```
from tensorflow import keras
```

```
(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
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
print(x_train.shape)
print(x_test.shape)
```

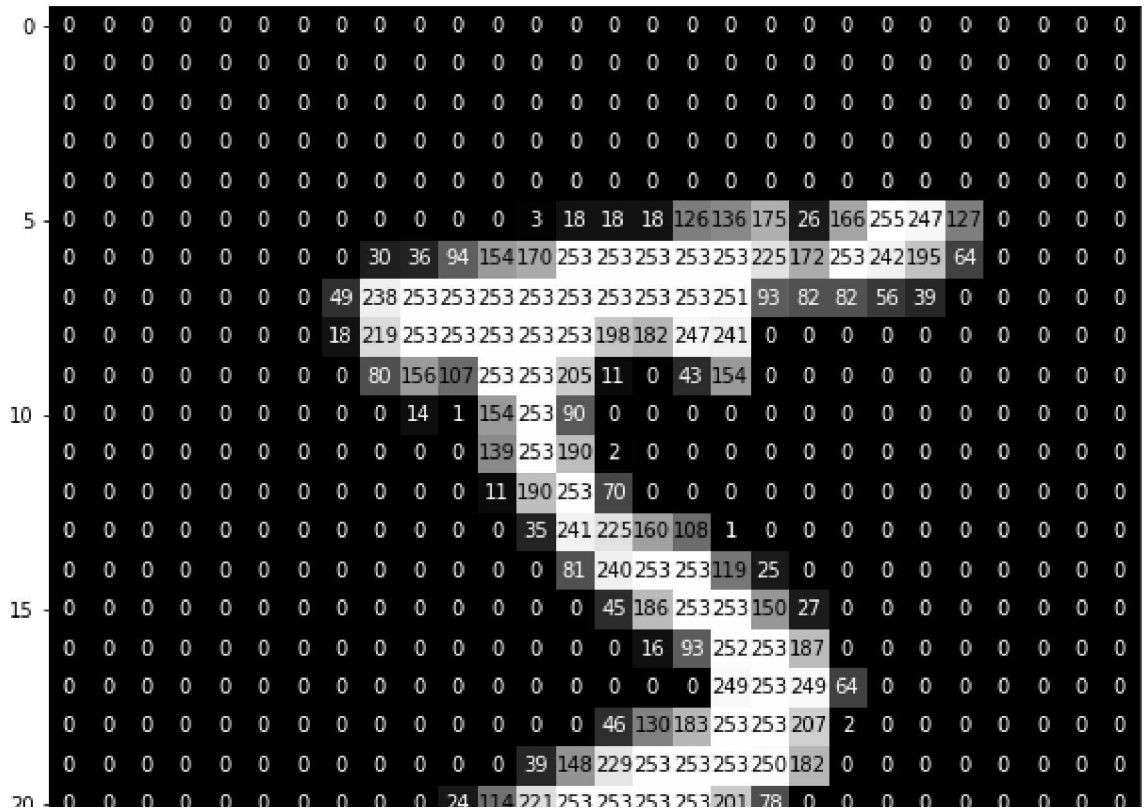
```
(60000, 28, 28)
(10000, 28, 28)
```

```
print(y_train.shape)
print(y_test.shape)
```

```
(60000,)
(10000,)
```

```
def visualize_input(img, ax):
    ax.imshow(img, cmap='gray')
    width, height = img.shape
    thresh = img.max()/2.5
    for x in range(width):
        for y in range(height):
            ax.annotate(str(round(img[x][y],2)), xy=(y,x),
                        horizontalalignment='center',
                        verticalalignment='center',
                        color='white' if img[x][y]<thresh else 'black')
```

