Dependable AI

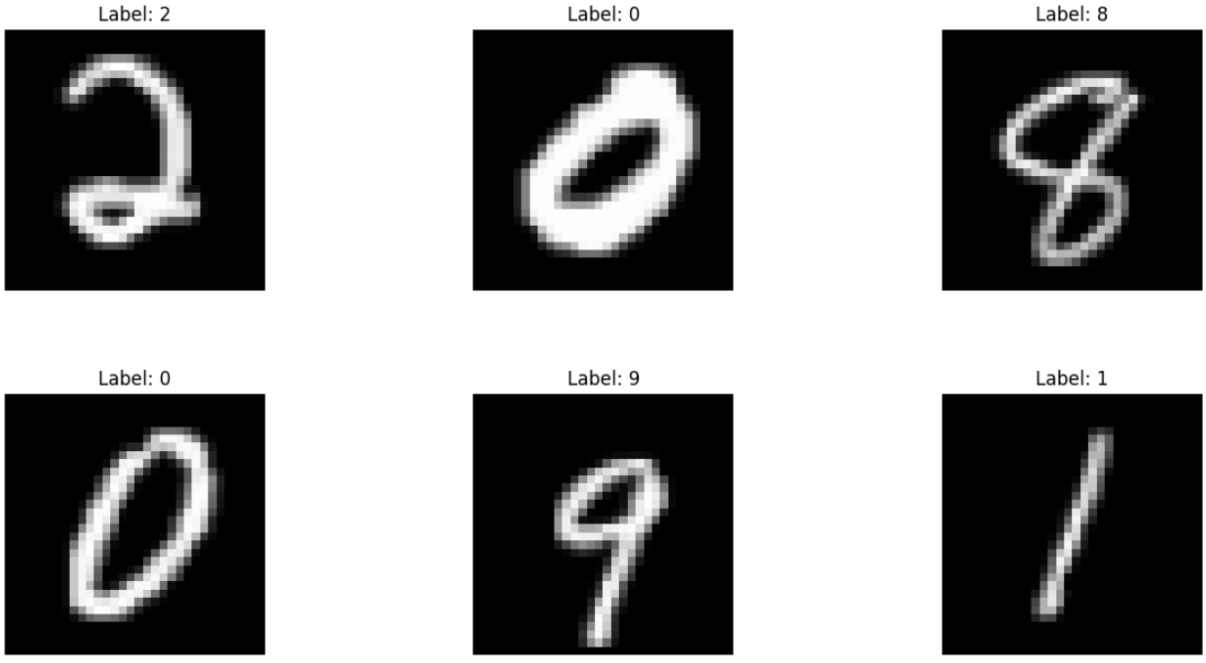
Programming Assignment-3

- Devi Prasad Maharathy (B20AI053)

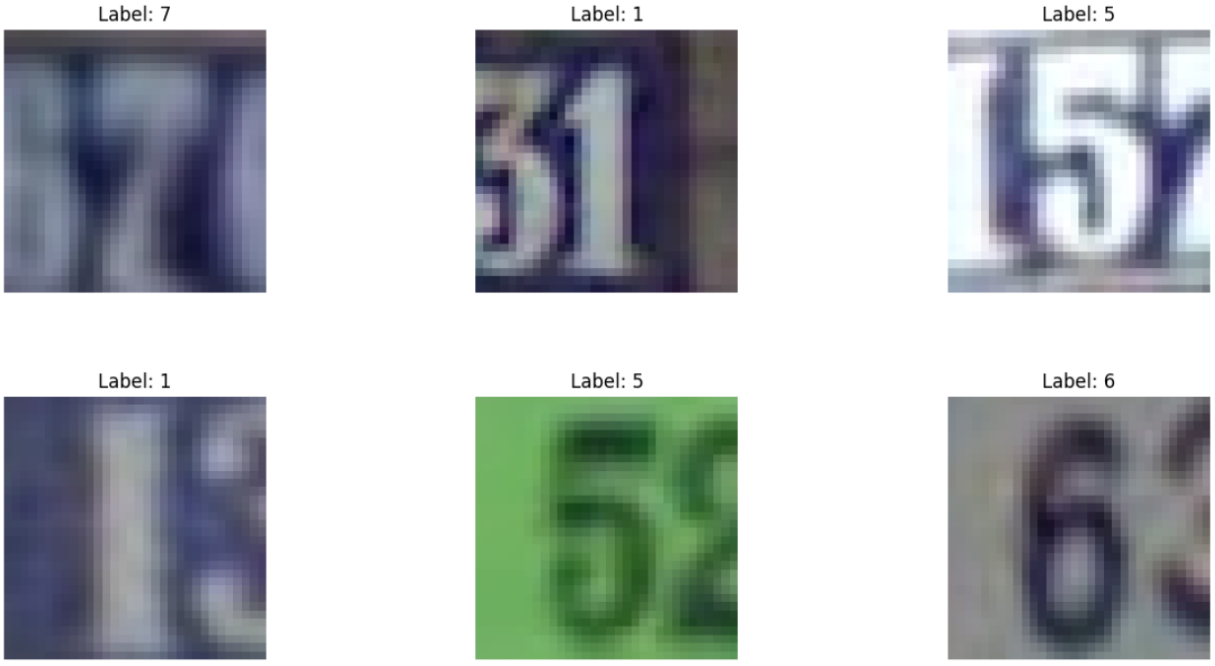
Q1)

A.Perform (0-9) digit classification task using federated setup by performing aggregation at the central server.

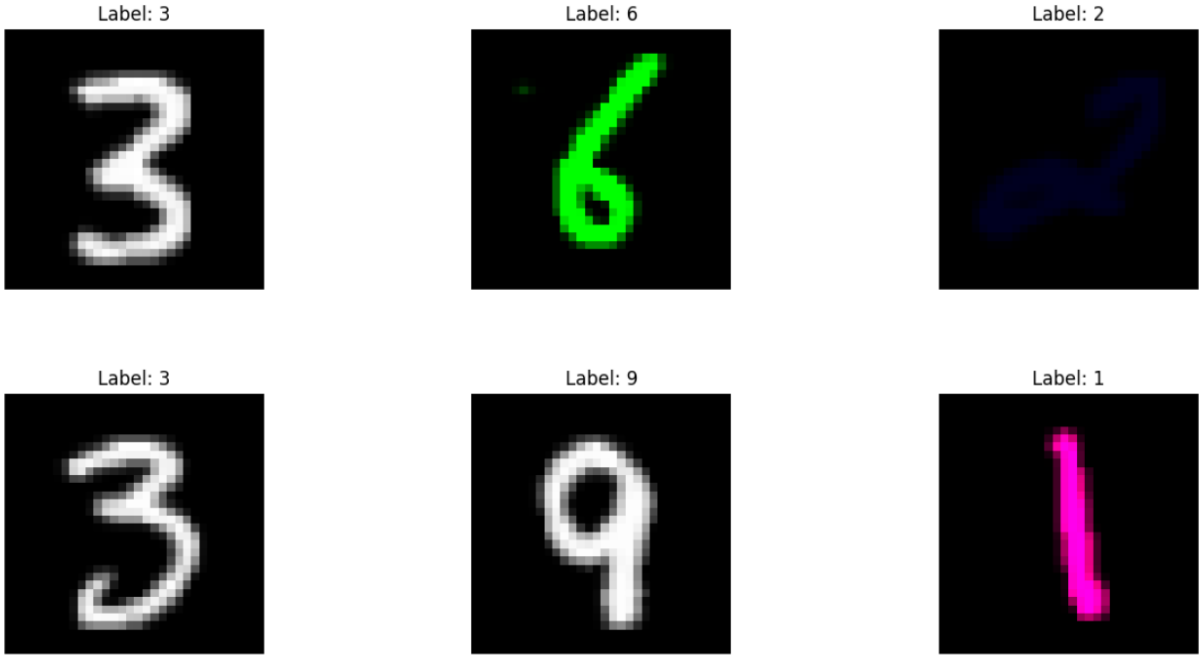
Dataset:MNIST,SVHN and Colored MNIST which was generated from MNIST dataset were used.
MNIST



SVHN



Colored MNIST



I did the necessary preprocessing and converted the 28x28 size images of MNIST and Colored MNIST to 32x32 as the SVHN dataset is of size 32x32 whereas the MNIST and colored MNIST are of size 28x28.I did so to pass them through a common CNN architecture.

