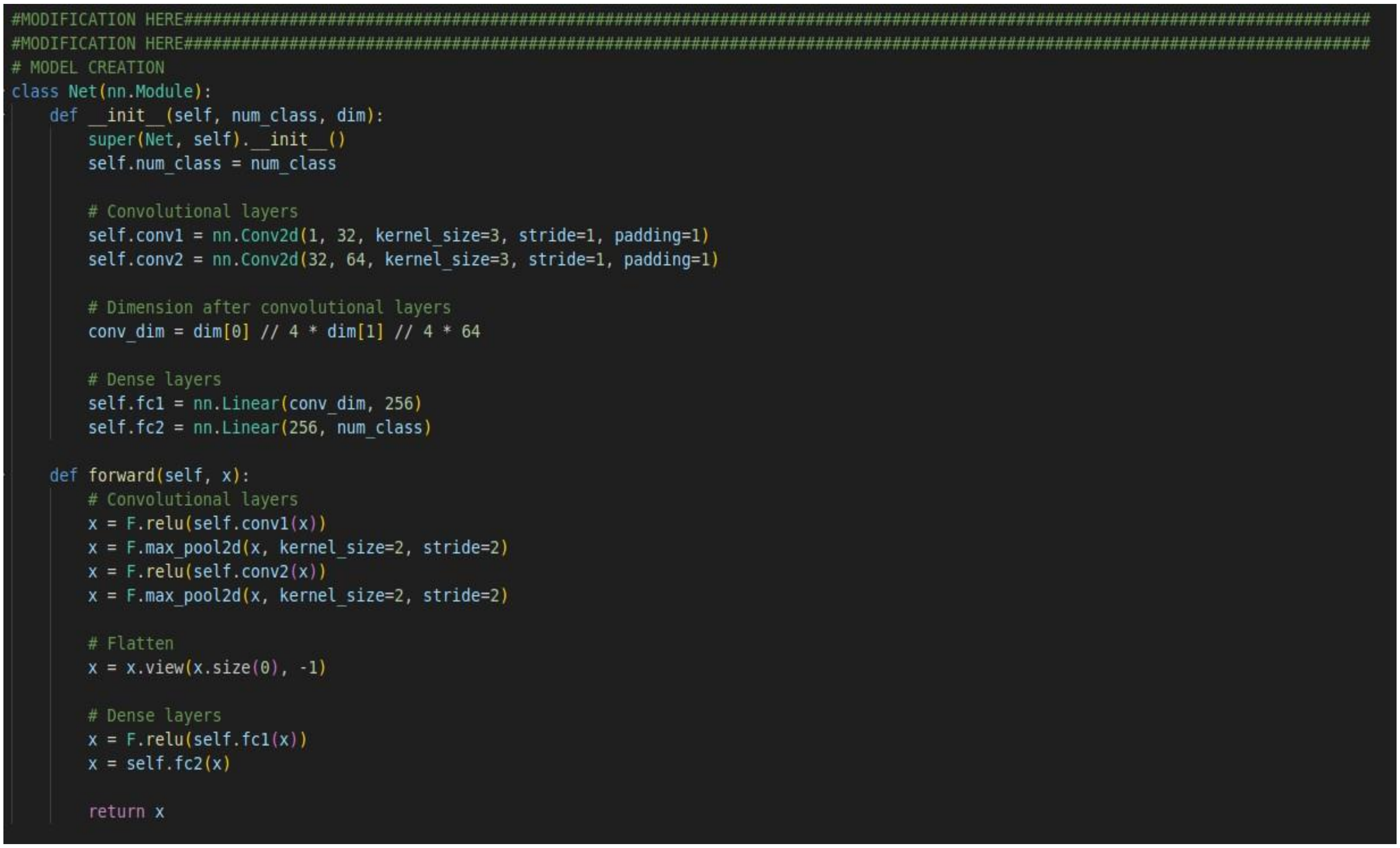
Task 1

1-1- Modifications in code

helper.py File modification:

1- *Creating Advance deep learning architecture CNN with 2 Convolutional and 2 dense layers*