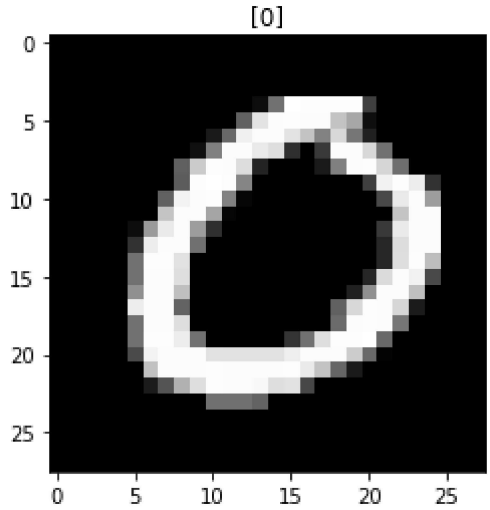
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111)
visualize_input(x_train[0],ax )
plt.show()
```



```
print(y_train)
print(y_test)
```

```
[5 0 4 ... 5 6 8]
[7 2 1 ... 4 5 6]
```

```
# lets see some random images and its labels
import random
import matplotlib.pyplot as plt
i = random.randint(0,60000)
plt.imshow(x_train[i],cmap='gray') # Color map
plt.title([y_train[i]])
plt.show()
```



```
# How many images are there in every digit?
import numpy as np
np.unique(y_train,return_counts=True)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8),
 array([5923, 6742, 5958, 6131, 5842, 5421, 5918, 6265, 5851, 5949]))
```

```
np.unique(y_test,return_counts=True)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8),
 array([ 980, 1135, 1032, 1010,  982,  892,  958, 1028,  974, 1009]))
```

```
# Normalization : Scaling down the value to a specific range(0-1)
x_train=x_train/255
x_test = x_test/255
```

```
# AFter Normalization
print(x_train.max())
print(x_train.min())
```

```
1.0
0.0
```

```
from keras.layers import Dense
from keras.layers import Flatten
model = keras.models.Sequential()
model.add(Flatten(input_shape=(28,28),)) # 784
model.add(Dense(392,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

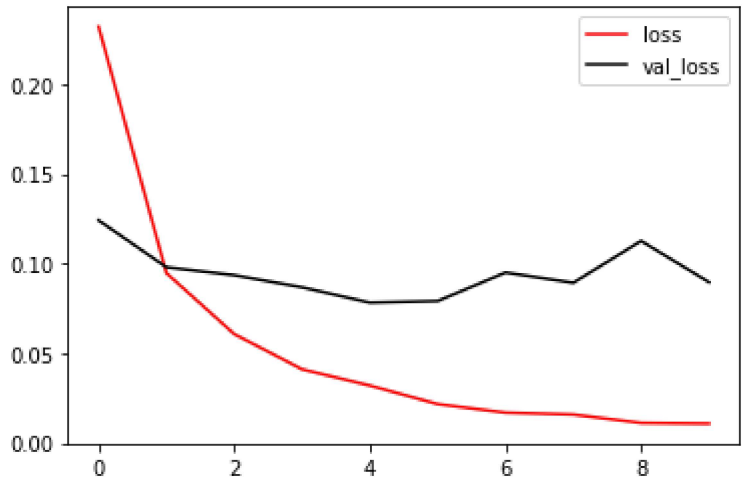
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(x_train,y_train,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
1500/1500 [==============================] - 7s 3ms/step - loss: 0.2319 - accuracy: 0.93
Epoch 2/10
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0947 - accuracy: 0.97
Epoch 3/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0607 - accuracy: 0.98
Epoch 4/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0411 - accuracy: 0.98
Epoch 5/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0320 - accuracy: 0.98
Epoch 6/10
```

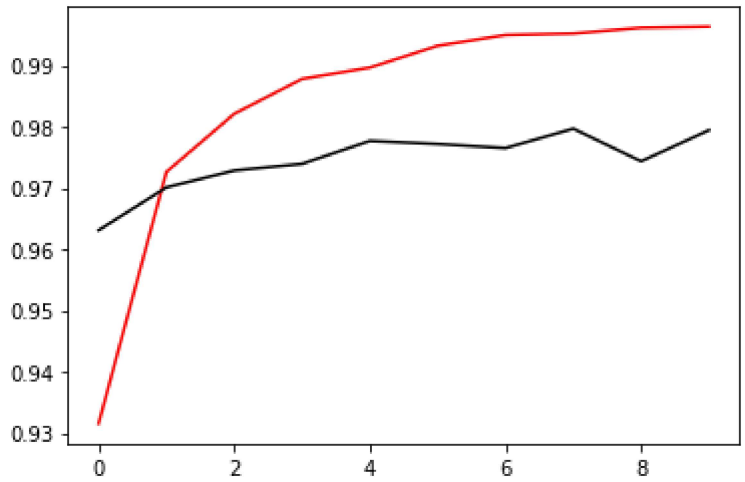
```
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0217 - accuracy: 0.99
Epoch 7/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0169 - accuracy: 0.99
Epoch 8/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0159 - accuracy: 0.99
Epoch 9/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0112 - accuracy: 0.99
Epoch 10/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0108 - accuracy: 0.99
```