CNN architecture.

```
class CNN(nn.Module):
  def __init__(self):
    super(CNN,self).__init__()
    self.conv1=nn.Conv2d(3,32,3,padding=1)
    self.bn1=nn.BatchNorm2d(32)
    self.conv2=nn.Conv2d(32,64,3,padding=1)
    self.bn2=nn.BatchNorm2d(64)
    self.conv3=nn.Conv2d(64,128,3,padding=1)
    self.maxpool=nn.MaxPool2d(kernel_size=2,stride=2)
    self.bn3=nn.BatchNorm2d(128)
    self.act=nn.LeakyReLU()
    self.fc1=nn.Linear(128*16*16,256)
    self.fc2=nn.Linear(256,128)
    self.fc3=nn.Linear(128,10)

  def forward(self,x):
    x=self.conv1(x)
    x=self.bn1(x)
    x=self.act(x)
    x=self.conv2(x)
    x=self.bn2(x)
    x=self.act(x)
    x=self.conv3(x)
    x=self.bn3(x)
    x=self.maxpool(x)
    x=self.act(x)
    x=x.reshape(x.size(0),-1)
    # x = torch.flatten(x, 1)
    x=self.fc1(x)
    x=self.fc2(x)
    x=self.fc3(x)
    return x
```

```python
def update_client(client_model,optimizer,train_loader,num_epochs=5):
  client_model.train()
  client_loss=[]
  for e in range(num_epochs):
    running_loss=0
    for batch_idx,(data,target) in enumerate(train_loader):
      data,target=data.to(device),target.to(device)
      client_model.zero_grad()
      output=client_model(data)
      loss=criterion(output,target)
      running_loss+=loss.item()
      loss.backward()
      optimizer.step()
    client_loss.append(running_loss)
  return loss.item(),client_loss
```

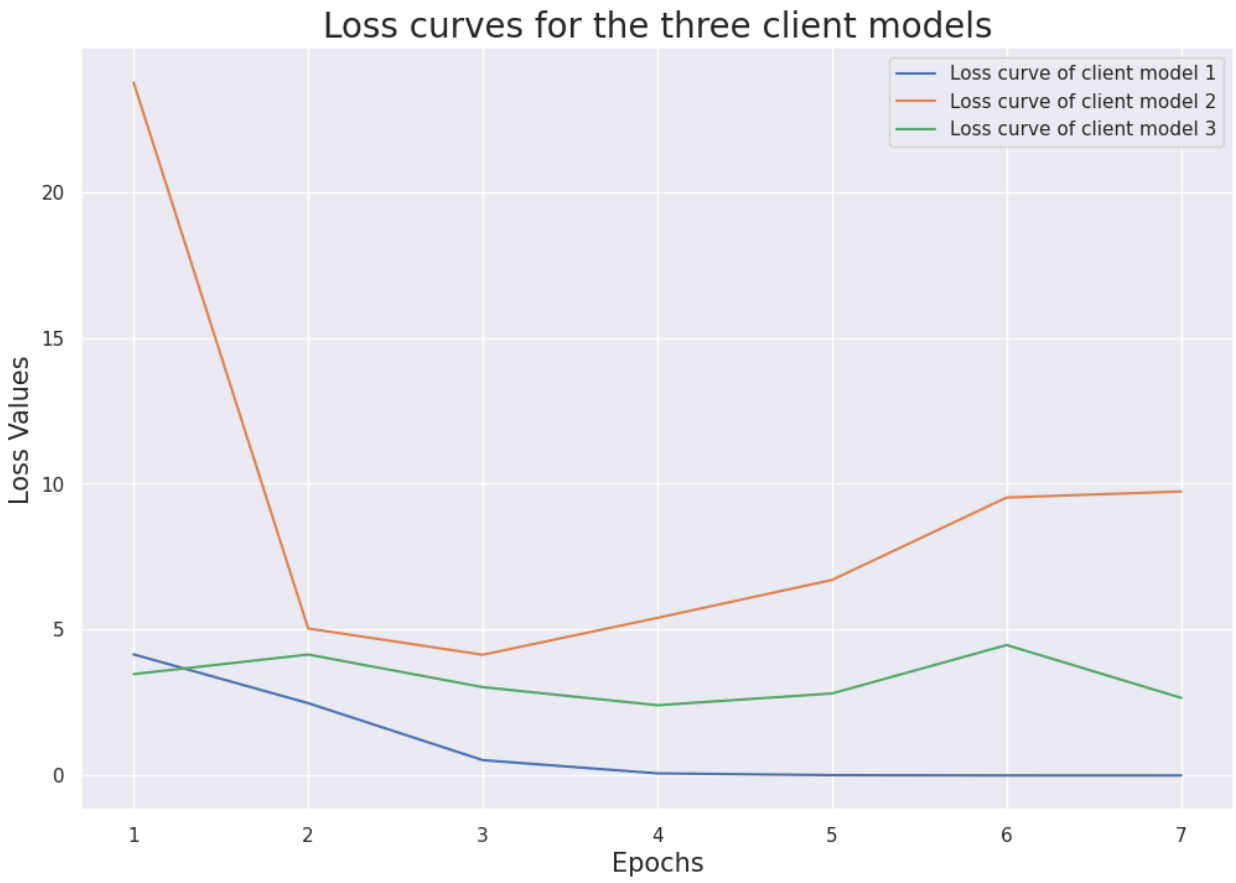
This function takes in each of the client model ,trains them on the training dataset of each of the respective dataset i.e. MNIST for client model 1,SVHN for client model 2,Colored MNIST for client model 3.

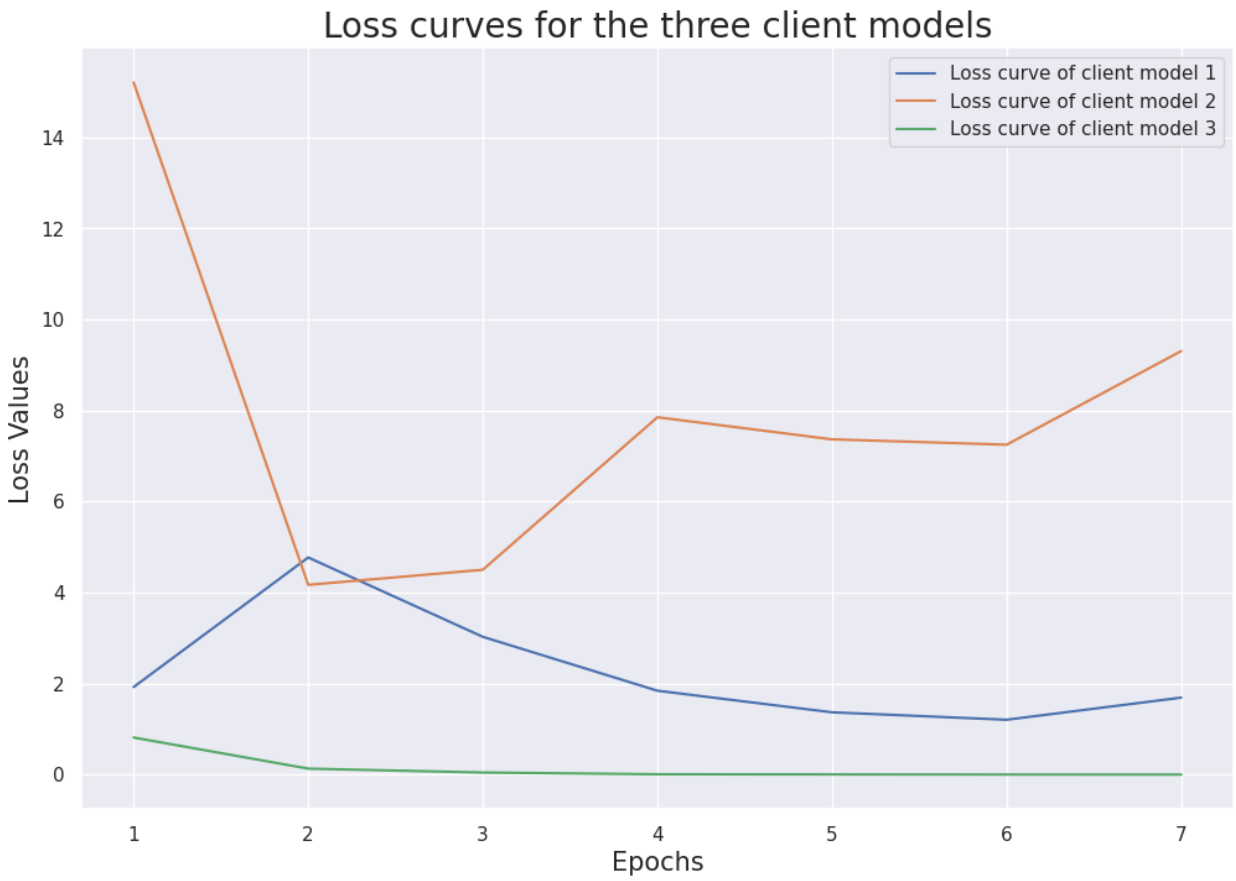
```python
def aggregate_loss(central_server, client_models):
    # Create a list of state dictionaries for all client models
    client_dicts = [model.state_dict() for model in client_models]

    # Aggregate the parameters of all client models by taking the mean
    new_dict = {}
    for k in central_server.state_dict():
        new_dict[k] = torch.stack([client_dicts[i][k].float() for i in range(len(client_dicts))], 0).mean(0)

    # Load the aggregated parameters to the global model
    central_server.load_state_dict(new_dict)

    # Load the aggregated parameters to all client models
    for model in client_models:
        model.load_state_dict(new_dict)
```