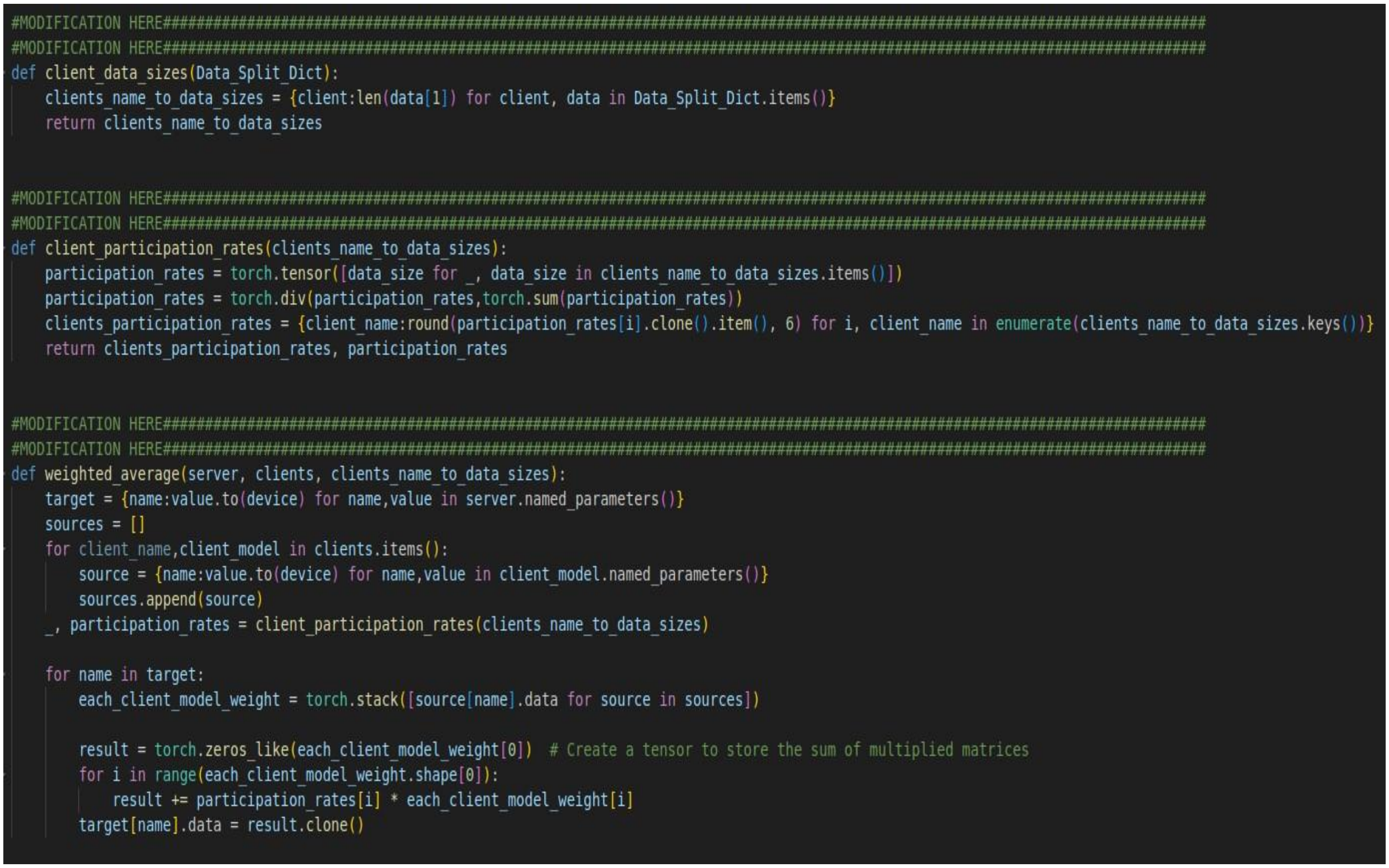


```
#MODIFICATION HERE#############################################################################################################################
#MODIFICATION HERE#############################################################################################################################
# MODEL CREATION
class Net(nn.Module):
    def __init__(self, num_class, dim):
        super(Net, self).__init__()
        self.num_class = num_class

        # Convolutional layers
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)

        # Dimension after convolutional layers
        conv_dim = dim[0] // 4 * dim[1] // 4 * 64

        # Dense layers
        self.fc1 = nn.Linear(conv_dim, 256)
        self.fc2 = nn.Linear(256, num_class)

    def forward(self, x):
        # Convolutional layers
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, kernel_size=2, stride=2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, kernel_size=2, stride=2)

        # Flatten
        x = x.view(x.size(0), -1)

        # Dense layers
        x = F.relu(self.fc1(x))
        x = self.fc2(x)

        return x
```

