Assignment: 3

Procedure

- Get dataset and trim it to 10000 for train and 5000 for test at all clients and federated server and central server
- Now at all clients train clients model and pass it to federated server.
- While train model at client note class wise accuracy, overall accuracy and confusion matrix of client and similar at federated server and central model
- For aggregation I used FedAVG at fedmodel
- After training fedmodel and centralized model compare the results

Part A

- Implemented in code

Part B

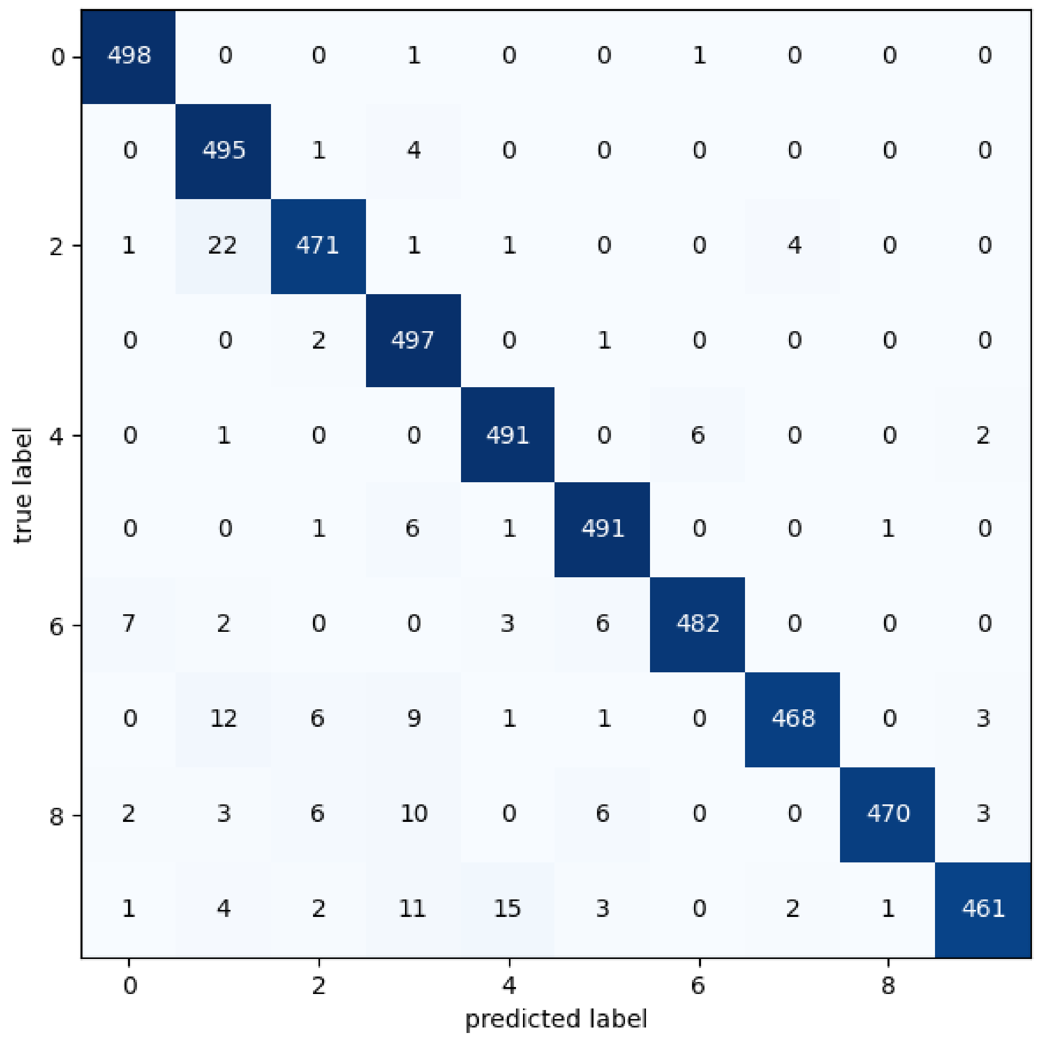
Result:

