In [1]:

```
import tensorflow as tf
from keras import layers, models
from tensorflow import keras
import numpy as np
```

In [2]:

```
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://sto
rage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 48s 4us/step
```

In [3]:

```
X_train.shape
```

Out[3]:

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

In [4]:

```
X_test.shape
```

Out[4]:

```
(10000, 28, 28)
```

In [5]:

```
X_train[3]
```

Out[5]:

```
array([[  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,   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,   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,   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, 124, 253, 255,  63,   0,   0,   0,   0,
          0,   0],
```

In [6]:

```
import pandas as pd
```

In [7]:

```
pd.DataFrame(X_train[0])
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	18	19	20	21	22	23	24	25	26	27
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	175	26	166	255	247	127	0	0	0	0
6	0	0	0	0	0	0	0	0	30	36	...	225	172	253	242	195	64	0	0	0	0
7	0	0	0	0	0	0	0	49	238	253	...	93	82	82	56	39	0	0	0	0	0
8	0	0	0	0	0	0	0	18	219	253	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	80	156	...	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	14	...	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	...	25	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	...	150	27	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	...	253	187	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	...	253	249	64	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	...	253	207	2	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	...	250	182	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	...	78	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	23	66	...	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	18	171	219	253	...	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	55	172	226	253	253	253	...	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	136	253	253	253	212	135	...	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

28 rows × 28 columns

In [8]:

```
X_train = X_train / 255
X_test = X_test / 255
```

In [9]:

```
pd.DataFrame(X_train[0])
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	18	19	20	21	
0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
1	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
2	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
3	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
4	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
5	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.686275	0.101961	0.650980	1.000000	0.96
6	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.117647	0.141176	...	0.882353	0.674510	0.992157	0.949020	0.76
7	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.192157	0.933333	0.992157	...	0.364706	0.321569	0.321569	0.219608	0.15
8	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.070588	0.858824	0.992157	...	0.000000	0.000000	0.000000	0.000000	0.00
9	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.313725	0.611765	...	0.000000	0.000000	0.000000	0.000000	0.00
10	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.054902	...	0.000000	0.000000	0.000000	0.000000	0.00
11	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
12	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
13	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
14	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.098039	0.000000	0.000000	0.000000	0.00
15	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.588235	0.105882	0.000000	0.000000	0.00
16	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.992157	0.733333	0.000000	0.000000	0.00
17	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.992157	0.976471	0.250980	0.000000	0.00
18	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.992157	0.811765	0.007843	0.000000	0.00
19	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.980392	0.713725	0.000000	0.000000	0.00
20	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.305882	0.000000	0.000000	0.000000	0.00
21	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.090196	0.258824	...	0.000000	0.000000	0.000000	0.000000	0.00
22	0.0	0.0	0.0	0.0	0.000000	0.000000	0.070588	0.670588	0.858824	0.992157	...	0.000000	0.000000	0.000000	0.000000	0.00
23	0.0	0.0	0.0	0.0	0.215686	0.674510	0.886275	0.992157	0.992157	0.992157	...	0.000000	0.000000	0.000000	0.000000	0.00
24	0.0	0.0	0.0	0.0	0.533333	0.992157	0.992157	0.992157	0.831373	0.529412	...	0.000000	0.000000	0.000000	0.000000	0.00
25	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
26	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
27	0.0	0.0	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00

28 rows × 28 columns

In [10]:

```
X_train = X_train.reshape(-1,28,28,1) #training set
```

In [11]:

```
X_train.shape
```

Out[11]:

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

In [12]:

```
X_test = X_test.reshape(-1,28,28,1) #test set
```

In [13]:

```
X_test.shape
```

Out[13]:

```
(10000, 28, 28, 1)
```

In [14]:

```
convolutional_neural_network = models.Sequential([
    layers.Conv2D(filters=25, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

In [15]:

```
convolutional_neural_network.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accurac
```

In [16]:

```
convolutional_neural_network.fit(X_train, y_train, epochs  = 10)
```

```
Epoch 1/10
1875/1875 [==============================] - 41s 20ms/step - loss: 0.2255 - accuracy: 0.9316
Epoch 2/10
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0758 - accuracy: 0.9762
Epoch 3/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0549 - accuracy: 0.9831
Epoch 4/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0431 - accuracy: 0.9866
Epoch 5/10
1875/1875 [==============================] - 37s 20ms/step - loss: 0.0348 - accuracy: 0.9897
Epoch 6/10
1875/1875 [==============================] - 36s 19ms/step - loss: 0.0274 - accuracy: 0.9911
Epoch 7/10
1875/1875 [==============================] - 38s 20ms/step - loss: 0.0224 - accuracy: 0.9931
Epoch 8/10
1875/1875 [==============================] - 36s 19ms/step - loss: 0.0197 - accuracy: 0.9937
Epoch 9/10
1875/1875 [==============================] - 35s 19ms/step - loss: 0.0164 - accuracy: 0.9947
Epoch 10/10
1875/1875 [==============================] - 35s 18ms/step - loss: 0.0134 - accuracy: 0.9955
```

Out[16]:

```
<keras.callbacks.History at 0x22a15363af0>
```

In [17]:

```
convolutional_neural_network.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 4s 8ms/step - loss: 0.0501 - accuracy: 0.9888
```

Out[17]:

```
[0.050106536597013474, 0.9887999892234802]
```

In [18]:

```
y_predicted_by_model = convolutional_neural_network.predict(X_test)
```

```
313/313 [==============================] - 2s 7ms/step
```

In [19]:

```
np.argmax(y_predicted_by_model[0])
```

Out[19]:

```
7
```

In [ ]:

```

```