2- *client_data_sizes(): takes the dictionary of client names & corresponding data that would be passed to this client & returns dictionary of client names & corresponding size of this data.*

client_participation_rates(): takes a dictionary of client names & corresponding size of the data that would be passed to it & returns dictionary of client names & participation rate evaluated from corresponding data size passed to the client Also returns the same participation rates of each client but without client names in a torch tensor data structure that would be used in calculating weighted average as the participation rates dictionary would be used in presentation & printing resulted participation rates from clients data sizes.

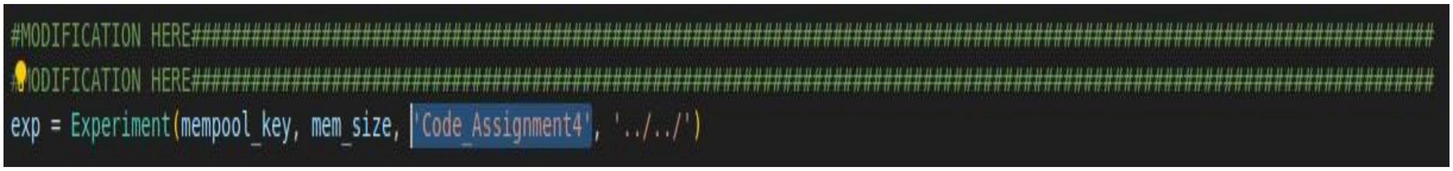
Weighted_average(): replaces the federated average with weighted average to Apply weighted Averaging aggregation methods regarding model parameter & their client's data size done by participation Rate by Multiply these participation rates with clients' weights. After that, aggregating of all adjusted weight using variable result that is tensor of size of model parameters that accumulates the multiplication result for each clients' weights & participation rate.



```python
#MODIFICATION HERE#############################################################################################################################
#MODIFICATION HERE#############################################################################################################################
def client_data_sizes(Data_Split_Dict):
    clients_name_to_data_sizes = {client:len(data[1]) for client, data in Data_Split_Dict.items()}
    return clients_name_to_data_sizes


#MODIFICATION HERE#############################################################################################################################
#MODIFICATION HERE#############################################################################################################################
def client_participation_rates(clients_name_to_data_sizes):
    participation_rates = torch.tensor([data_size for _, data_size in clients_name_to_data_sizes.items()])
    participation_rates = torch.div(participation_rates,torch.sum(participation_rates))
    clients_participation_rates = {client_name:round(participation_rates[i].clone().item(), 6) for i, client_name in enumerate(clients_name_to_data_sizes.keys())}
    return clients_participation_rates, participation_rates


#MODIFICATION HERE#############################################################################################################################
#MODIFICATION HERE#############################################################################################################################
def weighted_average(server, clients, clients_name_to_data_sizes):
    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)
    _, participation_rates = client_participation_rates(clients_name_to_data_sizes)

    for name in target:
        each_client_model_weight = torch.stack([source[name].data for source in sources])

        result = torch.zeros_like(each_client_model_weight[0])  # Create a tensor to store the sum of multiplied matrices
        for i in range(each_client_model_weight.shape[0]):
            result += participation_rates[i] * each_client_model_weight[i]
        target[name].data = result.clone()
```