Client 1:

```
 Overall Accuracy of Client Model: 0.9085

Class wise Accuracy:
Class 0: 0.9980
Class 1: 0.9880
Class 2: 0.9840
Class 3: 0.9900
Class 4: 0.9760
Class 5: 0.9740
Class 6: 0.9680
Class 7: 0.9700
Class 8: 0.9740
Class 9: 0.9500

Confusion Metric
```

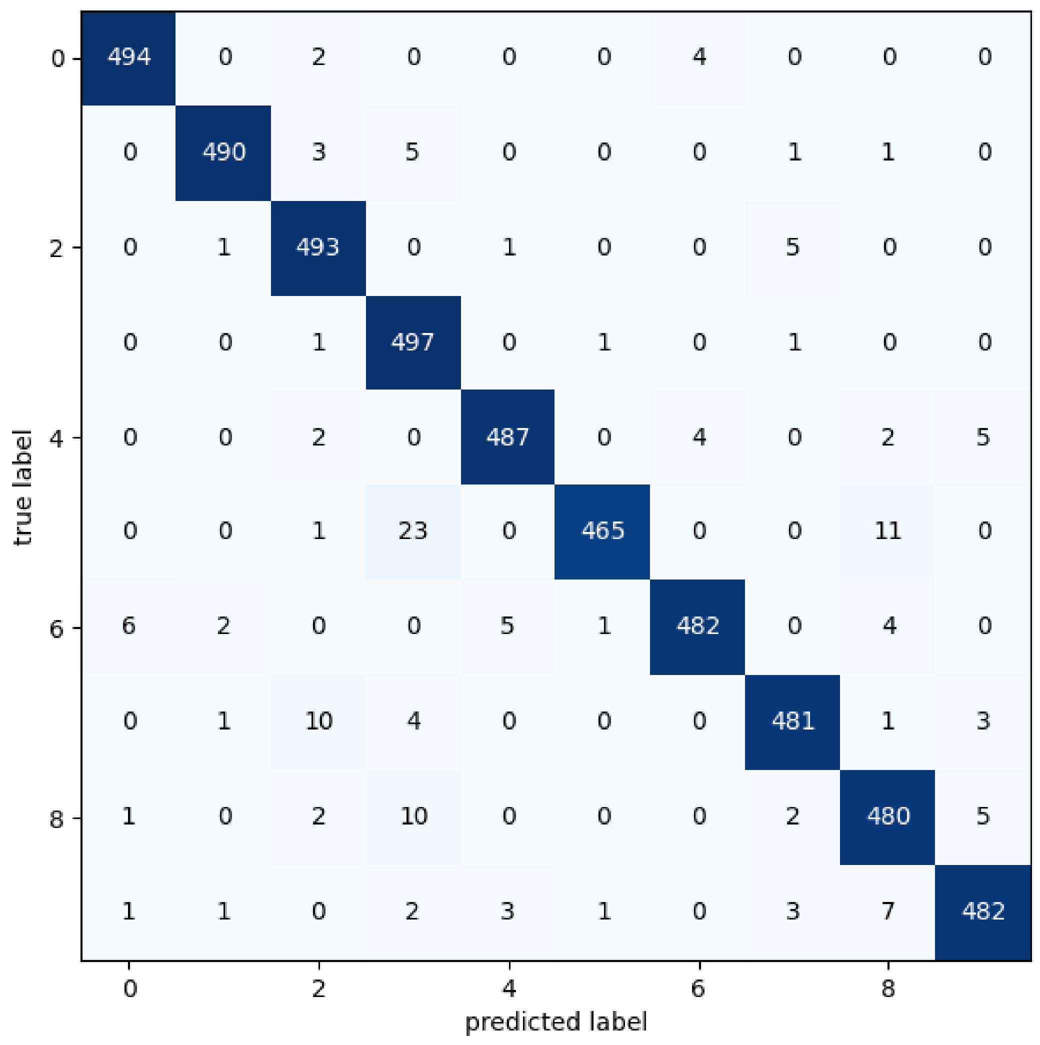


Client 2:

```
Overall Accuracy of Client Model: 0.8994

Class wise Accuracy:
Class 0: 0.9980
Class 1: 0.9860
Class 2: 0.9820
Class 3: 0.9940
Class 4: 0.9700
Class 5: 0.9460
Class 6: 0.9640
Class 7: 0.9500
Class 8: 0.9520
Class 9: 0.9580
```

```
Confusion Metric
```

