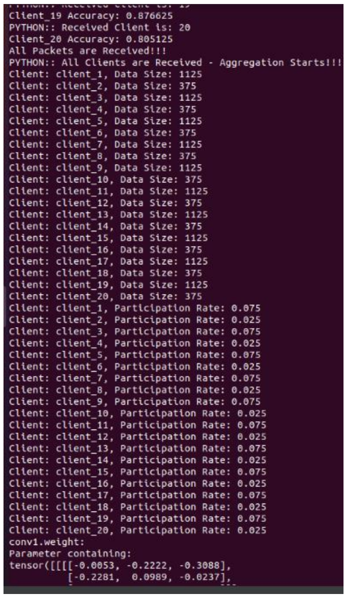


```
Creating Model...
client_name:client_1  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_2  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_3  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_4  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_5  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_6  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_7  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_8  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_9  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_10  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_11  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_12  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_13  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_14  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_15  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_16  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_17  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_18  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
client_name:client_19  data_size:torch.Size([900, 1, 36, 36])  label_size:torch.Size([900])
client_name:client_20  data_size:torch.Size([300, 1, 36, 36])  label_size:torch.Size([300])
total_data:12000
*****************************************************
```

- Calculate each client’s participation rate and print them
  For round 0

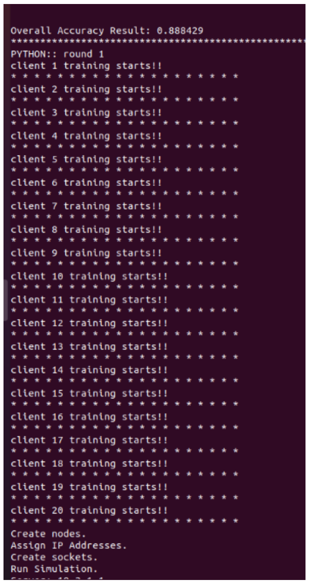


```
*****************************************************
PYTHON:: round 0
client 1 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 2 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 3 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 4 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 5 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 6 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 7 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 8 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 9 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 10 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 11 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 16 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 17 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 18 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 19 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 20 training starts!!
* * * * * * * * * * * * * * * * * * * *
build
Create nodes.
```

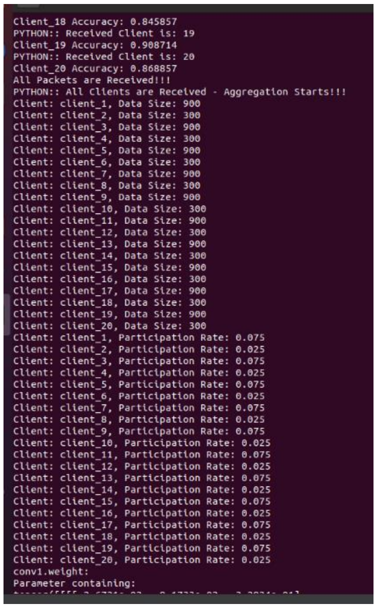


```
PYTHON:: Received Client is: 19
Client_19 Accuracy: 0.876625
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.805125
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
Client: client_1, Data Size: 1125
Client: client_2, Data Size: 375
Client: client_3, Data Size: 1125
Client: client_4, Data Size: 375
Client: client_5, Data Size: 1125
Client: client_6, Data Size: 375
Client: client_7, Data Size: 1125
Client: client_8, Data Size: 375
Client: client_9, Data Size: 1125
Client: client_10, Data Size: 375
Client: client_11, Data Size: 1125
Client: client_12, Data Size: 375
Client: client_13, Data Size: 1125
Client: client_14, Data Size: 375
Client: client_15, Data Size: 1125
Client: client_16, Data Size: 375
Client: client_17, Data Size: 1125
Client: client_18, Data Size: 375
Client: client_19, Data Size: 1125
Client: client_20, Data Size: 375
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.025
Client: client_3, Participation Rate: 0.075
Client: client_4, Participation Rate: 0.025
Client: client_5, Participation Rate: 0.075
Client: client_6, Participation Rate: 0.025
Client: client_7, Participation Rate: 0.075
Client: client_8, Participation Rate: 0.025
Client: client_9, Participation Rate: 0.075
Client: client_10, Participation Rate: 0.025
Client: client_11, Participation Rate: 0.075
Client: client_12, Participation Rate: 0.025
Client: client_13, Participation Rate: 0.075
Client: client_14, Participation Rate: 0.025
Client: client_15, Participation Rate: 0.075
Client: client_16, Participation Rate: 0.025
Client: client_17, Participation Rate: 0.075
Client: client_18, Participation Rate: 0.025
Client: client_19, Participation Rate: 0.075
Client: client_20, Participation Rate: 0.025
conv1.weight:
Parameter containing:
tensor([[[[-0.0053, -0.2222, -0.3088],
          [-0.2281,  0.0989, -0.0237],
```