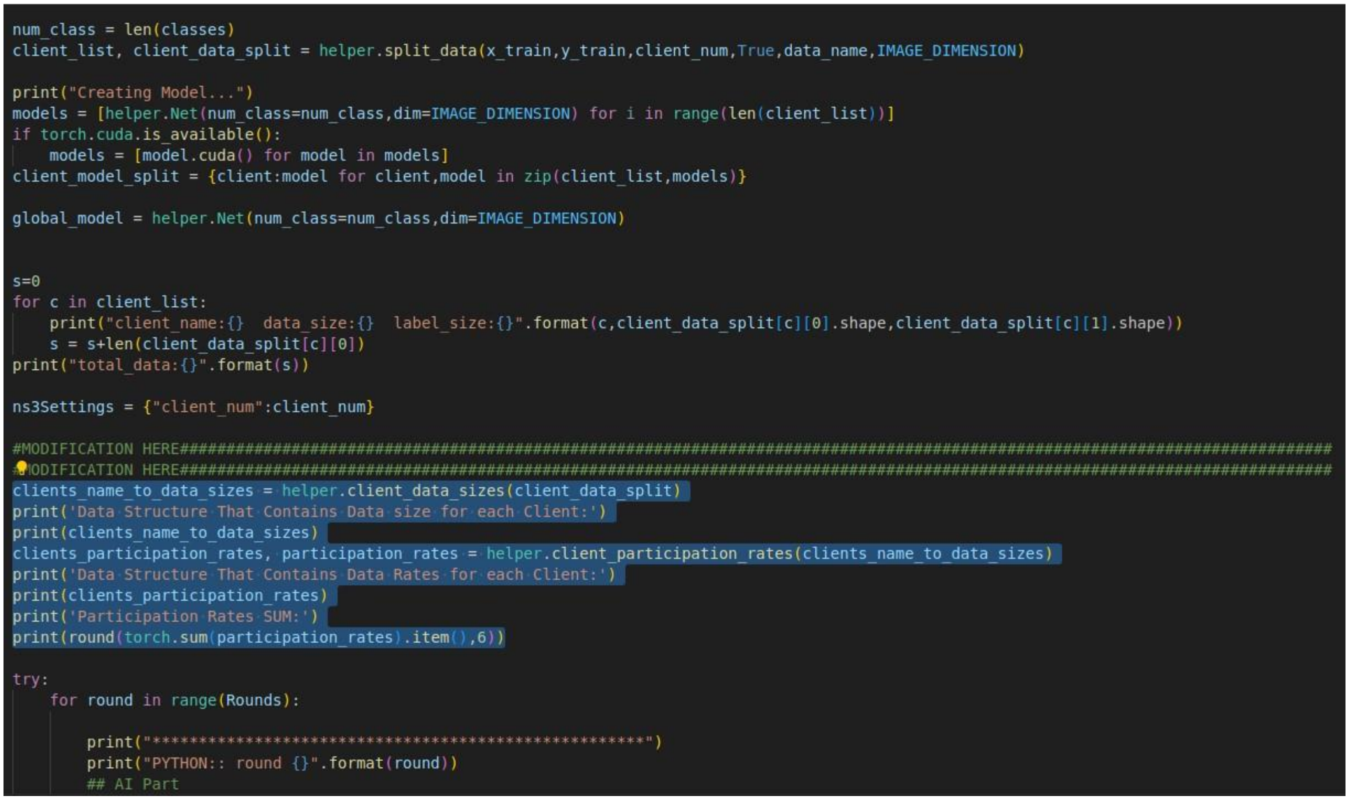
run.py File modification:

1- Change path of experiment to the folder contains project files: Code_Assignment4