This function then takes all the client models, extracts their model dictionaries and then sums them out so as to produce the aggregate weight as described in the mathematical formulation below.Finally this function loads the parameters of the central server into each of the client models.

B.Report the class-wise accuracy results for all three datasets at their respective client side and at the central server also. Report overall classification Accuracy and Confusion Matrix.

After training for 7 epochs for 10 rounds with a batch size of 64 and using Adam optimizer with learning rate 0.001 we get.

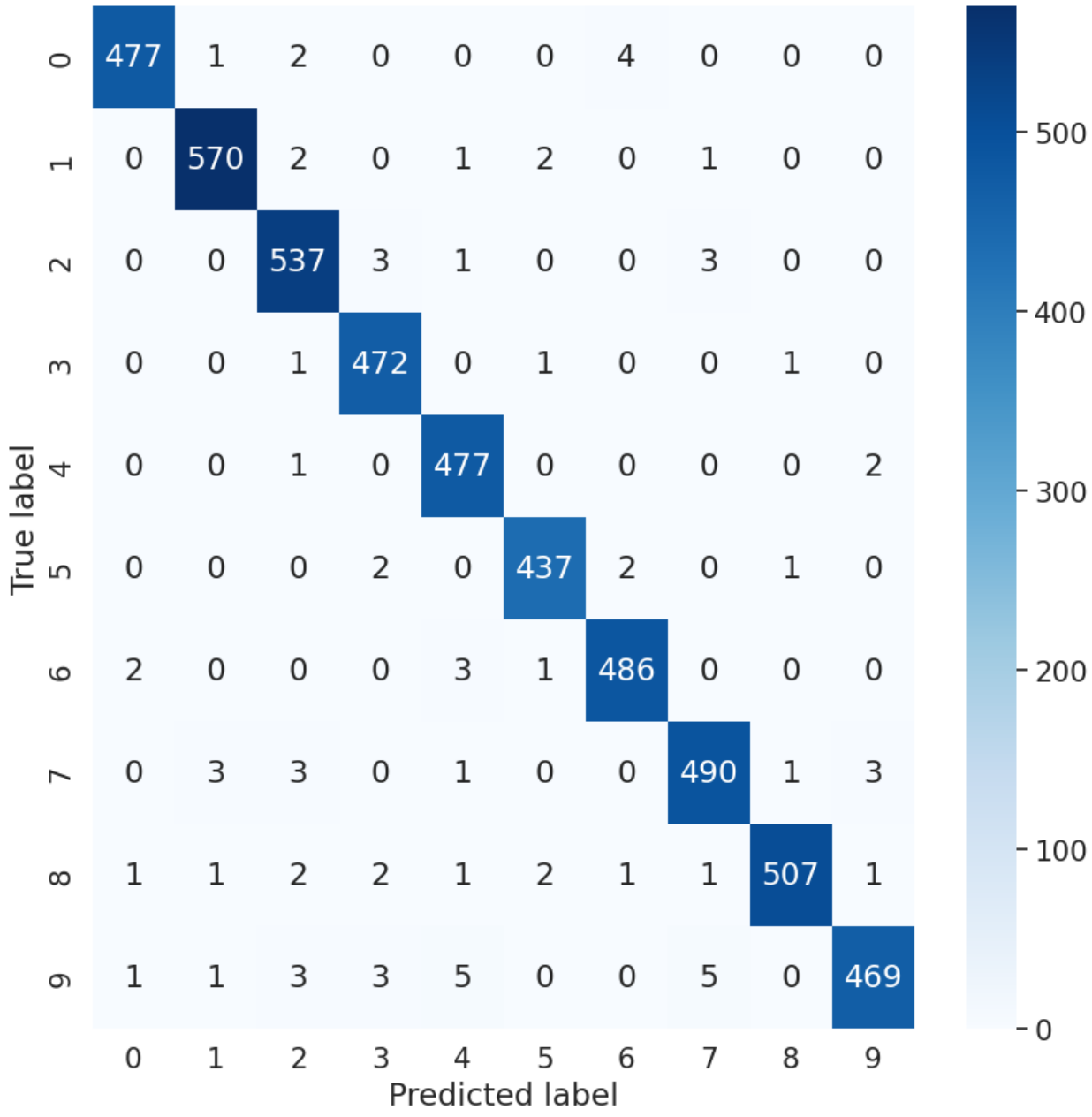




Class Wise accuracy,overall accuracy and confusion matrix scores at respective client side:

- MNIST at client 1

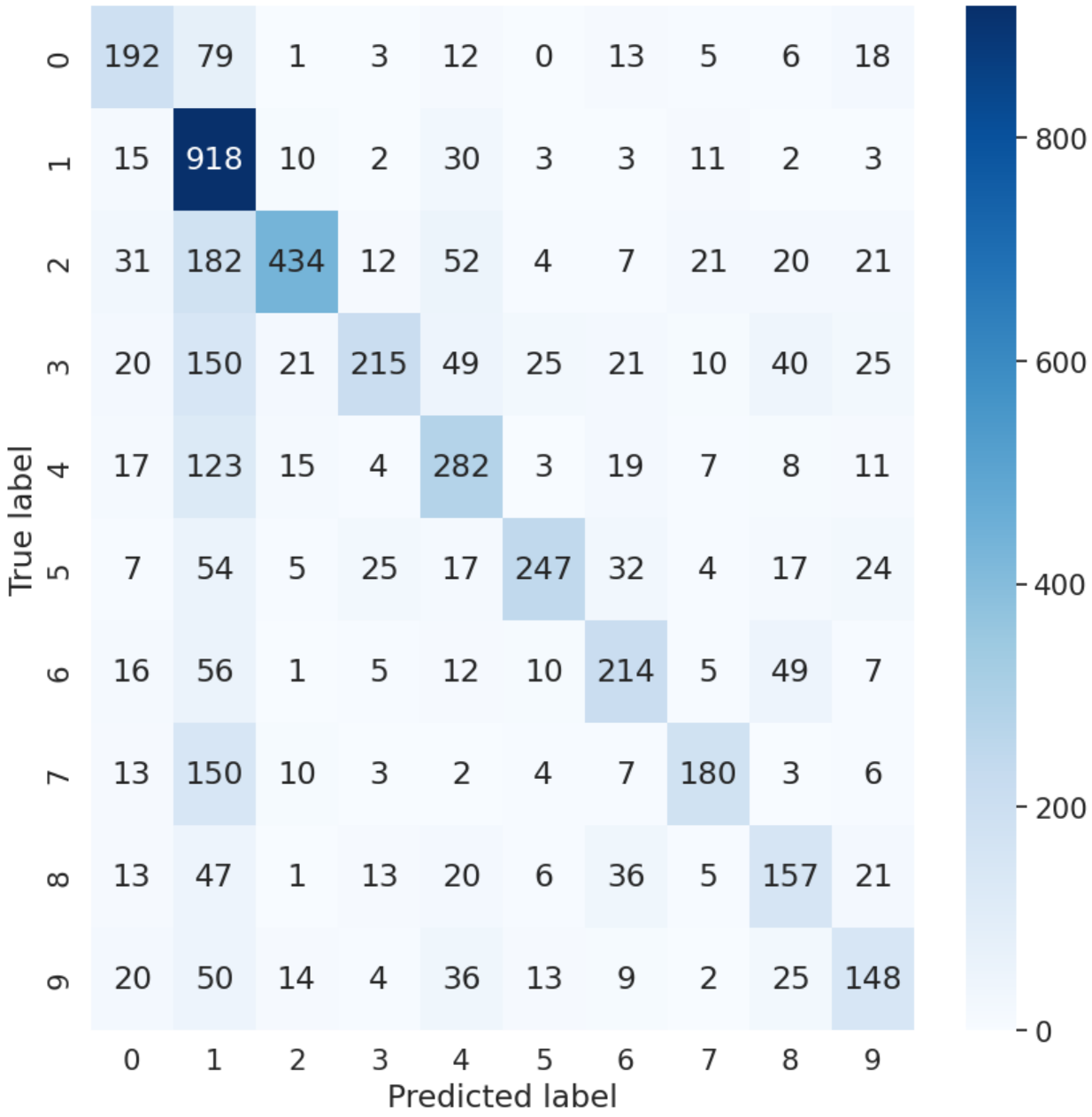
```
Overall Accuracy of client server 1: 98.44000000000001 %


Classwise Accuracy of client server 1:
For class 0:  98.55371900826447 %
For class 1:  98.95833333333334 %
For class 2:  98.71323529411765 %
For class 3:  99.36842105263159 %
For class 4:  99.375 %
For class 5:  98.86877828054298 %
For class 6:  98.78048780487805 %
For class 7:  97.80439121756487 %
For class 8:  97.6878612716763 %
For class 9:  96.30390143737166 %
```