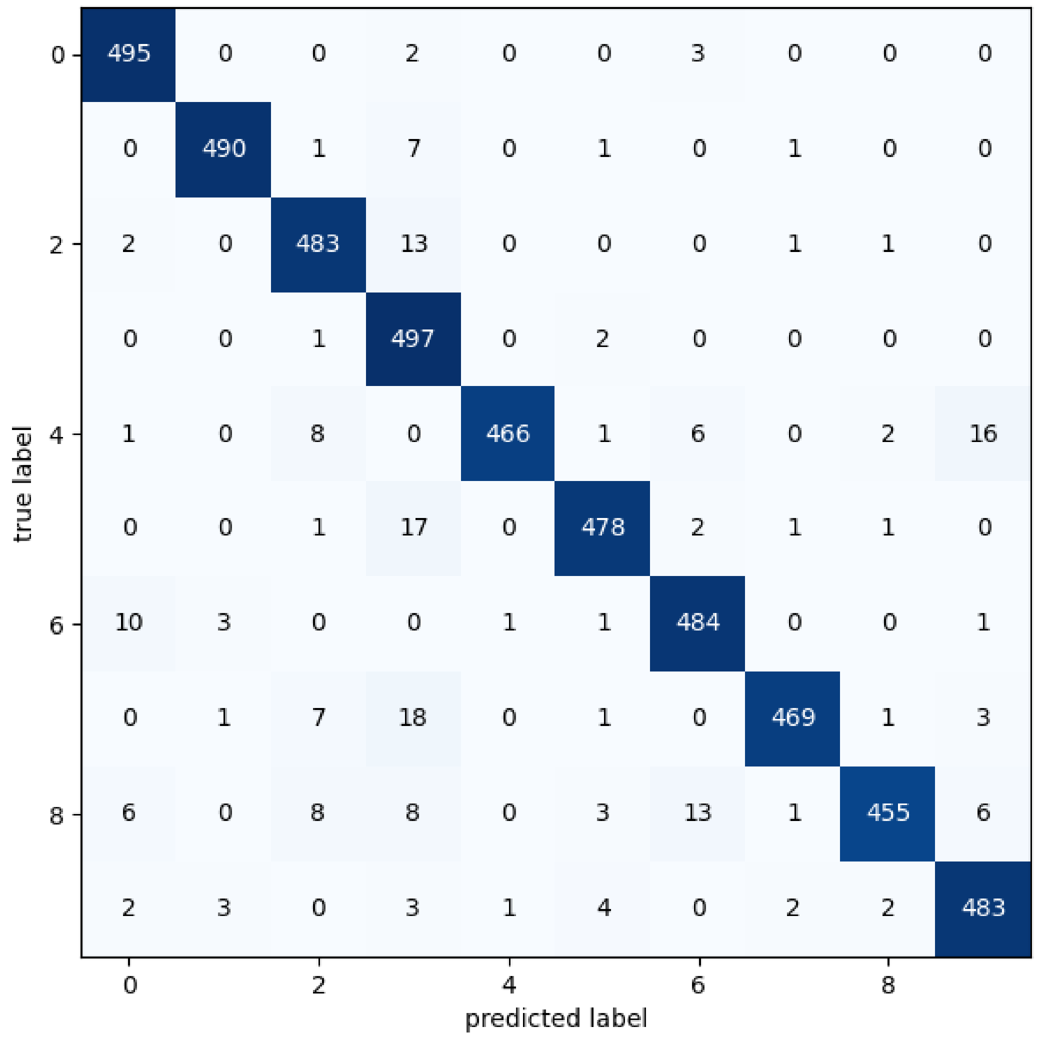


Client 3:

```
Overall Accuracy of Client Model: 0.8899

Class wise Accuracy:
Class 0: 0.9900
Class 1: 0.9800
Class 2: 0.9660
Class 3: 0.9940
Class 4: 0.9320
Class 5: 0.9560
Class 6: 0.9680
Class 7: 0.9380
Class 8: 0.9100
Class 9: 0.9660

Confusion Metric