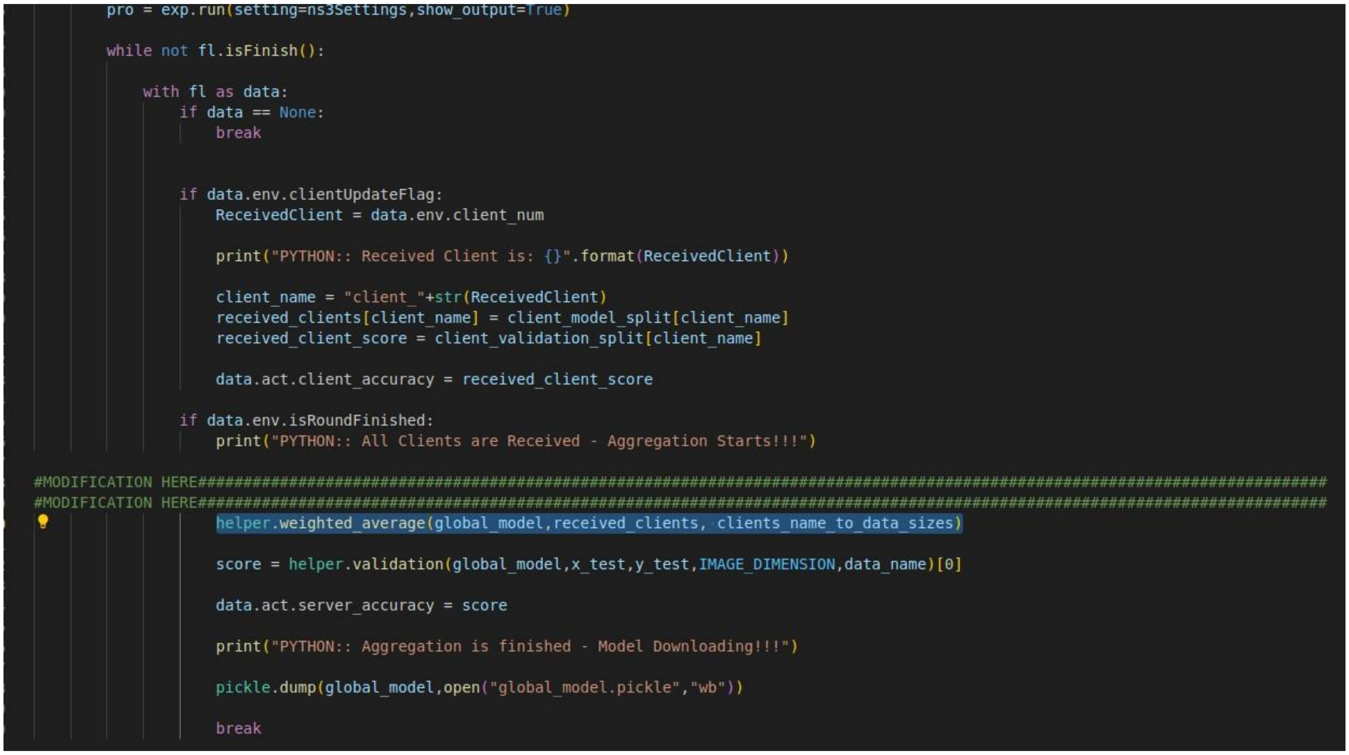
```
#MODIFICATION HERE######################################################################
#MODIFICATION HERE######################################################################
exp = Experiment(mempool_key, mem_size, 'Code_Assignment4', '../../')
```

2- Print Clients' Names & it's proper corresponding Data Size & Print Clients' Names & it's proper corresponding Participation Rate & assessment code that make sure all clients participation rate summation is 1 by Printing the summation.



```
num_class = len(classes)
client_list, client_data_split = helper.split_data(x_train,y_train,client_num,True,data_name,IMAGE_DIMENSION)

print("Creating Model...")
models = [helper.Net(num_class=num_class,dim=IMAGE_DIMENSION) for i in range(len(client_list))]
if torch.cuda.is_available():
    models = [model.cuda() for model in models]
client_model_split = {client:model for client,model in zip(client_list,models)}

global_model = helper.Net(num_class=num_class,dim=IMAGE_DIMENSION)


s=0
for c in client_list:
    print("client_name:{}  data_size:{}  label_size:{}".format(c,client_data_split[c][0].shape,client_data_split[c][1].shape))
    s = s+len(client_data_split[c][0])
print("total_data:{}".format(s))

ns3Settings = {"client_num":client_num}

#MODIFICATION HERE##################################################################################################
#MODIFICATION HERE##################################################################################################
clients_name_to_data_sizes = helper.client_data_sizes(client_data_split)
print('Data Structure That Contains Data size for each Client:')
print(clients_name_to_data_sizes)
clients_participation_rates, participation_rates = helper.client_participation_rates(clients_name_to_data_sizes)
print('Data Structure That Contains Data Rates for each Client:')
print(clients_participation_rates)
print('Participation Rates SUM:')
print(round(torch.sum(participation_rates).item(),6))

try:
    for round in range(Rounds):

        print("*******************************************************")
        print("PYTHON:: round {}".format(round))
        ## AI Part
```