- SVHN at client 2

```
Overall Accuracy of client server 2: 59.74 %
```

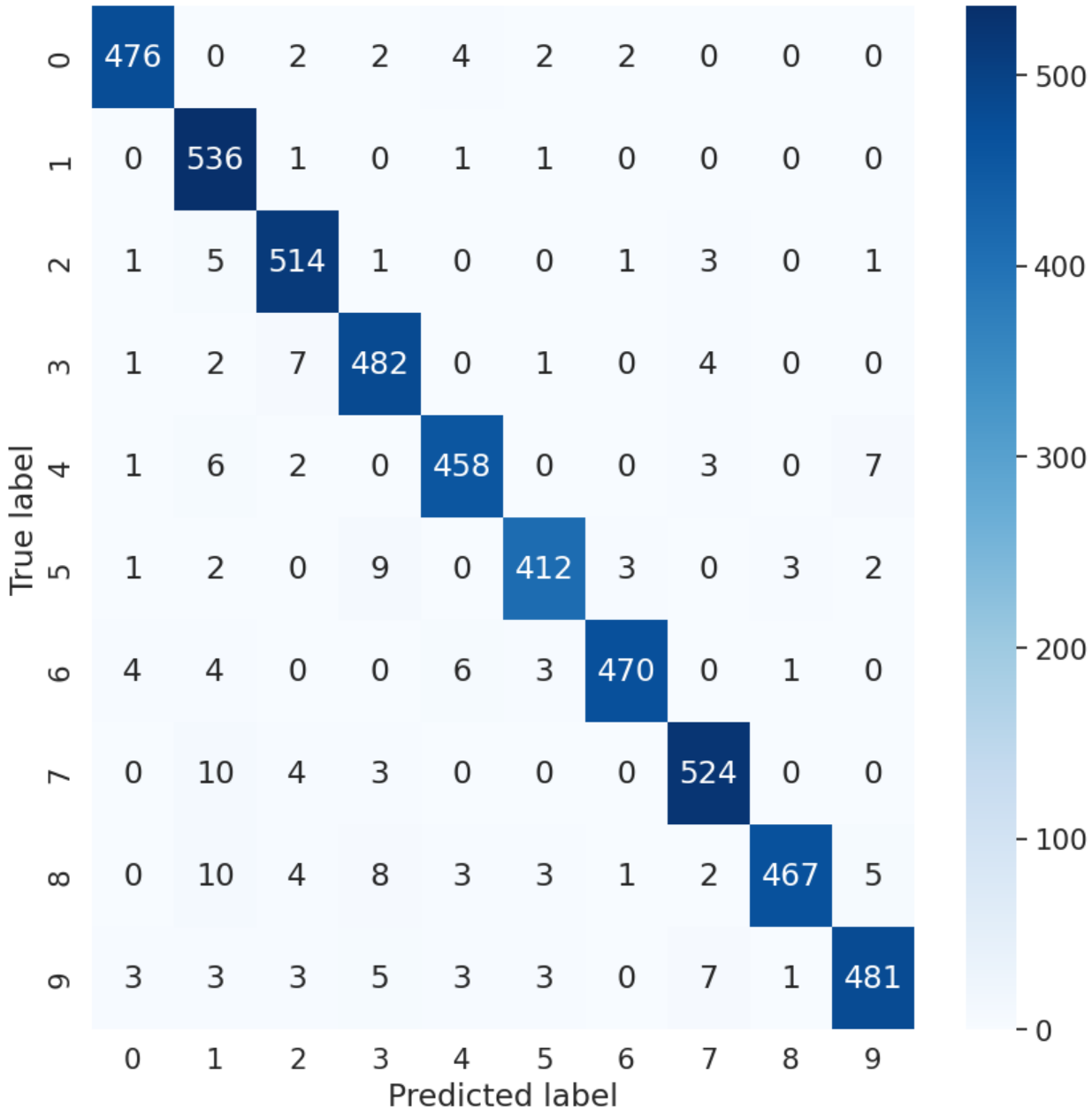
```
Classwise Accuracy of client server 2:
For class 0:  58.35866261398176 %
For class 1:  92.07622868605817 %
For class 2:  55.35714285714286 %
For class 3:  37.32638888888889 %
For class 4:  57.668711656441715 %
For class 5:  57.17592592592593 %
For class 6:  57.06666666666666 %
For class 7:  47.61904761904761 %
For class 8:  49.21630094043887 %
For class 9:  46.10591900311526 %
```