```



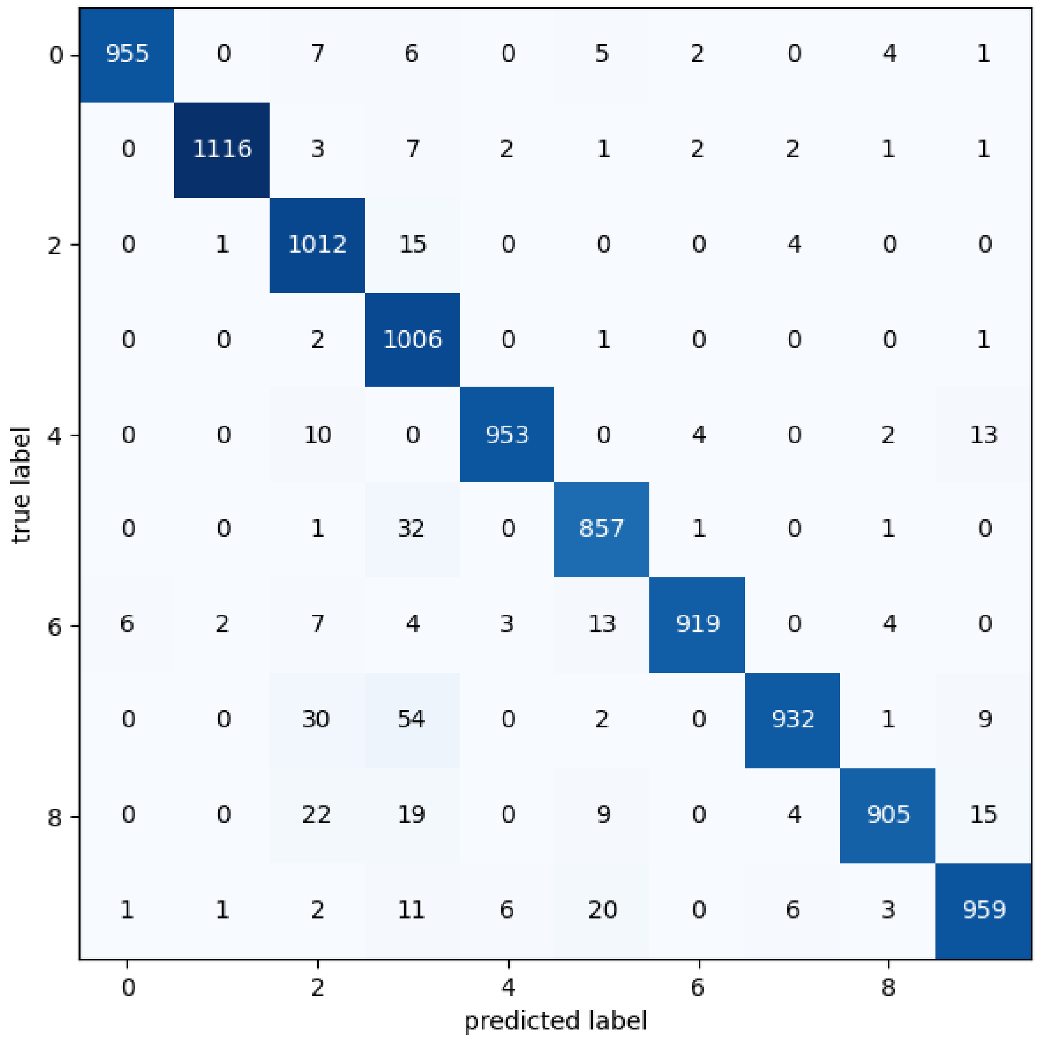
	0		2		4		6		8	
0	495	0	0	2	0	0	3	0	0	0
	0	490	1	7	0	1	0	1	0	0
2	2	0	483	13	0	0	0	1	1	0
	0	0	1	497	0	2	0	0	0	0
4	1	0	8	0	466	1	6	0	2	16
	0	0	1	17	0	478	2	1	1	0
6	10	3	0	0	1	1	484	0	0	1
	0	1	7	18	0	1	0	469	1	3
8	6	0	8	8	0	3	13	1	455	6
	2	3	0	3	1	4	0	2	2	483

