Assignment-02-Deep Learning (1)

November 7, 2023

1 Title of Assignment-2:

```
Implementing Feedforward neural networks with Keras and TensorFlow
a. Import the necessary packages
b. Load the training and testing data (MNIST)
c. Define the network architecture using Keras
d. Train the model using SGD
e. Evaluate the network
f. Plot the training loss and accuracy
```

2 Importing libraries

[1]:
```
#importing necessary libraries
import tensorflow as tf
from tensorflow import keras
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

3 Loading and preparing the data

MNIST stands for "Modified National Institute of Standards and Technology". It is a dataset of 70,000 handwritten images. Each image is of 28x28 pixels i.e. about 784 features. Each feature represents only one pixel's intensity i.e. from 0(white) to 255(black). This database is further divided into 60,000 training and 10,000 testing images.

[3]:
```
#import dataset and split into train and test data
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

[4]:
```
#to see length of training dataset
len(x_train)
```

[4]: 60000

[5]:
```
##to see length of testing dataset
len(x_test)
```

[5]: 10000

[6]:
```
#shape of training dataset  60,000 images having 28*28 size
x_train.shape
```

[6]: (60000, 28, 28)

[7]:
```
#shape of testing dataset  10,000 images having 28*28 size
x_test.shape
```

[7]: (10000, 28, 28)

[12]:
```
x_train[0]
```

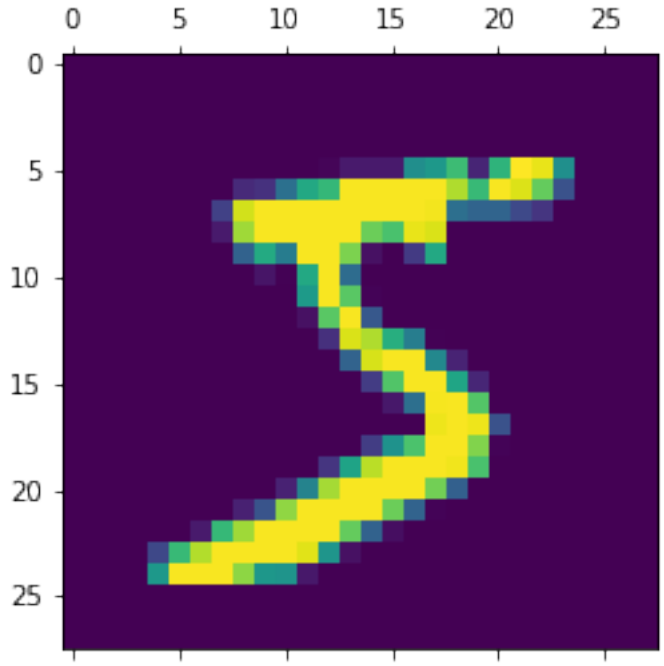
[12]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   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,   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,  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]], dtype=uint8)
```

[8]:
```
#to see how first image look
plt.matshow(x_train[0])
```

[8]:
```
<matplotlib.image.AxesImage at 0x29b1cccc130>
```



[9]:
```
#normalize the images by scaling pixel intensities to the range 0,1

x_train = x_train / 255
x_test = x_test / 255
```

[10]:
```
x_train[0]
```

[10]:
```
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.        , 0.01176471, 0.07058824, 0.07058824,
 0.07058824, 0.49411765, 0.53333333, 0.68627451, 0.10196078,
 0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.11764706, 0.14117647,
 0.36862745, 0.60392157, 0.66666667, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.88235294, 0.6745098 ,
 0.99215686, 0.94901961, 0.76470588, 0.25098039, 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.19215686, 0.93333333, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.98431373, 0.36470588, 0.32156863,
 0.32156863, 0.21960784, 0.15294118, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.07058824, 0.85882353, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.77647059,
```