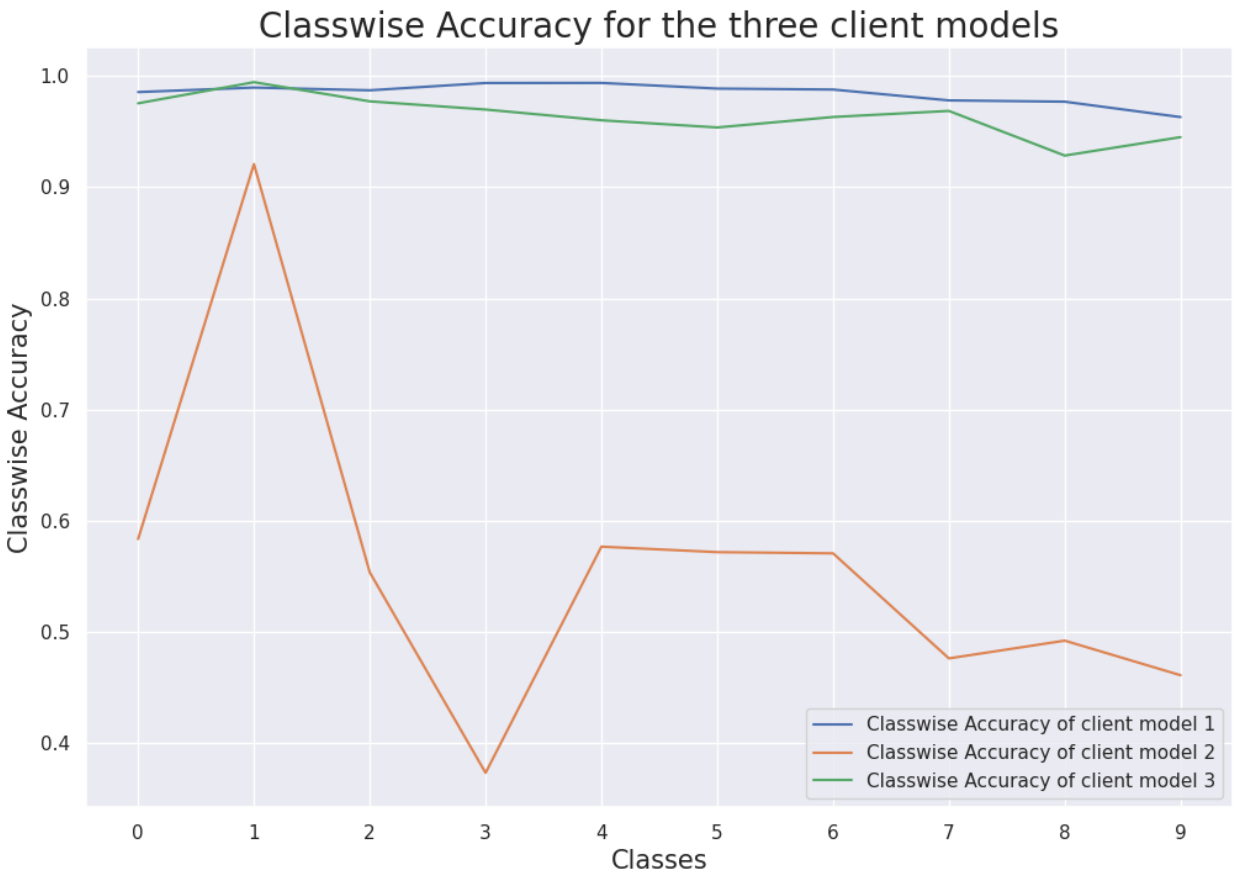


- Colored MNIST at client 3

```
Overall Accuracy of client server 3: 96.39999999999999 %
```

```
Classwise Accuracy of client server 3:
For class 0:  97.54098360655738 %
For class 1:  99.44341372912801 %
For class 2:  97.71863117870723 %
For class 3:  96.98189134808854 %
For class 4:  96.0167714884696 %
For class 5:  95.37037037037037 %
For class 6:  96.31147540983606 %
For class 7:  96.85767097966729 %
For class 8:  92.84294234592446 %
For class 9:  94.49901768172889 %
```


True label
Predicted label
0 1 2 3 4 5 6 7 8 9
476 0 2 2 4 2 2 0 0 0
0 536 1 0 1 1 0 0 0 0
1 5 514 1 0 0 1 3 0 1
1 2 7 482 0 1 0 4 0 0
1 6 2 0 458 0 0 3 0 7
1 2 0 9 0 412 3 0 3 2
4 4 0 0 6 3 470 0 1 0
0 10 4 3 0 0 0 524 0 0
0 10 4 8 3 3 1 2 467 5
3 3 3 5 3 3 0 7 1 481
500
400
300
200
100
0

