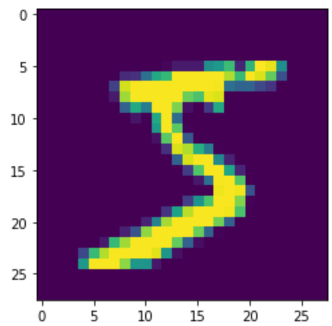
```
# Import packages and set numpy random seed
import numpy as np
np.random.seed(7)
import tensorflow as tf
mnist = tf.keras.datasets.mnist

from keras.utils import np_utils
```

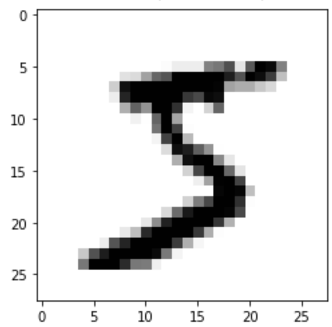
```
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

```
# display image
import matplotlib.pyplot as plt
plt.imshow(X_train[0])
plt.show()
plt.imshow(X_train[0], cmap=plt.cm.binary)
```



```
<matplotlib.image.AxesImage at 0x7f2955a4c750>
```



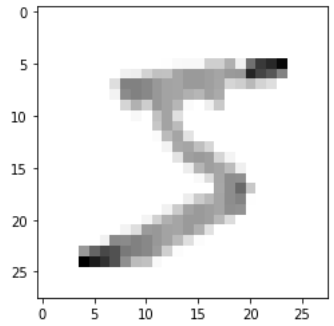
```
#The image is represented into a matrix
print(X_train[0])
```

```
[[  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   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
```

```
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
  25   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
 150  27   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
 253 187   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
 253 249  64   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
 253 207   2   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
 250 182   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
  78   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

```
#normalize the image
X_train = tf.keras.utils.normalize(X_train,axis=1)
X_test = tf.keras.utils.normalize(X_test,axis=1)
plt.imshow(X_train[0], cmap=plt.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x7f29559cd5d0>
```




```
##The image is represented into a matrix after normalize
print(X_train[0])
```

```
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.05298497 0.42752138 0.4219755  0.45852825 0.43408872 0.37314701
 0.33153488 0.25273681 0.11646967 0.01312603 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.37491383 0.56222061
 0.66525569 0.63253163 0.48748768 0.45852825 0.43408872 0.359873
 0.17428513 0.01425695 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.92705966 0.82698729
 0.74473314 0.63253163 0.4084877  0.24466922 0.22648107 0.02359823
 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.        ]]
```

```
print(y_train[0])
```

```
5
```

```
# Resize the image into 1D
img_size = 28
x_train = np.array(X_train).reshape(-1,img_size,img_size,1)
x_test = np.array(X_test).reshape(-1,img_size,img_size,1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
```

```
6# create model
model = Sequential()
# first conv layer
model.add(Conv2D(64, (3, 3), input_shape=x_train.shape[1:], activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#second conv layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#third conv layer
model.add(Conv2D(64,(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

#model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))

model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)
```

```
 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 10)                330

=================================================================
Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0
_________________________________________________________________
```

