Listing 2.1 Loading the MNIST dataset in Keras

```
#Taking mnist dataset from tensorflow.keras
from tensorflow.keras.datasets import mnist

#loading dataset of mnist having train images with labels and testing images with labels.
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

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

```
#print number number of images and labels available in training dataset
print(len(train_images))
print(len(train_labels))
```

```
60000
60000
```

```
#print number number of images and labels available in testing dataset
print(len(test_images))
print(len(test_labels))
```

```
10000
10000
```

```
#presenting test and train dataset shape.
print(test_images.shape)
train_images.shape
```

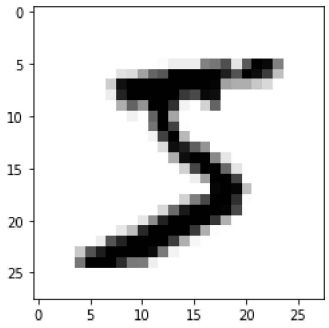
```
(10000, 28, 28)
(60000, 28, 28)
```

```
#presenting image and its label of training dataset at index 0 i.e. first image of training dataset
import matplotlib.pyplot as plt

digit = train_images[0]
print("Class Label:",train_labels[0])
#digit.shape

plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```

```
Class Label: 5
```




```
#presenting·image·and·its·label·of·testing·dataset·at·index·0·i.e.·first·image·of·testing·dataset
digit·=·test_images[0]
print("Class·Label:",test_labels[0])
#digit.shape

plt.imshow(digit,·cmap=plt.cm.binary)
plt.show()
```

```
Class Label: 7
```

```
#Printing all the training labels, number of labels and their unique values present in training dataset
import numpy as np
print(train_labels)
print(len(train_labels))
print(np.unique(train_labels))
```

```
[5 0 4 ... 5 6 8]
60000
[0 1 2 3 4 5 6 7 8 9]
```

```
#Printing all the testing labels, number of labels and their unique values present in testing dataset
import numpy as np
print(test_labels)
print(len(test_labels))
print(np.unique(test_labels))
```

```
[7 2 1 ... 4 5 6]
10000
[0 1 2 3 4 5 6 7 8 9]
```

```
import numpy as np
#The randint() method returns an integer number selected element from the specified range.
b=np.random.randint(0,255,size=(28,28))
print(b.shape)
#The ravel() function is used to create a contiguous flattened array. A 1-D array, containing the elements of the input, is returned.
b1=b.ravel()
print(b1.shape)
```

```
(28, 28)
(784,)
```

Listing 2.2 The network architecture

```
from tensorflow.keras import models
from tensorflow.keras import layers

network = models.Sequential()
#network=model.add(layers.Dense(output,activation_function ,input_shape))
#network=model.add(layers.Dense(512,activation='relu' ,input_shape=(784)))
#just set your last layer activation function with page 114 of Deep learning with python
#This problem is related to Multiclass, sigle label classification(Softmax, categorical_crossentropy)
#Output shound be set with the power of 2

network.add(layers.Dense(512, activation='relu', input_shape=(28*28, )))
network.add(layers.Dense(10, activation='softmax'))
```

Listing 2.3 The compilation step

```
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
#optimizer=rmsprop
#metrics=['f1','precision','accuracy']
#los function=see page 114 of Deep learning with python


# 3 dimensional image image samples,width, heigth
train_images.shape
```

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

```
#The ndim is an attribute in the pandas DataFrame which is used to get the integer/number representation of dimensions of the given DataFrame
train_images.ndim
```

```
3
```

```
#normilization since maximum intensity in gray scale is 255
50/255
```

```
0.19607843137254902
```

Listing 2.4 Preparing the image data

```
train_images = train_images.reshape((60000, 28 * 28)) #changing into 1D by 28*28
train_images = train_images.astype('float32') / 255 #Normalization


test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255 #Normalization
```

Listing 2.5 Preparing the labels

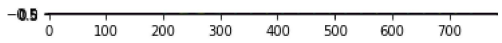
```
train_labels[0]
```

```
5
```

```
#Coverting 2D(28,28) into 1D(784) and presenting image and its label of testing dataset at index 0 i.e. first image of testing dataset
digit = test_images[0]
print("Class Label:",test_labels[0])
#digit.shape

plt.imshow([digit])
plt.show()
```

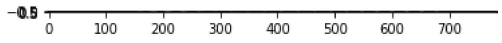
```
Class Label: 7
```




```
#Coverting 2D(28,28) into 1D(784) and presenting image and its label of training dataset at index 0 i.e. first image of training dataset
digit = train_images[0]
print("Class Label:",train_labels[0])
#digit.shape

plt.imshow([digit])
plt.show()
```

```
Class Label: 5
```




```
print(train_images.shape)
print(test_images.shape)
```

```
(60000, 784)
(10000, 784)
```

One-Hot-Encoding

```
from tensorflow.keras.utils import to_categorical

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


train_labels[0]
```

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

Now Train your Model

```
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
Epoch 1/5
469/469 [==============================] - 5s 10ms/step - loss: 0.2577 - accuracy: 0.9250
Epoch 2/5
469/469 [==============================] - 5s 10ms/step - loss: 0.1036 - accuracy: 0.9690
Epoch 3/5
469/469 [==============================] - 5s 10ms/step - loss: 0.0681 - accuracy: 0.9797
Epoch 4/5
469/469 [==============================] - 5s 10ms/step - loss: 0.0494 - accuracy: 0.9855
Epoch 5/5
469/469 [==============================] - 6s 13ms/step - loss: 0.0374 - accuracy: 0.9882
<keras.callbacks.History at 0x7f4f5fc3b580>
```

Now check accuracy of Testing dataset

```
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0614 - accuracy: 0.9812
test_acc: 0.9811999797821045
```

✓ 1s completed at 5:38 PM