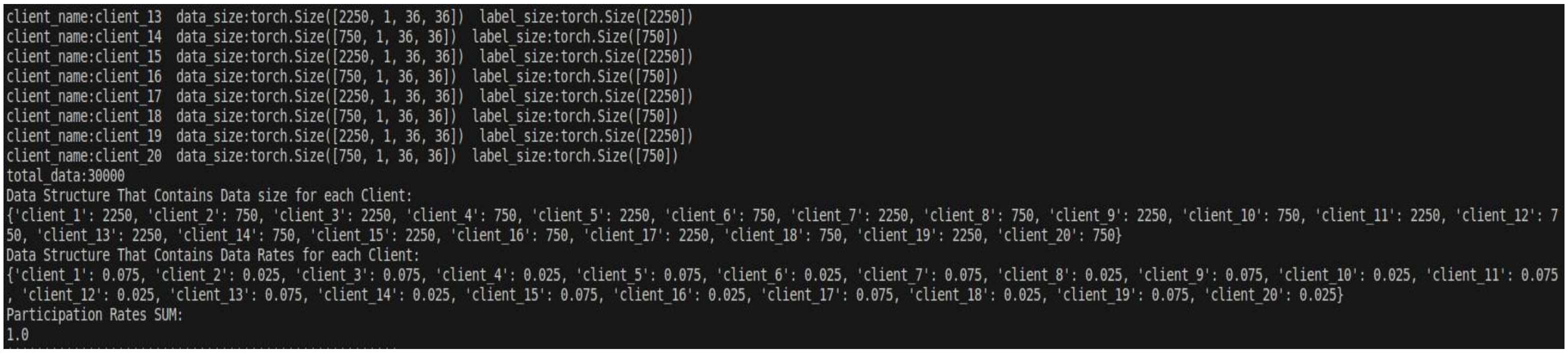
3- *Replace the usage of federated_average() function with weighted_average() function that consider participation rate of clients in weights updates.*



```
        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!!!")

#MODIFICATION HERE############################################################################################################################
#MODIFICATION HERE############################################################################################################################
                    helper.weighted_average(global_model,received_clients, clients_name_to_data_sizes)

                    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
```

1-2- Results

Clients’ Data Sizes & Participation Rates Data Structures Results & Participation Rates Assessment result :



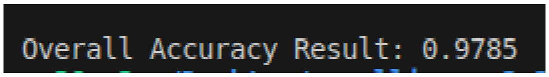
```
client_name:client_13  data_size:torch.Size([2250, 1, 36, 36])  label_size:torch.Size([2250])
client_name:client_14  data_size:torch.Size([750, 1, 36, 36])  label_size:torch.Size([750])
client_name:client_15  data_size:torch.Size([2250, 1, 36, 36])  label_size:torch.Size([2250])
client_name:client_16  data_size:torch.Size([750, 1, 36, 36])  label_size:torch.Size([750])
client_name:client_17  data_size:torch.Size([2250, 1, 36, 36])  label_size:torch.Size([2250])
client_name:client_18  data_size:torch.Size([750, 1, 36, 36])  label_size:torch.Size([750])
client_name:client_19  data_size:torch.Size([2250, 1, 36, 36])  label_size:torch.Size([2250])
client_name:client_20  data_size:torch.Size([750, 1, 36, 36])  label_size:torch.Size([750])
total_data:30000
Data Structure That Contains Data size for each Client:
{'client_1': 2250, 'client_2': 750, 'client_3': 2250, 'client_4': 750, 'client_5': 2250, 'client_6': 750, 'client_7': 2250, 'client_8': 750, 'client_9': 2250, 'client_10': 750, 'client_11': 2250, 'client_12': 7
50, 'client_13': 2250, 'client_14': 750, 'client_15': 2250, 'client_16': 750, 'client_17': 2250, 'client_18': 750, 'client_19': 2250, 'client_20': 750}
Data Structure That Contains Data Rates for each Client:
{'client_1': 0.075, 'client_2': 0.025, 'client_3': 0.075, 'client_4': 0.025, 'client_5': 0.075, 'client_6': 0.025, 'client_7': 0.075, 'client_8': 0.025, 'client_9': 0.075, 'client_10': 0.025, 'client_11': 0.075
, 'client_12': 0.025, 'client_13': 0.075, 'client_14': 0.025, 'client_15': 0.075, 'client_16': 0.025, 'client_17': 0.075, 'client_18': 0.025, 'client_19': 0.075, 'client_20': 0.025}
Participation Rates SUM:
1.0
```