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

```
print(len(x_train))
print(len(y_train))
```

```
60000
60000
```

```
# Fit the model
model.fit(x_train, y_train, epochs=5, validation_split = 0.3, batch_size=1)
```

```
Epoch 1/5
42000/42000 [==============================] - 234s 6ms/step - loss: 0.2390 - accuracy: 0.9268 - val_loss: 0.1213 - val_accuracy: 0
Epoch 2/5
42000/42000 [==============================] - 244s 6ms/step - loss: 0.1174 - accuracy: 0.9678 - val_loss: 0.1284 - val_accuracy: 0
Epoch 3/5
42000/42000 [==============================] - 247s 6ms/step - loss: 0.1071 - accuracy: 0.9721 - val_loss: 0.1190 - val_accuracy: 0
Epoch 4/5
42000/42000 [==============================] - 245s 6ms/step - loss: 0.1073 - accuracy: 0.9743 - val_loss: 0.0992 - val_accuracy: 0
Epoch 5/5
42000/42000 [==============================] - 252s 6ms/step - loss: 0.1004 - accuracy: 0.9750 - val_loss: 0.1066 - val_accuracy: 0
<keras.callbacks.History at 0x7f2951c0cb90>
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, batch_size=1)
print("test loss on 10000 test samples", test_loss)
print("validation accuracy", test_acc)
```

```
10000/10000 [==============================] - 21s 2ms/step - loss: 0.1171 - accuracy: 0.9712
test loss on 10000 test samples 0.11708095669746399
validation accuracy 0.9711999893188477
```

```
predic = model.predict([x_test])
print(predic)
```

```
[[1.7640687e-38 3.8629141e-29 3.8235444e-17 ... 1.0000000e+00
  3.5550427e-24 6.9909089e-15]
 [1.7735304e-21 5.5612005e-20 1.0000000e+00 ... 6.8582212e-10
  2.0438761e-14 3.7846590e-22]
 [3.8005124e-16 1.0000000e+00 2.2976892e-14 ... 2.4129138e-14
  5.6753829e-10 2.2403523e-11]
 ...
 [3.8266545e-31 2.2397079e-14 2.7200797e-20 ... 3.9611259e-14
  4.5218351e-13 4.8122090e-10]
 [2.2751331e-19 1.3038688e-21 2.4242223e-20 ... 8.3450585e-23
  6.0527749e-10 9.7729069e-14]
 [2.1919782e-09 1.2884725e-12 2.8577558e-16 ... 4.3964496e-21
  6.6421477e-09 8.1658560e-12]]
```

```
print(np.argmax(predic[0]))
```

```
7
```

```
plt.imshow(X_test[0])
```

```
<matplotlib.image.AxesImage at 0x7f294e38f850>
```

```
print(np.argmax(predic[128]))
```

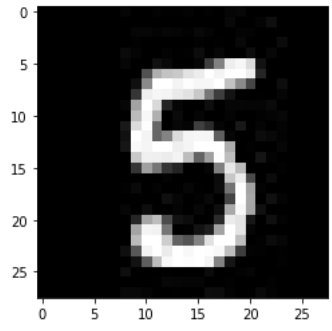
```
8
```

```
from google.colab import files
from keras.preprocessing import image
import matplotlib.image as mpimg
import cv2
%matplotlib inline
print("Masukkan gambar")
uploaded = files.upload()

for fn in uploaded.keys():

# predicting images
  path = fn
  img = cv2.imread(path)
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  resized = cv2.resize(gray,(28,28), interpolation=cv2.INTER_AREA)
  newimg = tf.keras.utils.normalize(resized,axis=1)
  newimg = np.array(newimg).reshape(-1,img_size,img_size,1)
  imgplot = plt.imshow(img)
  plt.show()
  #x = image.img_to_array(img)
  #x = np.expand_dims(x, axis=0)

  #images = np.vstack([x])
  classes = model.predict(newimg)
  output = np.argmax(classes)
  # print(fn)
  if output==0:
    print('Hasil Pengenalan : Angka 0')
  elif output==1:
    print('Hasil Pengenalan : Angka 1')
  elif output==2:
    print('Hasil Pengenalan : Angka 2')
  elif output==3:
    print('Hasil Pengenalan : Angka 3')
  elif output==4:
    print('Hasil Pengenalan : Angka 4')
  elif output==5:
    print('Hasil Pengenalan : Angka 5')
  elif output==6:
    print('Hasil Pengenalan : Angka 6')
  elif output==7:
    print('Hasil Pengenalan : Angka 7')
  elif output==8:
    print('Hasil Pengenalan : Angka 8')
  else:
    print('Hasil Pengenalan : Angka 9')
```

```
Masukkan gambar
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving img_11.jpg to img_11.jpg
```



```
Hasil Pengenalan : Angka 5
```