```
import matplotlib.pyplot as plt
plt.plot((history.history['loss']),color='red',label='loss')
plt.plot((history.history['val_loss']),color='black',label='val_loss')
plt.legend()
plt.show()
```




```
plt.plot(history.history['accuracy'],label='accuracy',color='red')
plt.plot(history.history['val_accuracy'],label='val_accuracy',color='black')
```

```
[<matplotlib.lines.Line2D at 0x7fa18244f6d0>]
```



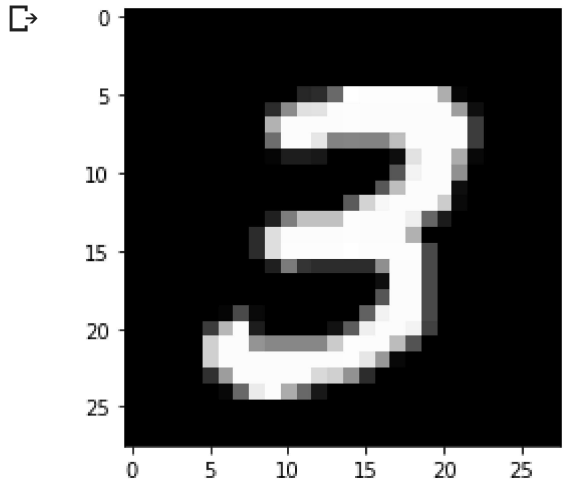

```
# Evaluate on test data
```

```
y_pred = model.predict(x_test)
y_pred = np.argmax(y_pred,axis=1)
y_pred
```

```
array([7, 2, 1, ..., 4, 5, 6])
```

```
from keras.preprocessing import image
```

```
img = image.load_img(path="/content/MNIST_digit.png", color_mode= 'grayscale', target_size=(2
img = image.img_to_array(img)
plt.imshow(image.array_to_img(img), cmap="gray")
img = img.astype('float')/255
test_img = img.reshape((1, 28, 28, 1))
#img_class = model.predict_classes(test_img)
img_class = np.argmax(model.predict(test_img), axis = 1)
prediction = img_class[0]
```




```
prediction
```

```
3
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
accuracy_score(y_pred,y_test)
```

```
0.9803
```

```
confusion_matrix(y_pred,y_test)
```

```
array([[ 970,    0,    2,    1,    2,    1,    1,    2,    3,    2],
       [   1, 1122,    0,    0,    0,    0,    3,    0,    0,    2],
       [   1,    2, 1015,    1,    1,    0,    0,    9,    3,    0],
       [   1,    4,    2,  995,    1,    8,    1,    5,    5,    5],
```

```
[   1,    0,    1,    0,  947,    0,    1,    0,    0,    6],
[   1,    0,    1,    3,    0,  873,    3,    0,    6,    7],
[   3,    2,    2,    0,   14,    5,  948,    0,    2,    0],
[   0,    3,    6,    2,    2,    0,    0, 1005,    3,    6],
[   1,    2,    3,    3,    3,    2,    1,    5,  949,    2],
[   1,    0,    0,    5,   12,    3,    0,    2,    3,  979]])
```

```
model.save("mnist.hdf5")
```