Federated Server Model:

```
Overall Accuracy of Client Model: 0.9376

Class wise Accuracy:
Class 0: 0.9745
Class 1: 0.9833
Class 2: 0.9806
Class 3: 0.9960
Class 4: 0.9705
Class 5: 0.9608
Class 6: 0.9593
Class 7: 0.9066
Class 8: 0.9292
Class 9: 0.9504

Confusion Metric
```



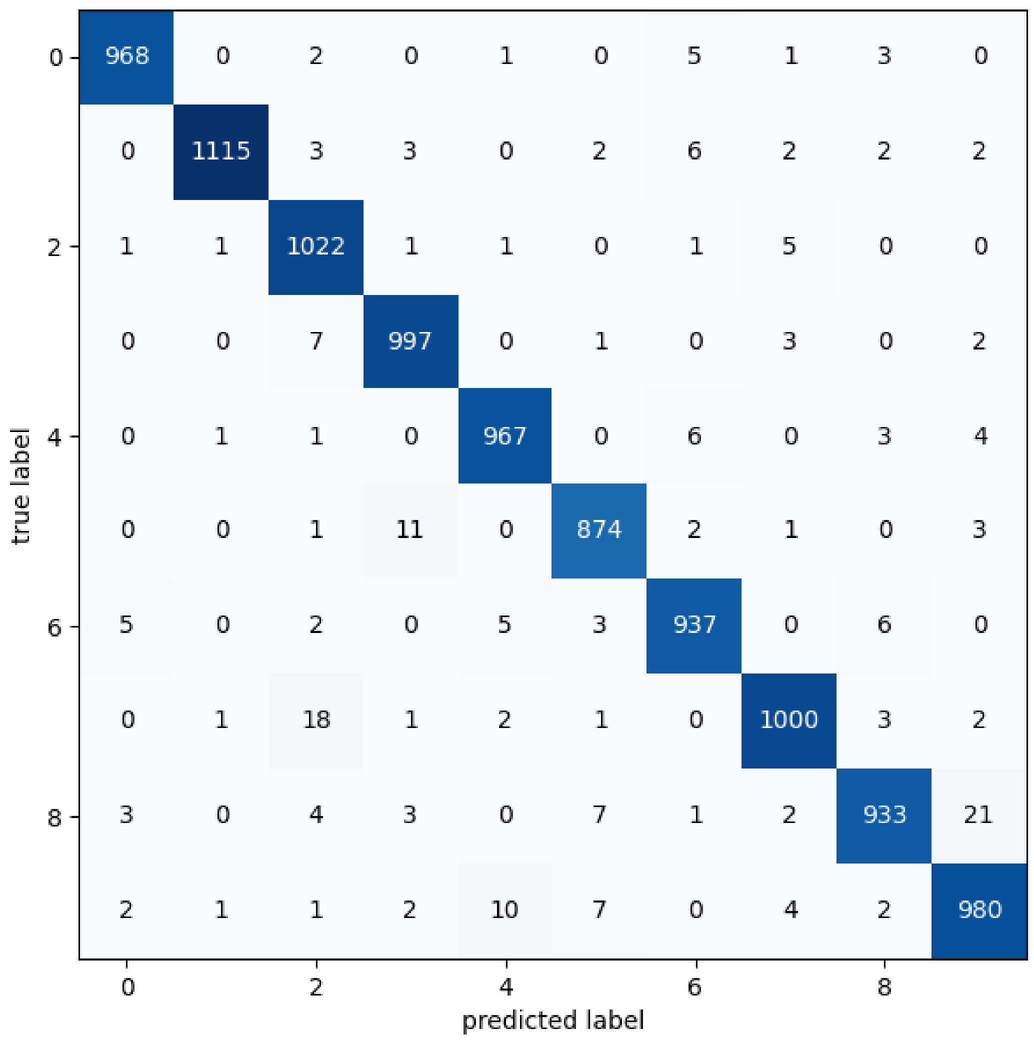
	0	1	2	3	4	5	6	7	8	9
0	955	0	7	6	0	5	2	0	4	1
1	0	1116	3	7	2	1	2	2	1	1
2	0	1	1012	15	0	0	0	4	0	0
3	0	0	2	1006	0	1	0	0	0	1
4	0	0	10	0	953	0	4	0	2	13
5	0	0	1	32	0	857	1	0	1	0
6	6	2	7	4	3	13	919	0	4	0
7	0	0	30	54	0	2	0	932	1	9
8	0	0	22	19	0	9	0	4	905	15
9	1	1	2	11	6	20	0	6	3	959