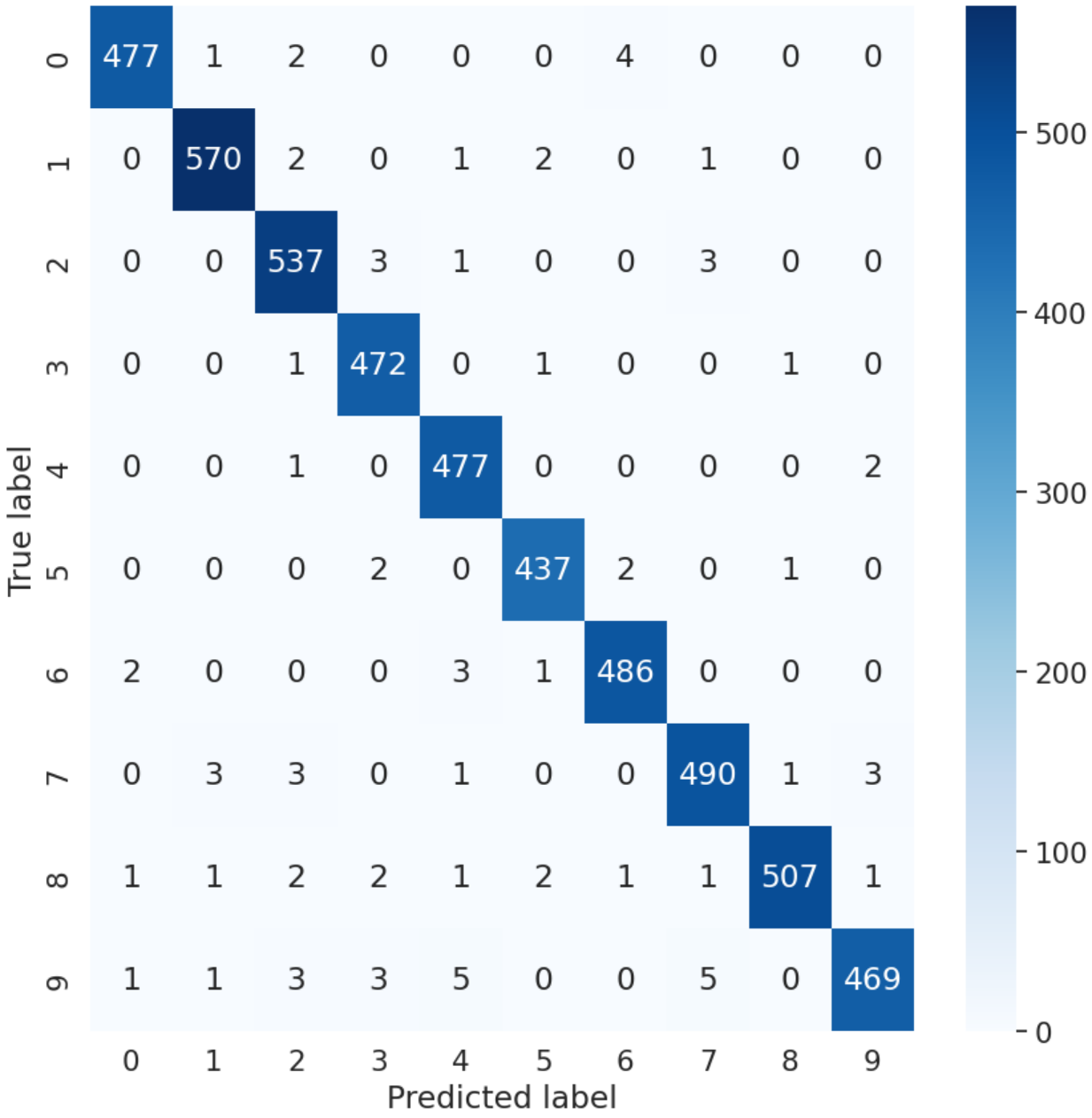


Class Wise accuracy scores at the central_server

- MNIST

```
Accuracy of central server on MNIST: 98.44000000000001 %
```

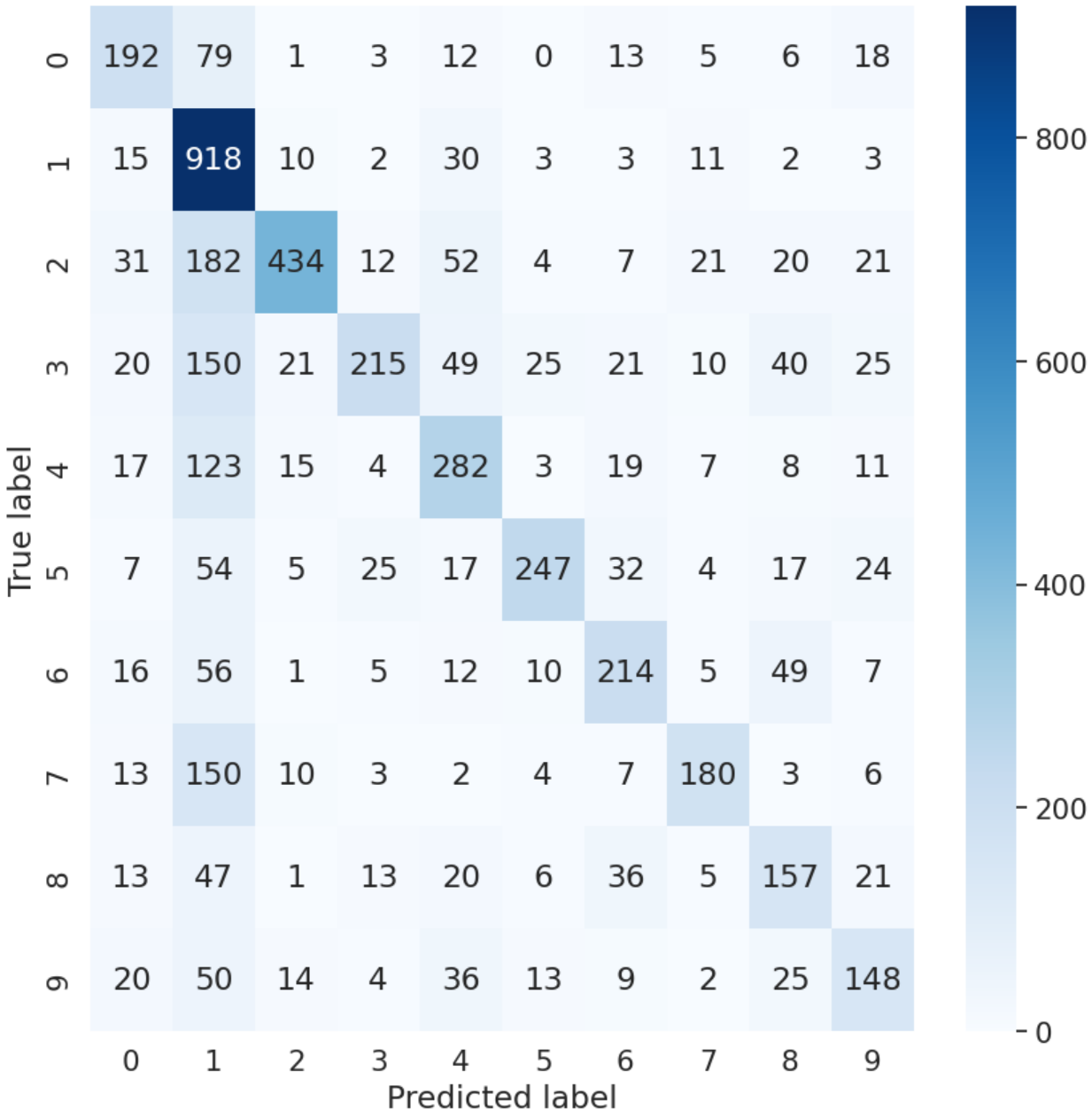
```
Classwise Accuracy of central server on MNIST:
For class 0:  98.55371900826447 %
For class 1:  98.95833333333334 %
For class 2:  98.71323529411765 %
For class 3:  99.36842105263159 %
For class 4:  99.375 %
For class 5:  98.86877828054298 %
For class 6:  98.78048780487805 %
For class 7:  97.80439121756487 %
For class 8:  97.6878612716763 %
For class 9:  96.30390143737166 %
```



- SVHN

```
Accuracy of central server on SVHN: 59.74 %
```

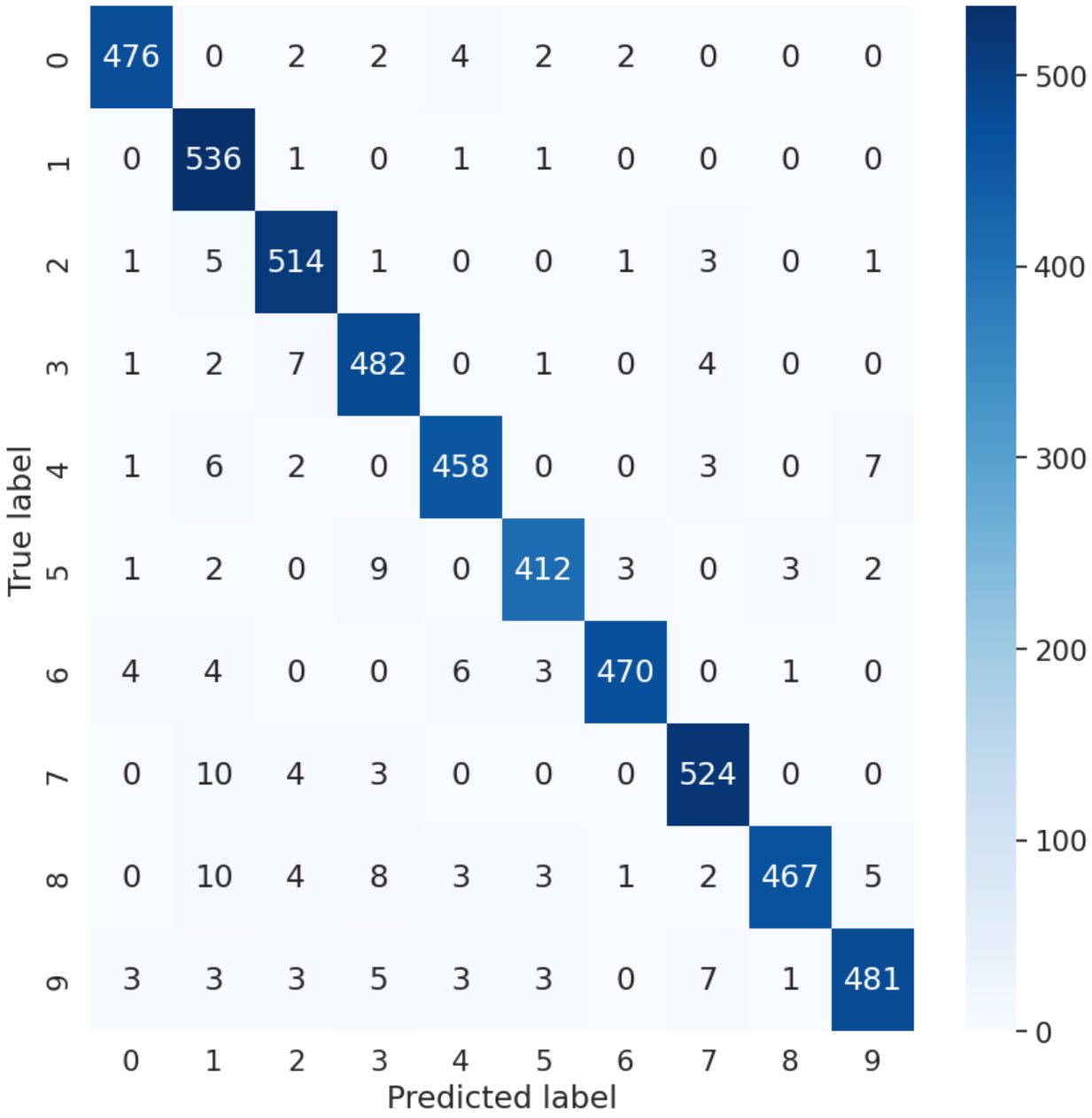
```
Classwise Accuracy of central server on SVHN:
For class 0:  58.35866261398176 %
For class 1:  92.07622868605817 %
For class 2:  55.35714285714286 %
For class 3:  37.32638888888889 %
For class 4:  57.668711656441715 %
For class 5:  57.17592592592593 %
For class 6:  57.06666666666666 %
For class 7:  47.61904761904761 %
For class 8:  49.21630094043887 %
For class 9:  46.10591900311526 %
```