For round 1

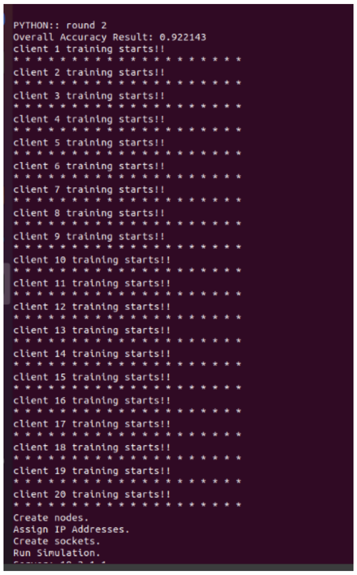


```
Overall Accuracy Result: 0.888429
************************************************************
PYTHON:: round 1
client 1 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 2 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 3 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 4 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 5 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 6 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 7 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 8 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 9 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 10 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 11 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 16 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 17 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 18 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 19 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 20 training starts!!
* * * * * * * * * * * * * * * * * * * *
Create nodes.
Assign IP Addresses.
Create sockets.
Run Simulation.
```

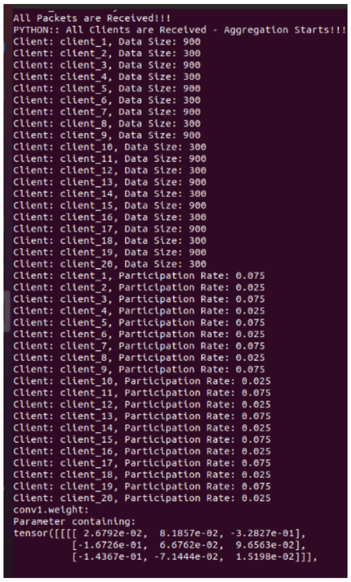


```
Client_18 Accuracy: 0.845857
PYTHON:: Received Client is: 19
Client_19 Accuracy: 0.908714
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.868857
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
Client: client_1, Data Size: 900
Client: client_2, Data Size: 300
Client: client_3, Data Size: 900
Client: client_4, Data Size: 300
Client: client_5, Data Size: 900
Client: client_6, Data Size: 300
Client: client_7, Data Size: 900
Client: client_8, Data Size: 300
Client: client_9, Data Size: 900
Client: client_10, Data Size: 300
Client: client_11, Data Size: 900
Client: client_12, Data Size: 300
Client: client_13, Data Size: 900
Client: client_14, Data Size: 300
Client: client_15, Data Size: 900
Client: client_16, Data Size: 300
Client: client_17, Data Size: 900
Client: client_18, Data Size: 300
Client: client_19, Data Size: 900
Client: client_20, Data Size: 300
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.025
Client: client_3, Participation Rate: 0.075
Client: client_4, Participation Rate: 0.025
Client: client_5, Participation Rate: 0.075
Client: client_6, Participation Rate: 0.025
Client: client_7, Participation Rate: 0.075
Client: client_8, Participation Rate: 0.025
Client: client_9, Participation Rate: 0.075
Client: client_10, Participation Rate: 0.025
Client: client_11, Participation Rate: 0.075
Client: client_12, Participation Rate: 0.025
Client: client_13, Participation Rate: 0.075
Client: client_14, Participation Rate: 0.025
Client: client_15, Participation Rate: 0.075
Client: client_16, Participation Rate: 0.025
Client: client_17, Participation Rate: 0.075
Client: client_18, Participation Rate: 0.025
Client: client_19, Participation Rate: 0.075
Client: client_20, Participation Rate: 0.025
conv1.weight:
Parameter containing:
```