Centralized Based Server:

```
Overall Accuracy of Client Model: 0.9552

Class wise Accuracy:
Class 0: 0.9878
Class 1: 0.9824
Class 2: 0.9903
Class 3: 0.9871
Class 4: 0.9847
Class 5: 0.9798
Class 6: 0.9781
Class 7: 0.9728
Class 8: 0.9579
Class 9: 0.9713
```

Confusion Metric

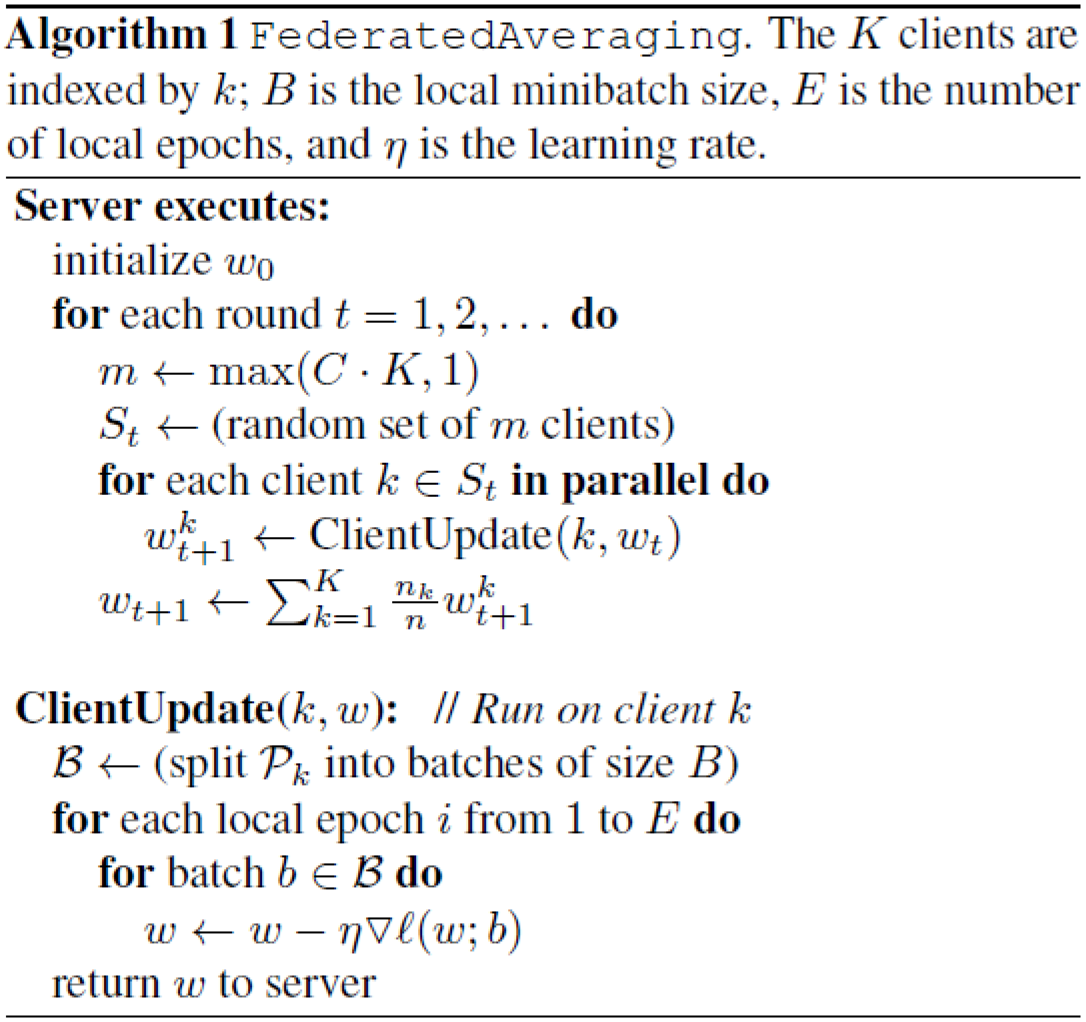


	0		2		4		6		8	
0	968	0	2	0	1	0	5	1	3	0
	0	1115	3	3	0	2	6	2	2	2
2	1	1	1022	1	1	0	1	5	0	0
	0	0	7	997	0	1	0	3	0	2
4	0	1	1	0	967	0	6	0	3	4
	0	0	1	11	0	874	2	1	0	3
6	5	0	2	0	5	3	937	0	6	0
	0	1	18	1	2	1	0	1000	3	2
8	3	0	4	3	0	7	1	2	933	21
	2	1	1	2	10	7	0	4	2	980

true label
predicted label

Part C:

- In federated learning, the FedAVG (Federated Averaging) algorithm is commonly used to aggregate the model updates from the client devices at the central server. The FedAVG algorithm computes the average of the model parameters across all the participating client devices, weighted by the number of training examples on each device. The mathematical explanation of the FedAVG algorithm is as follows:
- Let N be the total number of participating client devices in the federated learning system, and let w_t be the model parameters (e.g., weights and biases) of the i-th client device after training on its local data. Each client device trains on a subset of the overall training data, denoted as n_k. Let n = |n_k| be the number of training examples on the i-th device.



Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize $w_0$
  for each round $t = 1, 2, \ldots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate($k, w$):   // Run on client $k$
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

- The FedAVG algorithm aggregates the model updates from the client devices by computing the weighted average of the model parameters as follows:

- where w_avg is the averaged model parameters at the central server, and |n| is the total number of training examples across all client devices.
- The term (|n-k|/|n|) represents the weight assigned to the i-th device in the aggregation process, which reflects the proportion of training examples on that device relative to the total number of training examples across all devices. The FedAVG algorithm applies this weighting to ensure that devices with more data have a greater influence on the aggregated model than those with less data.
- The FedAVG algorithm is an iterative process that repeats until convergence or a stopping criterion is met. In each iteration, the client devices train their local models on their respective data, compute their model updates, and send them to the central server for aggregation using the FedAVG algorithm. The central server then updates its global model with the averaged parameters and sends the updated model back to the client devices for further training. This process continues until the global model converges to a satisfactory level of accuracy.

Part D:

- In federated learning, multiple client devices participate in training a shared machine learning model without sharing their data with each other.
- After each client trains its local model on its respective data, the model's parameters are sent to a central server for aggregation.
- The FedAVG algorithm is a widely-used aggregation technique in federated learning that computes the weighted average of the model parameters across all client devices.

