#TEAM ID : PNT2022TMID04039

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

```
(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 [==============================] - 2s 0us/step
```

```
print(x_train.shape, y_train.shape)
```

```
(60000, 28, 28) (60000,)
```

```
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)
```

```
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

```
batch_size = 128
num_classes = 10
epochs = 10
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5,
5),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, 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'])


hist = model.fit(x_train,
y_train,batch_size=batch_size,epochs=epochs,verbose=1,validation_data=
(x_test, y_test))
print("The model has successfully trained")

Epoch 1/10
469/469 [==============================] - 13s 7ms/step - loss: 2.3033
- accuracy: 0.1040 - val_loss: 2.2908 - val_accuracy: 0.1491
Epoch 2/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2903
- accuracy: 0.1265 - val_loss: 2.2772 - val_accuracy: 0.1735
Epoch 3/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2787
- accuracy: 0.1435 - val_loss: 2.2636 - val_accuracy: 0.2168
Epoch 4/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2674
- accuracy: 0.1600 - val_loss: 2.2498 - val_accuracy: 0.2720
Epoch 5/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2545
- accuracy: 0.1805 - val_loss: 2.2349 - val_accuracy: 0.3280
Epoch 6/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2431
- accuracy: 0.1962 - val_loss: 2.2183 - val_accuracy: 0.3812
Epoch 7/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2284
- accuracy: 0.2133 - val_loss: 2.1996 - val_accuracy: 0.4277
Epoch 8/10
469/469 [==============================] - 3s 6ms/step - loss: 2.2127
- accuracy: 0.2312 - val_loss: 2.1790 - val_accuracy: 0.4667
Epoch 9/10
469/469 [==============================] - 3s 6ms/step - loss: 2.1947
- accuracy: 0.2481 - val_loss: 2.1563 - val_accuracy: 0.4926
Epoch 10/10
469/469 [==============================] - 3s 6ms/step - loss: 2.1765
- accuracy: 0.2620 - val_loss: 2.1314 - val_accuracy: 0.5165
The model has successfully trained

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

```
Test loss: 2.1314213275909424
Test accuracy: 0.5164999961853027
```