

# 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.

```
253
254 def federated_averaging(server, clients, client_data_split):
255     client_data_sizes = {client_name: len(data_split[0]) for client_name, data_split in client_data_split.items()}
256
257     for client_name, data_split in client_data_split.items():
258         data_size = len(data_split[0])
259         print(f"Client: {client_name}, Data Size: {data_size}")
260
261     # Calculate and print each client's participation rate
262     total_data_size = sum(len(data_split[0]) for data_split in client_data_split.values())
263     for client_name, data_split in client_data_split.items():
264         participation_rate = len(data_split[0]) / total_data_size
265         print(f"Client: {client_name}, Participation Rate: {participation_rate}")
266
267     target = {name: value.to(device) for name, value in server.named_parameters()}
268     sources = []
269     for client_name, client_model in clients.items():
270         source = {name: value.to(device) for name, value in client_model.named_parameters()}
271         sources.append(source)
272
273     for name in target:
274         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)])
275         target[name].data = torch.sum(weighted_params, dim=0).clone()
276
277     # Print the weighted aggregation with the participation rates
278     for name, value in target.items():
279         print(f"{name}: {value}")
```

Python Tab Width: 8 Ln 254, Col 61

```
"""run.py"""
"""helper.py"""
106 client_validation_split[client_name] = helper.validation(client_model_split[client_name], x_test, y_test, IMAGE_DIMENSION, data_name)[0]
107
108 ## Simulation Part
109 received_clients = {}
110
111 exp.reset()
112 pro = exp.run(setting=ns3Settings, show_output=True)
113 while not fl.isFinish():
114
115     with fl as data:
116         if data == None:
117             break
118
119
120         if data.env.clientUpdateFlag:
121             ReceivedClient = data.env.client_num
122
123             print("PYTHON: Received Client is: {}".format(ReceivedClient))
124
125             client_name = "client_" + str(ReceivedClient)
126             received_clients[client_name] = client_model_split[client_name]
127             received_client_score = client_validation_split[client_name]
128
129             data.act.client_accuracy = received_client_score
130
131         if data.env.isRoundFinished:
132             print("PYTHON: All Clients are Received - Aggregation Starts!!!")
133
134             helper.federated_averaging/global_model, received_clients, client_data_split[
135
136             score = helper.validation(global_model, x_test, y_test, IMAGE_DIMENSION, data_name)[0]
137
138             data.act.server_accuracy = score
139
140             print("PYTHON: Aggregation is Finished - Model Downloading!!!")
141             pickle.dump(global_model, open("global_model.pickle", "wb"))
142
143             break
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

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

```
164 # MODEL CREATION
165 class Net(nn.Module):
166     def __init__(self, num_class, dtn):
167         super(Net, self).__init__()
168         #self.flatten_size = dtn[0] + dtn[1]
169         self.input_layer = nn.Flatten()
170         self.conv1 = nn.Conv2d(1, 64, 3, 1, 1)
171         self.conv2 = nn.Conv2d(64, 128, 3, 1, 1)
172         self.fc1 = nn.Linear(128 * (dtn[0]//4) * (dtn[1]//4), 256)
173         self.fc2 = nn.Linear(256, num_class)
174     def forward(self, x):
175         x = F.relu(self.conv1(x))
176         x = F.max_pool2d(x, 2)
177         x = F.relu(self.conv2(x))
178         x = F.max_pool2d(x, 2)
179         x = x.view(x.size(0), -1)
180         x = F.relu(self.fc1(x))
181         x = self.fc2(x)
182     return x
183
```

### 3. Define each Client's data size

```
run.py                                     *helper.py
36
37 exp = Experiment(mempool_key, mem_size, 'Tutorial_8', './././')
38
39 num_pocket = 8
40 fl = NS3ATRL(memblock_key, Env, Act)
41 flag = True
42
43 Rounds = 5
44 local_epochs = 20
45 batch_size = 128
46
47 accuracy = 0
48
49 # SET SEED
50 seedNum = 1
51 torch.manual_seed(seedNum)
52 np.random.seed(seedNum)
53
54 DATA_SIZE = 12000
55 VALIDATION_SPLIT = 0.7
56 DATA_ID_DICT = {'1': 'mnist'}
57 IMAGE_DIMENSION = [36, 36]
58
59
```



## 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 `synchronize_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 `synchronize_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 19
Client 19 Accuracy: 0.876025
PYTHON: Received Client is: 20
Client 20 Accuracy: 0.885125
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([[[[-0.0653, -0.2222, -0.3088],
          [-0.2201,  0.0989, -0.0217]],

