Video link :
https://drive.google.com/file/d/16v3Fcq7gBFHT2nn2_rm8zSUOwnfiORgx/view?usp=sharing

Q1.1: Use the use case in the class:
a. Add more Dense layers to the existing code and check how the accuracy changes.

By adding more layers and changing activation functions, the accuracy is either reduced or the same, which shows that more input layers only decreases the performance of the model.

Q1.2: Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.

Tried 2 different models, first one is basic 2 layer model that gave 62.27% accuracy and later modified the model with few changes like below but still couldn't improve the model accuracy:

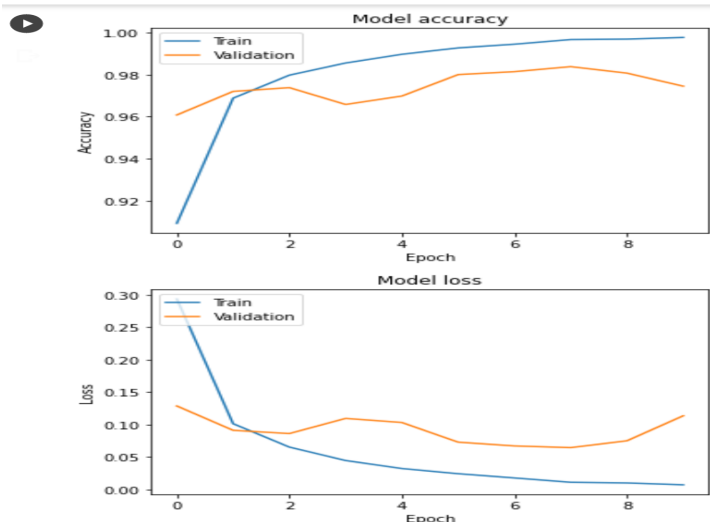
1. Added a Dropout layer after each Dense layer to reduce overfitting.
2. Increased the number of epochs to 50 to allow the model to train for longer.
3. Decreased the learning rate to 0.001 to improve convergence and prevent oscillations.
4. Increased the batch size to 64 to improve convergence and training speed.

Q1.3: Normalize the data before feeding the data to the model and check how the normalization changes your accuracy (code given below).
from sklearn.preprocessing import StandardScaler sc = StandardScaler()

Although normalization can improve model performance by bringing input features to a similar scale, we do not see any improvement in breast cancer dataset here. The accuracy is 62.28%.

Q2.1: Plot the loss and accuracy for both training data and validation data using the history object in the source code.

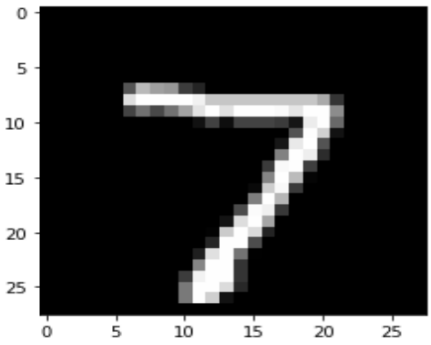
Model Accuracy and Loss graphs are below:



Q2.2: Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

The prediction for one image is as below:

```
[10]
      # Plot the first image in the test dataset
      plt.imshow(test_images[0], cmap='gray')
      plt.show()
```



```
[11]  image = test_data[0].reshape(1, dimData)
      prediction = model.predict(image)
      print("Model prediction: ", np.argmax(prediction))
```

```
1/1 [==============================] - 0s 91ms/step
Model prediction:  7
```

Q2.3: We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

Changing the model by increasing hidden layers and activation to tanh & sigmoid, the accuracy improved from 97.45% to 97.85%.

Q2.4: Run the same code without scaling the images and check the performance?

Not scaling data here has reduced the accuracy score to 96.44%. So it is good we scale the data for this particular dataset.