For round 2

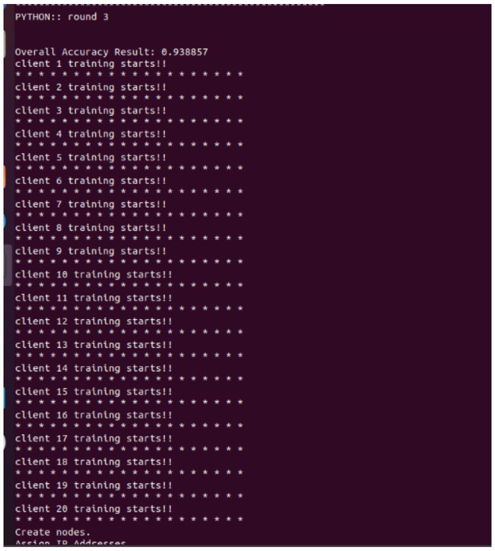


```
PYTHON:: round 2
Overall Accuracy Result: 0.922143
client 1 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 2 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 3 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 4 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 5 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 6 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 7 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 8 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 9 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 10 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 11 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 16 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 17 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 18 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 19 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 20 training starts!!
* * * * * * * * * * * * * * * * * * * *
Create nodes.
Assign IP Addresses.
Create sockets.
Run Simulation.
```

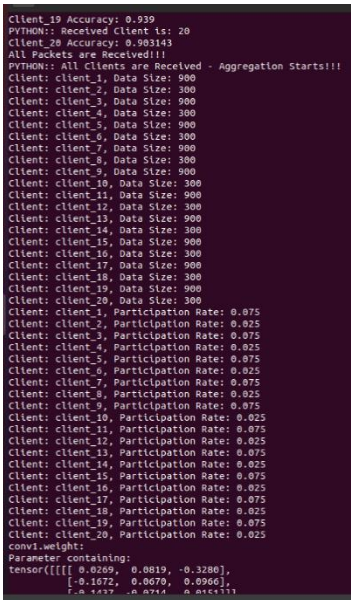


```
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
Client: client_1, Data Size: 900
Client: client_2, Data Size: 300
Client: client_3, Data Size: 900
Client: client_4, Data Size: 300
Client: client_5, Data Size: 900
Client: client_6, Data Size: 300
Client: client_7, Data Size: 900
Client: client_8, Data Size: 300
Client: client_9, Data Size: 900
Client: client_10, Data Size: 300
Client: client_11, Data Size: 900
Client: client_12, Data Size: 300
Client: client_13, Data Size: 900
Client: client_14, Data Size: 300
Client: client_15, Data Size: 900
Client: client_16, Data Size: 300
Client: client_17, Data Size: 900
Client: client_18, Data Size: 300
Client: client_19, Data Size: 900
Client: client_20, Data Size: 300
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.025
Client: client_3, Participation Rate: 0.075
Client: client_4, Participation Rate: 0.025
Client: client_5, Participation Rate: 0.075
Client: client_6, Participation Rate: 0.025
Client: client_7, Participation Rate: 0.075
Client: client_8, Participation Rate: 0.025
Client: client_9, Participation Rate: 0.075
Client: client_10, Participation Rate: 0.025
Client: client_11, Participation Rate: 0.075
Client: client_12, Participation Rate: 0.025
Client: client_13, Participation Rate: 0.075
Client: client_14, Participation Rate: 0.025
Client: client_15, Participation Rate: 0.075
Client: client_16, Participation Rate: 0.025
Client: client_17, Participation Rate: 0.075
Client: client_18, Participation Rate: 0.025
Client: client_19, Participation Rate: 0.075
Client: client_20, Participation Rate: 0.025
conv1.weight:
Parameter containing:
tensor([[[[ 2.6792e-02,  8.1857e-02, -3.2827e-01],
          [-1.6726e-01,  6.6762e-02,  9.6563e-02],
          [-1.4367e-01, -7.1444e-02,  1.5198e-02]]],
```

For round 3

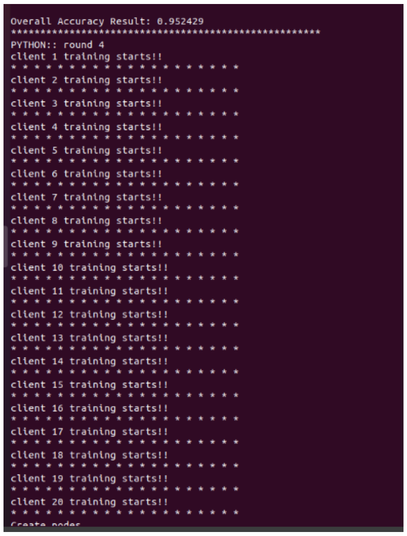


```
PYTHON:: round 3


Overall Accuracy Result: 0.938857
client 1 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 2 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 3 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 4 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 5 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 6 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 7 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 8 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 9 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 10 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 11 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 16 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 17 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 18 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 19 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 20 training starts!!
* * * * * * * * * * * * * * * * * * * *
Create nodes.
Assign IP Addresses
```

