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

```
(xtrain,ytrain) , (xtest,ytest) = keras.datasets.mnist.load_data()
```

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

Scaling the train and test dataset

```
xtrain= xtrain/255
```

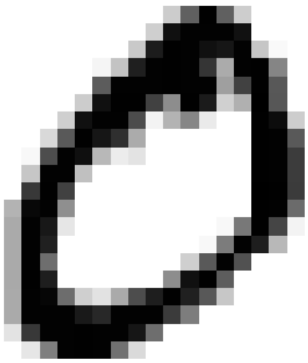
```
xtrain.shape
```

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

```
# converting 2D xtrain to 1D xtrain
xtrain = xtrain.reshape(-1,28,28,1)
xtrain.shape
```

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

```
import matplotlib as mpl
import matplotlib.pyplot as plt
some_digit = xtrain[1]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap = mpl.cm.binary, interpolation="nearest")
plt.axis("off")
plt.show()
```



```
ytrain[0]
```

```
5
```

```
xtest = xtest/255
```

```
xtest.shape
```

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

```
# conveting 2D xtest to 1D xtest
xtest = xtest.reshape(-1,28,28,1)
xtest.shape
```

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

Training Convulutional network

```
cnn = models.Sequential([

    layers.Conv2D( filters=25, kernel_size=(3,3) ,activation='relu',input_shape=(28,28,1)),
    layers.MaxPool2D((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')
])
```

```
cnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy' ,metrics= ['accuracy'])
cnn.fit(xtrain,ytrain,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 18s 4ms/step - loss: 0.2160 - accuracy: 0.9331
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0752 - accuracy: 0.9762
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0544 - accuracy: 0.9836
Epoch 4/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0414 - accuracy: 0.9873
Epoch 5/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0336 - accuracy: 0.9896
Epoch 6/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0274 - accuracy: 0.9911
Epoch 7/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0231 - accuracy: 0.9926
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0177 - accuracy: 0.9942
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0162 - accuracy: 0.9950
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0136 - accuracy: 0.9953
<keras.callbacks.History at 0x7f299d4f5c60>
```

Evaluting the model

```
cnn.evaluate(xtest,ytest)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0566 - accuracy: 0.9867
[0.05658264830708504, 0.9866999983787537]
```

```
y_predicted = cnn.predict(xtest)
y_predicted[0]
```

```
313/313 [==============================] - 1s 2ms/step
array([9.1959514e-08, 1.4398065e-06, 8.7110129e-06, 4.2598340e-06,
       3.5262445e-07, 9.5149881e-07, 2.4636893e-10, 9.9997354e-01,
       8.0676255e-06, 2.5791312e-06], dtype=float32)
```

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

```
7
```

```
y_labels = [np.argmax(i) for i in y_predicted]
```

```
y_labels[:5]
```

```
[7, 2, 1, 0, 4]
```