```
import tensorflow as tf
```

```
from tensorflow import keras
```

```
from keras.datasets import mnist
```

```
from keras.layers import Dense, Flatten, Conv2D, AveragePooling2D
```

```
from keras.models import Sequential
```

```
(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
```

```
x_train.shape
```

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

```
x_train.shape[1]
```

```
28
```

```
#performing Reshapping

x_train = x_train.reshape(x_train.shape[0],28,28,1)
x_test = x_test.reshape(x_test.shape[0],28,28,1)
```

```
x_train.shape
```

```
(60000, 28, 28, 1)
```

```
# Normalisation - Min. Max Scaling (0,1)

x_train = x_train / 255
x_test = x_test / 255
```

```
y_train[0]
```

```
5
```

```
# One hot encoding

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

```
y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

```
model = Sequential()
```

```
model.add(Conv2D(6, kernel_size=(5,5), padding ='valid', activation = 'tanh', input_shape=(28,28,1)))
```

```
model.add(AveragePooling2D(pool_size=(2,2), strides = 2, padding = 'valid'))
```

```
model.add(Conv2D(16, kernel_size=(5,5), padding ='valid', activation = 'tanh'))
model.add(AveragePooling2D(pool_size=(2,2), strides = 2, padding = 'valid'))
```

```
model.add(Flatten())
model.add(Dense(120, activation = 'tanh'))
model.add(Dense(84, activation = 'tanh'))
model.add(Dense(10, activation = 'softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 24, 24, 6)         156

 average_pooling2d (AverageP  (None, 12, 12, 6)        0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 8, 8, 16)          2416

 average_pooling2d_1 (Averag  (None, 4, 4, 16)         0
 ePooling2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 120)               30840

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 10)                850

=================================================================
Total params: 44,426
Trainable params: 44,426
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss=keras.metrics.categorical_crossentropy, optimizer = keras.optimizers.Adam(),metrics=['accuracy'])
```

```
model.fit(x_train,y_train, batch_size=128,epochs=10, verbose=1,validation_data=(x_test, y_test))
```

```
Epoch 1/10
469/469 [==============================] - 26s 51ms/step - loss: 0.3703 - accuracy: 0.8928 - val_loss: 0.1659 - val_accuracy: 0.9510
Epoch 2/10
469/469 [==============================] - 23s 49ms/step - loss: 0.1394 - accuracy: 0.9575 - val_loss: 0.1066 - val_accuracy: 0.9673
Epoch 3/10
469/469 [==============================] - 25s 53ms/step - loss: 0.0892 - accuracy: 0.9732 - val_loss: 0.0758 - val_accuracy: 0.9771
Epoch 4/10
469/469 [==============================] - 24s 52ms/step - loss: 0.0654 - accuracy: 0.9797 - val_loss: 0.0649 - val_accuracy: 0.9802
Epoch 5/10
469/469 [==============================] - 24s 51ms/step - loss: 0.0537 - accuracy: 0.9834 - val_loss: 0.0578 - val_accuracy: 0.9815
Epoch 6/10
469/469 [==============================] - 25s 53ms/step - loss: 0.0429 - accuracy: 0.9865 - val_loss: 0.0571 - val_accuracy: 0.9813
Epoch 7/10
469/469 [==============================] - 24s 51ms/step - loss: 0.0342 - accuracy: 0.9895 - val_loss: 0.0464 - val_accuracy: 0.9855
Epoch 8/10
469/469 [==============================] - 25s 54ms/step - loss: 0.0305 - accuracy: 0.9904 - val_loss: 0.0440 - val_accuracy: 0.9864
Epoch 9/10
469/469 [==============================] - 27s 58ms/step - loss: 0.0271 - accuracy: 0.9913 - val_loss: 0.0449 - val_accuracy: 0.9837
Epoch 10/10
469/469 [==============================] - 24s 52ms/step - loss: 0.0218 - accuracy: 0.9932 - val_loss: 0.0479 - val_accuracy: 0.9840
<keras.callbacks.History at 0x7ff54039f430>
```

```
score = model.evaluate(x_test, y_test)

print('test loss', score[0])
print('test accuracy', score[1])
```

```
313/313 [==============================] - 3s 7ms/step - loss: 0.0479 - accuracy: 0.9840
test loss 0.0478648878633976
test accuracy 0.984000027179718
```

```
x_test[9]
```

```
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]]])
```

```
predictions = model.predict(x_test)
```

```
313/313 [==============================] - 3s 8ms/step
```

```
import numpy as np
pred_image= np.argmax(predictions[10])
```

```
pred_image
```

```
0
```

✓ 0s completed at 12:50 PM