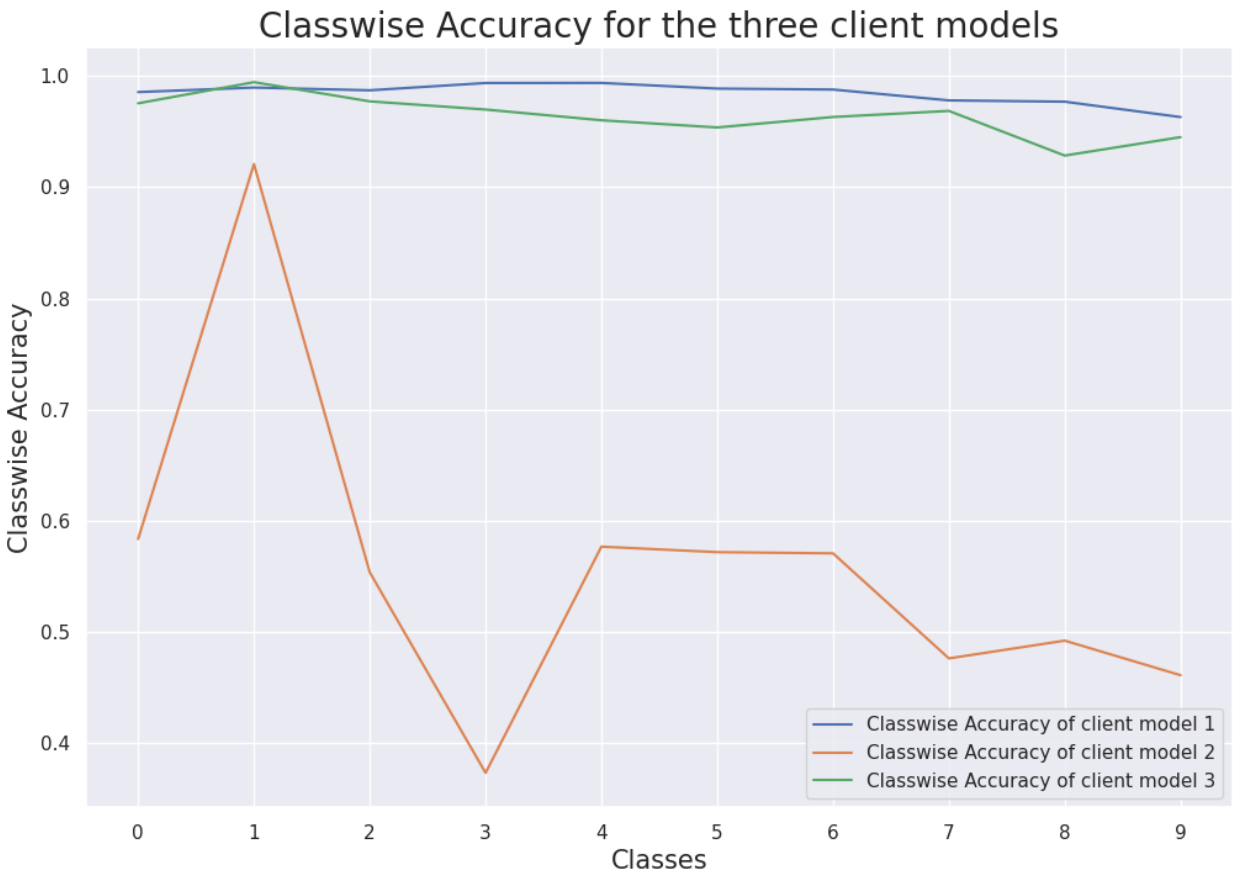
- Colored MNIST

```
Accuracy of central server on Coloured MNIST: 96.39999999999999 %
```

```
Classwise Accuracy of central server on Coloured MNIST:
For class 0:  97.54098360655738 %
For class 1:  99.44341372912801 %
For class 2:  97.71863117870723 %
For class 3:  96.98189134808854 %
For class 4:  96.0167714884696 %
For class 5:  95.37037037037037 %
For class 6:  96.31147540983606 %
For class 7:  96.85767097966729 %
For class 8:  92.84294234592446 %
For class 9:  94.49901768172889 %
```



Note-We are getting the same results as above as each of the client models have been updated by the central model/central server.

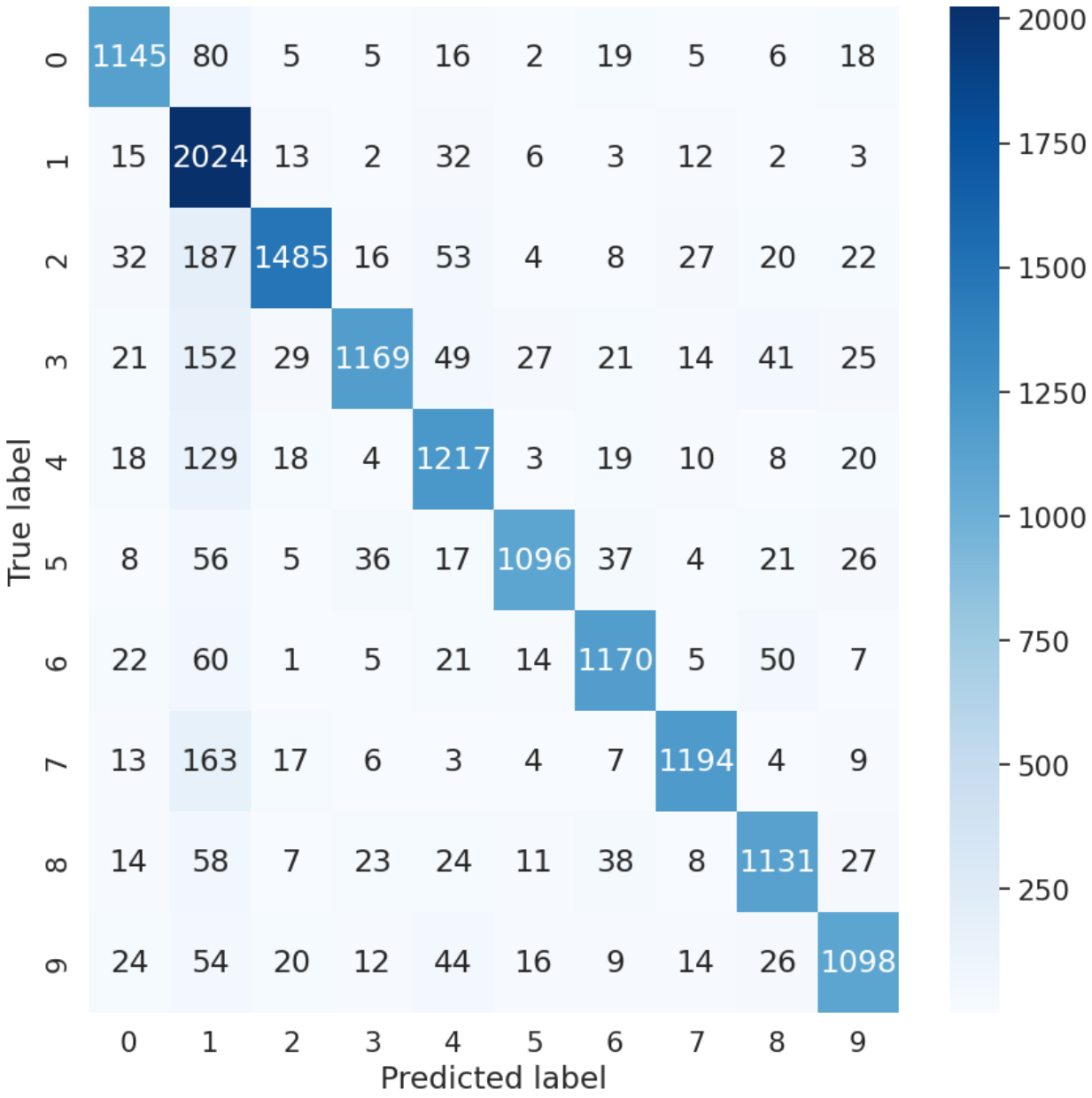


Reporting the overall classification accuracy and the confusion matrix

We will do this by testing on the central_server on the combination of all three datasets

```
Overall Accuracy:  84.86 %
```

```
Overall Classwise Accuracy:
For class 0:  88.00922367409684 %
For class 1:  95.83333333333334 %
For class 2:  80.09708737864078 %
For class 3:  75.51679586563309 %
For class 4:  84.16320885200554 %
For class 5:  83.92036753445635 %
For class 6:  86.34686346863468 %
For class 7:  84.08450704225352 %
For class 8:  84.3400447427293 %
For class 9:  83.37129840546697 %
```



C.Write the mathematical explanation of the function used to perform the aggregation at the central server.

DEVI PRASAD MAHARATHY
B20AI053

PAGE NO.:

c) Mathematical Explanation:-

Each client computes w_{t+1}^{k} ← client number
← epoch number.