```

## 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.
PYTHON: round 1

```

```

client_18 Accuracy: 0.845857
PYTHON: Received client list is: 19
Client_19 Accuracy: 0.868714
PYTHON: Received client list is: 20
client_20 Accuracy: 0.908074
All Packets are Received!!!
PYTHON: All client data are received - Aggregation Starts!!!!
Client: client_1, Data Size: 300
Client: client_2, Data Size: 300
Client: client_3, Data Size: 300
Client: client_4, Data Size: 300
Client: client_5, Data Size: 300
Client: client_6, Data Size: 300
Client: client_7, Data Size: 300
Client: client_8, Data Size: 300
Client: client_9, Data Size: 300
Client: client_10, Data Size: 300
Client: client_11, Data Size: 300
Client: client_12, Data Size: 300
Client: client_13, Data Size: 300
Client: client_14, Data Size: 300
Client: client_15, Data Size: 300
Client: client_16, Data Size: 300
Client: client_17, Data Size: 300
Client: client_18, Data Size: 300
Client: client_19, Data Size: 300
Client: client_20, Data Size: 300
Client: client_1, Participation Rate: 0.075
Client: client_2, Participation Rate: 0.075
Client: client_3, Participation Rate: 0.025
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: 0.075
Client: client_15, Participation Rate: 0.025
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
convl_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],
           [-1.0726e-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.
Create 10 addresses

```

```

Client_19 Accuracy: 0.939
PYTHON: Received Client 15: 20
Client_20 Accuracy: 0.983143
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.weights
Parameter containing:
tensor([[[[ 0.0269,  0.0819, -0.3280],
           [-0.1072,  0.0670,  0.0960],
           [-0.1037, -0.0070,  0.0111]]]])

```

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!!
*****
*****
fraba order

```

```

PYTHON: Received Client's Is: 19
Client_19 Accuracy: 0.947214
PYTHON: Received client's Is: 26
Client_26 Accuracy: 0.923809
All Packets are Received!!!
PYTHON: All clients are Received - Aggregation Starts!!!!
Client_1, Data Size: 900
Client_2, Data Size: 900
Client_3, Data Size: 900
Client_4, Data Size: 900
Client_5, Data Size: 900
Client_6, Data Size: 900
Client_7, Data Size: 900
Client_8, Data Size: 900
Client_9, Data Size: 900
Client_10, Data Size: 900
Client_11, Data Size: 900
Client_12, Data Size: 900
Client_13, Data Size: 900
Client_14, Data Size: 900
Client_15, Data Size: 900
Client_16, Data Size: 900
Client_17, Data Size: 900
Client_18, Data Size: 900
Client_19, Data Size: 900
Client_20, Data Size: 900
Client_1, Participation Rate: 0.675
Client_2, Participation Rate: 0.675
Client_3, Participation Rate: 0.675
Client_4, Participation Rate: 0.625
Client_5, Participation Rate: 0.675
Client_6, Participation Rate: 0.625
Client_7, Participation Rate: 0.675
Client_8, Participation Rate: 0.625
Client_9, Participation Rate: 0.675
Client_10, Participation Rate: 0.625
Client_11, Participation Rate: 0.675
Client_12, Participation Rate: 0.625
Client_13, Participation Rate: 0.675
Client_14, Participation Rate: 0.625
Client_15, Participation Rate: 0.675
Client_16, Participation Rate: 0.625
Client_17, Participation Rate: 0.675
Client_18, Participation Rate: 0.625
Client_19, Participation Rate: 0.675
Client_20, Participation Rate: 0.625
conv1.weight:
Parameter containing:
tensor([[[[ 0.2776, 0.6622, -0.3777],
           [ 0.4644, 0.4076, 0.4076]]]])

```

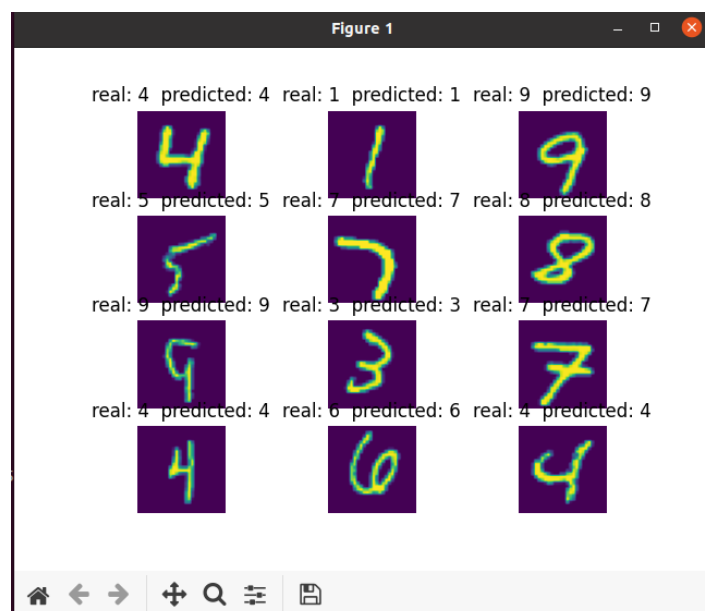
- Making weighted aggregation with these participation rates

```
fc2.weight:
Parameter containing:
tensor([[[[ 0.0005,  0.0532,  0.0232, ...,  0.0044,  0.0345,  0.0305],
          [ 0.0340,  0.0300,  0.0400, ...,  0.0530,  0.0179,  0.0378],
          [ 0.0547,  0.0503,  0.0376, ...,  0.0392,  0.0179,  0.0585],
          ...,
          [ 0.0264,  0.0337,  0.0177, ...,  0.0323,  0.0221,  0.0366],
          [ 0.0470,  0.0466,  0.0029, ...,  0.0021,  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