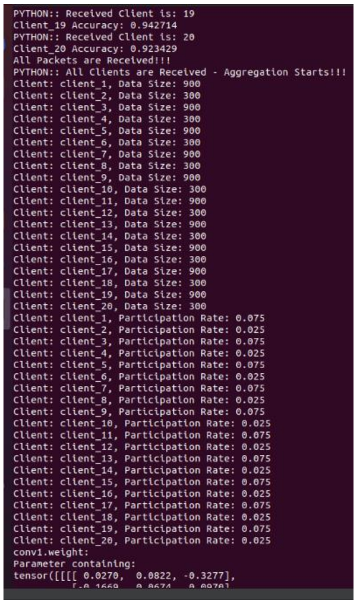


```
Client_19 Accuracy: 0.939
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.903143
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
Client: client_1, Data Size: 900
Client: client_2, Data Size: 300
Client: client_3, Data Size: 900
Client: client_4, Data Size: 300
Client: client_5, Data Size: 900
Client: client_6, Data Size: 300
Client: client_7, Data Size: 900
Client: client_8, Data Size: 300
Client: client_9, Data Size: 900
Client: client_10, Data Size: 300
Client: client_11, Data Size: 900
Client: client_12, Data Size: 300
Client: client_13, Data Size: 900
Client: client_14, Data Size: 300
Client: client_15, Data Size: 900
Client: client_16, Data Size: 300
Client: client_17, Data Size: 900
Client: client_18, Data Size: 300
Client: client_19, Data Size: 900
Client: client_20, Data Size: 300
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.025
Client: client_3, Participation Rate: 0.075
Client: client_4, Participation Rate: 0.025
Client: client_5, Participation Rate: 0.075
Client: client_6, Participation Rate: 0.025
Client: client_7, Participation Rate: 0.075
Client: client_8, Participation Rate: 0.025
Client: client_9, Participation Rate: 0.075
Client: client_10, Participation Rate: 0.025
Client: client_11, Participation Rate: 0.075
Client: client_12, Participation Rate: 0.025
Client: client_13, Participation Rate: 0.075
Client: client_14, Participation Rate: 0.025
Client: client_15, Participation Rate: 0.075
Client: client_16, Participation Rate: 0.025
Client: client_17, Participation Rate: 0.075
Client: client_18, Participation Rate: 0.025
Client: client_19, Participation Rate: 0.075
Client: client_20, Participation Rate: 0.025
conv1.weight:
Parameter containing:
tensor([[[[ 0.0269,  0.0819, -0.3280],
          [-0.1672,  0.0670,  0.0966],
          [-0.1437, -0.0714,  0.0151]]],
```

For round 4

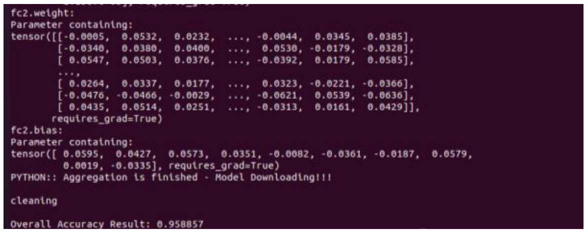


```
Overall Accuracy Result: 0.952429
*****************************************************
PYTHON:: round 4
client 1 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 2 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 3 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 4 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 5 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 6 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 7 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 8 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 9 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 10 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 11 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 16 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 17 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 18 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 19 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 20 training starts!!
* * * * * * * * * * * * * * * * * * * *
Create nodes
```