Sample From Rounds Results :

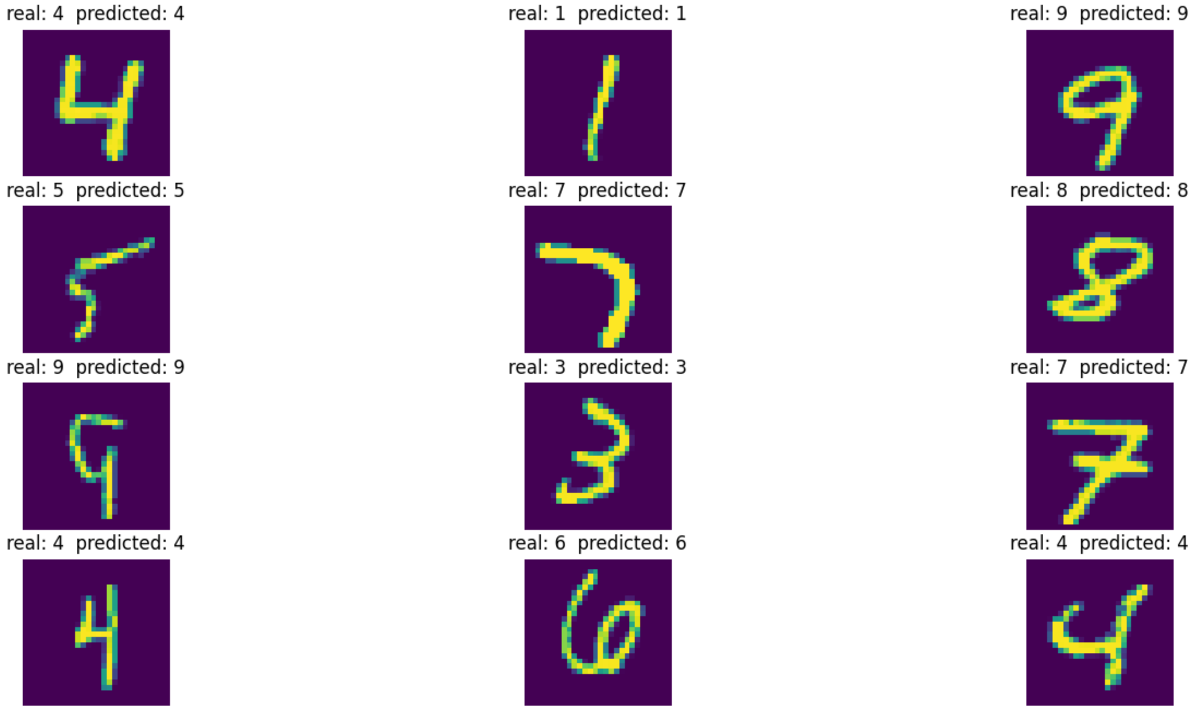
```
*****************************************************
PYTHON:: round 9

Overall Accuracy Result: 0.9785
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!!
```

