Experiment 5

Aim : Design and implement a fully connected deep neural network for a classification application

```
import numpy as np
import torch
import torchvision
import matplotlib.pyplot as plt
from time import time
from torchvision import datasets, transforms
from torch import nn, optim
```

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras import backend as K
#  to split the data of training and testing sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
batch_size = 128
num_classes = 10
epochs = 10
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,
3),activation='relu',input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,optimizer=ker
as.optimizers.Adadelta(),metrics=['accuracy'])
```

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# conversion of class vectors to matrices of  binary class *
```

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=
(x_test, y_test))

Epoch 1/10
469/469 [==============================] - 223s 472ms/step - loss:
2.2689 - accuracy: 0.1627 - val_loss: 2.2182 - val_accuracy: 0.4299
Epoch 2/10
469/469 [==============================] - 215s 458ms/step - loss:
2.1847 - accuracy: 0.3157 - val_loss: 2.1107 - val_accuracy: 0.6623
Epoch 3/10
469/469 [==============================] - 218s 465ms/step - loss:
2.0712 - accuracy: 0.4571 - val_loss: 1.9596 - val_accuracy: 0.7353
Epoch 4/10
469/469 [==============================] - 210s 449ms/step - loss:
1.9098 - accuracy: 0.5509 - val_loss: 1.7502 - val_accuracy: 0.7720
Epoch 5/10
469/469 [==============================] - 212s 451ms/step - loss:
1.7001 - accuracy: 0.6161 - val_loss: 1.4943 - val_accuracy: 0.7942
Epoch 6/10
469/469 [==============================] - 196s 417ms/step - loss:
1.4689 - accuracy: 0.6534 - val_loss: 1.2331 - val_accuracy: 0.8059
Epoch 7/10
  8/469 [..............................] - ETA: 3:49 - loss: 1.3557 -
accuracy: 0.6807

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```