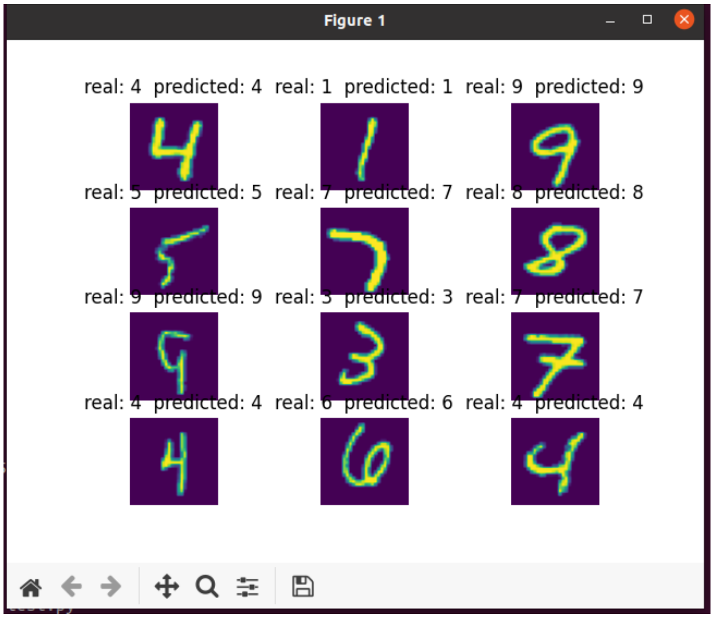
```
PYTHON:: Received Client is: 19
Client_19 Accuracy: 0.942714
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.923429
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
Client: client_1, Data Size: 900
Client: client_2, Data Size: 300
Client: client_3, Data Size: 900
Client: client_4, Data Size: 300
Client: client_5, Data Size: 900
Client: client_6, Data Size: 300
Client: client_7, Data Size: 900
Client: client_8, Data Size: 300
Client: client_9, Data Size: 900
Client: client_10, Data Size: 300
Client: client_11, Data Size: 900
Client: client_12, Data Size: 300
Client: client_13, Data Size: 900
Client: client_14, Data Size: 300
Client: client_15, Data Size: 900
Client: client_16, Data Size: 300
Client: client_17, Data Size: 900
Client: client_18, Data Size: 300
Client: client_19, Data Size: 900
Client: client_20, Data Size: 300
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.025
Client: client_3, Participation Rate: 0.075
Client: client_4, Participation Rate: 0.025
Client: client_5, Participation Rate: 0.075
Client: client_6, Participation Rate: 0.025
Client: client_7, Participation Rate: 0.075
Client: client_8, Participation Rate: 0.025
Client: client_9, Participation Rate: 0.075
Client: client_10, Participation Rate: 0.025
Client: client_11, Participation Rate: 0.075
Client: client_12, Participation Rate: 0.025
Client: client_13, Participation Rate: 0.075
Client: client_14, Participation Rate: 0.025
Client: client_15, Participation Rate: 0.075
Client: client_16, Participation Rate: 0.025
Client: client_17, Participation Rate: 0.075
Client: client_18, Participation Rate: 0.025
Client: client_19, Participation Rate: 0.075
Client: client_20, Participation Rate: 0.025
conv1.weight:
Parameter containing:
tensor([[[[ 0.0270,  0.0822, -0.3277],
          [-0.1669,  0.0674,  0.0970]
```

- Making weighted aggregation with these participation rates



```
fc2.weight:
Parameter containing:
tensor([[-0.0005,  0.0532,  0.0232,  ..., -0.0044,  0.0345,  0.0385],
        [-0.0340,  0.0380,  0.0400,  ...,  0.0530, -0.0179, -0.0328],
        [ 0.0547,  0.0503,  0.0376,  ..., -0.0392,  0.0179,  0.0585],
        ...,
        [ 0.0264,  0.0337,  0.0177,  ...,  0.0323, -0.0221, -0.0366],
        [-0.0476, -0.0466, -0.0029,  ..., -0.0621,  0.0539, -0.0636],
        [ 0.0435,  0.0514,  0.0251,  ..., -0.0313,  0.0161,  0.0429]],
       requires_grad=True)
fc2.bias:
Parameter containing:
tensor([ 0.0595,  0.0427,  0.0573,  0.0351, -0.0082, -0.0361, -0.0187,  0.0579,
         0.0019, -0.0335], requires_grad=True)
PYTHON:: Aggregation is finished - Model Downloading!!!

cleaning

Overall Accuracy Result: 0.958857
```

- Creating CNN architecture



1-4- List of files

custom_app_fl.cc

global_model.pickle

helper.py

run.py

test.py

Assignment4.pdf