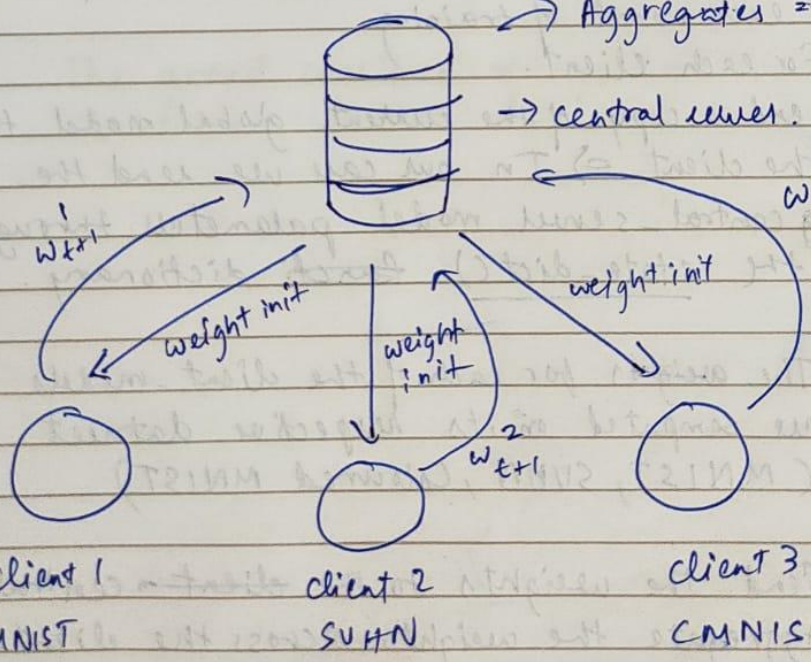
The central server then aggregates this by

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k}$$

↳ Final weights which is updated on the central-server.

It takes a weighted average for the weights recieved from the client-models

d)



Aggregation happens after each client has replied.

D.Write the detailed explanation of the federated learning algorithm with the diagrammatic representation used for the above solution.

DEVI PRASAD MAHARATHY.
B20AI053

PAGE NO.:

D) The type federated learning used here is :-
Federated Averaging

It is a popular federated learning algorithm to aggregate the model parameters of multiple client-models into a single central-model/ central-server. The goal of federated ~~learning~~ averaging is to train a global model that can generalize well across all clients while preserving the privacy of their local data.

Algorithm:-

1. Initialization of global model with initial parameters.
2. For each round of training
   → For each client.
   Send a copy of the current global model to the client ⇒ In our case we send the 'central_server model parameters through the state_dict() ~~function~~ dictionary.

   → The weights for each of the client-models are computed on its respective dataset (MNIST, SVHN, Coloured MNIST)

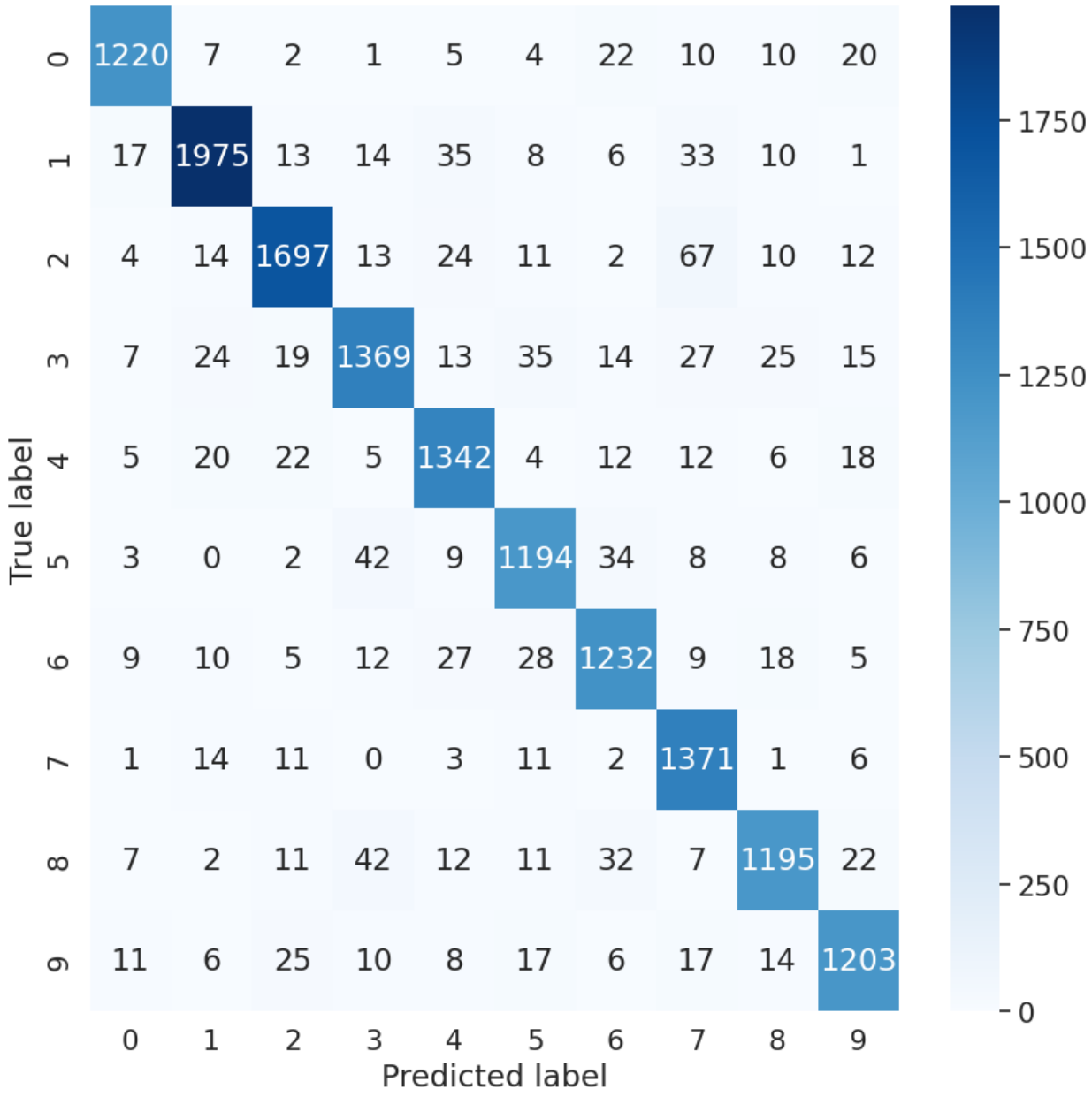
   → Send the weights to the ~~client~~ central server.
   → Aggregate the weights across the client-models into a single weight dictionary.
   → Update the global model by the aggregated weight dictionary.
3. Repeat this process.

First each of the client model loads the parameters of the global model using the model.state_dict() function and train each of the client model is trained for 7 epochs.This process is repeated for 10 rounds.

E.Compare the results of overall accuracy in federated setup with the baseline results calculated by combining all the datasets and training in non-federated setup. Do you observe any decrease/increase in accuracy for both the setups? State your answer with proper reasoning.

We have trained the model for 30 epochs

```
Overall Accuracy:  91.98666666666668 %


Overall Classwise Accuracy:
For class 0:  93.77401998462722 %
For class 1:  93.51325757575758 %
For class 2:  91.53182308522115 %
For class 3:  88.43669250645995 %
For class 4:  92.80774550484094 %
For class 5:  91.42419601837672 %
For class 6:  90.92250922509226 %
For class 7:  96.54929577464789 %
For class 8:  89.11260253542133 %
For class 9:  91.34396355353076 %
```



We observe that the training in non federated setup yields better results than federated setup.The client corresponding to the SVHN dataset in particular is giving the worst result out of the three client models of around only 59.74%.It has such a huge impact on the federated learning setup that I tried the federated learning approach with only the two client models with MNIST and colored MNIST respectively.It yielded an accuracy score of

around 90-95 %.But with the incorporation of the client model with SVHN dataset the accuracy came down to only 84.26%.

Reason: The main reason for such degradation is the fact that SVHN has a distribution which is different from that of MNIST and Colored MNIST which have more or less the same distribution (other than the fact that one is colored).This is a case of Non-IID data in federated learning.This is the reason why federated learning with only MNIST and Colored MNIST gave a result(90-95%) much better than with MNIST and Colored MNIST and SVHN(84.26%).

References:

- https://github.com/yonetaniryo/federated_learning_pytorch/blob/master/FL_pytorch.ipynb
- Federated Learning slides shared in class