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

- The weighted average is computed by assigning a weight to each client device that is proportional to the number of training examples on that device.
- The weight assigned to each device ensures that devices with more data have a greater influence on the aggregated model than those with less data.
- The aggregated model parameters are then used as the global model that is sent back to each client for further training.
- The FedAVG algorithm is an iterative process that repeats until convergence or a stopping criterion is met.

- At each iteration, the client devices train their local models on their respective data, compute their model updates, and send them to the central server for aggregation using the FedAVG algorithm.
- The central server then updates its global model with the averaged parameters and sends the updated model back to the client devices for further training.
- This process continues until the global model converges to a satisfactory level of accuracy.
- In essence, the FedAVG algorithm computes a weighted average of the model parameters across all participating client devices in order to aggregate model updates in federated learning. The privacy and security of the various data sets are maintained while ensuring that devices with more data have a bigger impact on the aggregated model.

Part E:

	Federated	**Centralized**
Overall Accuracy	0.9376	0.9552
Class wise accuracy		
Class 0	0.9745	0.9878
Class 1	0.9833	0.9824
Class 2	0.9806	0.9903
Class 3	0.9960	0.9871
Class 4	0.9705	0.9847
Class 5	0.9608	0.9798
Class 6	0.9593	0.9781
Class 7	0.9066	0.9728
Class 8	0.9292	0.9579
Class 9	0.9504	0.9713

- If we tried to train client's models in real time and then pass the weight to a federated server, we could achieve similar results, or it may be possible to get better accuracy than with a centralized model.

- In class wise accuracy all class accuracy are similar they have only just 1-2 % differences. But for class 7 we get 7% different.
- We have compared both federated and centralized models and have observed that there is a difference of 2% accuracy between them because in centralisd model we train test and evaluate all in the same system at a serial instance and in federated model there is a security issue to clients so in order to maintain it they send the train weights to the server. Next all the n values are combined in terms of average and hence a common final weight has been generated.
- Hence it is the average of all client model weights so accuracy of 1-3 % are considered to be negligible and can be ignored.

A federated model could be less accurate than a centralized model for a number of reasons, including the following:

- Data distribution: In federated learning, the data is split across a number of client devices, and each one builds a local model using the data that belongs to it. In comparison to a centralized model that has access to all data, the data distribution may not be representative of the distribution of data as a whole, which could lead to lower accuracy.
- Limited communication rounds: Only a certain number of communication rounds are allowed between client devices and the central server during federated learning. The federated approach could be less accurate than the centralized model if there aren't enough communication rounds to get a decent answer.
- Heterogeneous devices: Devices that are heterogeneous: In federated learning, client devices may have varying hardware setups, network setups, or training data sizes. In comparison to a centralized model, this heterogeneity may cause updates to the model to vary and convergence to occur more slowly.
- Privacy preservation: Federated learning is intended to protect the confidentiality and privacy of individual data sets. Neither the central server nor the client devices have access to the training data. In comparison to a centralized model with access to all data, this privacy-preserving mechanism may restrict the amount of data that is available for training and lead to lower accuracy.

It's vital to remember that not all tasks will benefit from federated learning more than centralized learning. Federated learning aims to make model training collaborative while maintaining security and privacy. A successful federated model would be one that maintains privacy while maintaining an acceptable level of accuracy.

Note: I also provide a video to explain the code. Thank You:)

References:

https://towardsdatascience.com/federated-learning-a-simple-implementation-of-fedavg-federated-averaging-with-pytorch-90187c9c9577

https://www.kaggle.com/code/puru98/federated-learning-pytorch

https://github.com/AshwinRJ/Federated-Learning-PyTorch/tree/master/src

https://flower.dev/docs/example-pytorch-from-centralized-to-federated.html

https://shreyansh26.github.io/post/2021-12-18_federated_optimization_fedavg/

https://www.dam.brown.edu/drp/talks/OhJoonKwon.pdf

Also from class videos recording.