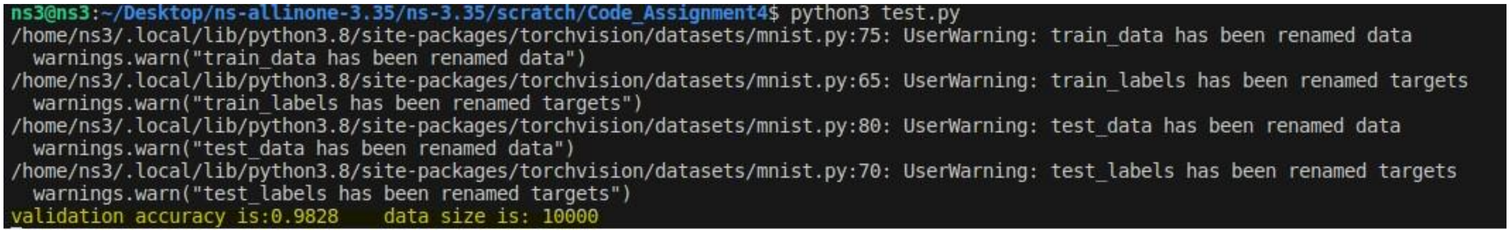
```
* * * * * * * * * * * * * * * * * * * *
client 12 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 13 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 14 training starts!!
* * * * * * * * * * * * * * * * * * * *
client 15 training starts!!
* * * * * * * * * * * * * * * * * * * *
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.
Server: 10.2.1.1
Client: 10.1.0.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.1.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.2.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.3.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.4.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.5.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.6.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.7.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.8.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.9.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.10.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.11.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.12.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.13.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.14.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.15.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.16.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.17.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.18.1 m_peer:03-07-0a:02:01:01:7d:11:00
Client: 10.1.19.1 m_peer:03-07-0a:02:01:01:7d:11:00
All Packets Are Sent by Client_1!!!
All Packets Are Sent by Client_2!!!
All Packets Are Sent by Client_3!!!
All Packets Are Sent by Client_4!!!
All Packets Are Sent by Client_5!!!
All Packets Are Sent by Client_6!!!
All Packets Are Sent by Client_7!!!
All Packets Are Sent by Client_8!!!
All Packets Are Sent by Client_9!!!
All Packets Are Sent by Client_10!!!
All Packets Are Sent by Client_11!!!
All Packets Are Sent by Client_12!!!
All Packets Are Sent by Client_13!!!
All Packets Are Sent by Client_14!!!
All Packets Are Sent by Client_15!!!
All Packets Are Sent by Client_16!!!
All Packets Are Sent by Client_17!!!
All Packets Are Sent by Client_18!!!
All Packets Are Sent by Client_19!!!
All Packets Are Sent by Client_20!!!
PYTHON:: Received Client is: 1
Client_1 Accuracy: 0.969667
PYTHON:: Received Client is: 2
Client_2 Accuracy: 0.966833
PYTHON:: Received Client is: 3
Client_3 Accuracy: 0.973
PYTHON:: Received Client is: 4
Client_4 Accuracy: 0.968
PYTHON:: Received Client is: 5
Client_5 Accuracy: 0.972333
PYTHON:: Received Client is: 6
Client_6 Accuracy: 0.97
PYTHON:: Received Client is: 7
Client_7 Accuracy: 0.97
PYTHON:: Received Client is: 8
Client_8 Accuracy: 0.966333
PYTHON:: Received Client is: 9
Client_9 Accuracy: 0.968333
PYTHON:: Received Client is: 10
Client_10 Accuracy: 0.968333
PYTHON:: Received Client is: 11
Client_11 Accuracy: 0.971333
PYTHON:: Received Client is: 12
Client_12 Accuracy: 0.968
PYTHON:: Received Client is: 13
Client_13 Accuracy: 0.972
PYTHON:: Received Client is: 14
Client_14 Accuracy: 0.966833
PYTHON:: Received Client is: 15
Client_15 Accuracy: 0.972833
PYTHON:: Received Client is: 16
Client_16 Accuracy: 0.968
```

```
PYTHON:: Received Client is: 17
Client_17 Accuracy: 0.970167
PYTHON:: Received Client is: 18
Client_18 Accuracy: 0.9675
PYTHON:: Received Client is: 19
Client_19 Accuracy: 0.967333
PYTHON:: Received Client is: 20
Client_20 Accuracy: 0.961167
All Packets are Received!!!
PYTHON:: All Clients are Received - Aggregation Starts!!!
PYTHON:: Aggregation is finished - Model Downloading!!!
cleaning
```



```
Overall Accuracy Result: 0.9785
```





```
ns3@ns3:~/Desktop/ns-allinone-3.35/ns-3.35/scratch/Code_Assignment4$ python3 test.py
/home/ns3/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:75: UserWarning: train_data has been renamed data
  warnings.warn("train_data has been renamed data")
/home/ns3/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:65: UserWarning: train_labels has been renamed targets
  warnings.warn("train_labels has been renamed targets")
/home/ns3/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:80: UserWarning: test_data has been renamed data
  warnings.warn("test_data has been renamed data")
/home/ns3/.local/lib/python3.8/site-packages/torchvision/datasets/mnist.py:70: UserWarning: test_labels has been renamed targets
  warnings.warn("test_labels has been renamed targets")
validation accuracy is:0.9828    data size is: 10000
```

1-3- List of files

1- helper.py

2- test.py

3- run.py

4- global_model_pickle

5- custom_app_fl_cc