```
 0.71372549, 0.96862745, 0.94509804, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31372549, 0.61176471,
 0.41960784, 0.99215686, 0.99215686, 0.80392157, 0.04313725,
 0.        , 0.16862745, 0.60392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.05490196,
 0.00392157, 0.60392157, 0.99215686, 0.35294118, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.54509804, 0.99215686, 0.74509804, 0.00784314,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.04313725, 0.74509804, 0.99215686, 0.2745098 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.1372549 , 0.94509804, 0.88235294,
 0.62745098, 0.42352941, 0.00392157, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.31764706, 0.94117647,
 0.99215686, 0.99215686, 0.46666667, 0.09803922, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.17647059,
 0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.0627451 , 0.36470588, 0.98823529, 0.99215686, 0.73333333,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.97647059, 0.99215686, 0.97647059,
 0.25098039, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.18039216,
 0.50980392, 0.71764706, 0.99215686, 0.99215686, 0.81176471,
 0.00784314, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.15294118, 0.58039216, 0.89803922,
 0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.09411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,
 0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.09019608, 0.25882353,
 0.83529412, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
 0.77647059, 0.31764706, 0.00784314, 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.07058824, 0.67058824, 0.85882353, 0.99215686,
 0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,
 0.03529412, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.21568627,
 0.6745098 , 0.88627451, 0.99215686, 0.99215686, 0.99215686,
 0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.53333333,
```

```
         0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,
         0.51764706, 0.0627451 , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ],
        [0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        , 0.        , 0.        ,
         0.        , 0.        , 0.        ]])
```

[ ]:
```
#Define the network architecture using Keras
```

4 Creating the model

The ReLU function is one of the most popular activation functions. It stands for "rectified linear unit". Mathematically this function is defined as: y = max(0,x)The ReLU function returns "0" if the input is negative and is linear if the input is positive.

The softmax function is another activation function. It changes input values into values that reach from 0 to 1.

[11]:
```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

[12]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0
```

```
 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

5 Compile the model

[13]:
```
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

6 Train the model

[14]:
```
history=model.fit(x_train, y_train,validation_data=(x_test,y_test),epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 12s 4ms/step - loss: 0.6387 -
accuracy: 0.8416 - val_loss: 0.3590 - val_accuracy: 0.9026
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3385 -
accuracy: 0.9053 - val_loss: 0.2948 - val_accuracy: 0.9180
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2921 -
accuracy: 0.9172 - val_loss: 0.2657 - val_accuracy: 0.9257
Epoch 4/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2625 -
accuracy: 0.9252 - val_loss: 0.2436 - val_accuracy: 0.9330
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2393 -
accuracy: 0.9330 - val_loss: 0.2254 - val_accuracy: 0.9366
Epoch 6/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2205 -
accuracy: 0.9377 - val_loss: 0.2107 - val_accuracy: 0.9400
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2042 -
accuracy: 0.9424 - val_loss: 0.1938 - val_accuracy: 0.9437
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1900 -
accuracy: 0.9466 - val_loss: 0.1837 - val_accuracy: 0.9466
Epoch 9/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1777 -
```

```
accuracy: 0.9500 - val_loss: 0.1717 - val_accuracy: 0.9497
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1670 -
accuracy: 0.9529 - val_loss: 0.1627 - val_accuracy: 0.9518
```

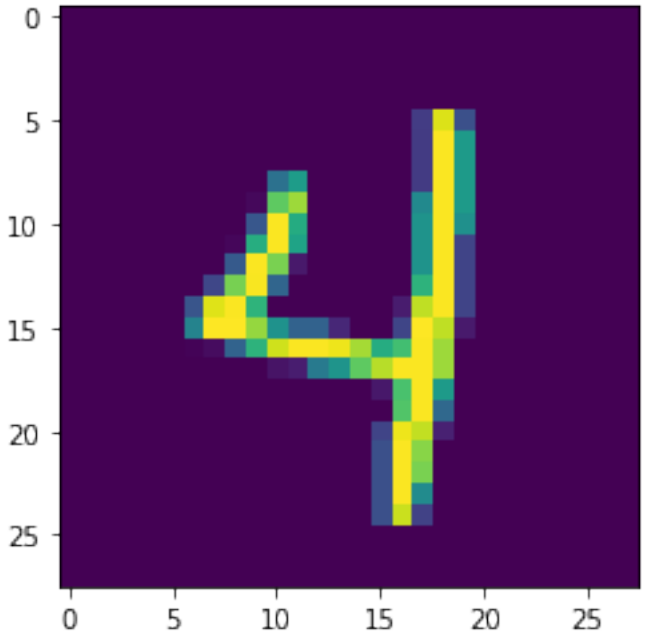
7 Evaluate the model

[15]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1627 -
accuracy: 0.9518
Loss=0.163
Accuracy=0.952
```

8 Making Prediction on New Data

[18]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



[19]:
```
#we use predict() on new data
predicted_value=model.predict(x_test)
print("Handwritten number in the image is= %d" %np.argmax(predicted_value[n]))
```

```
313/313 [==============================] - 1s 2ms/step
Handwritten number in the image is= 4
```

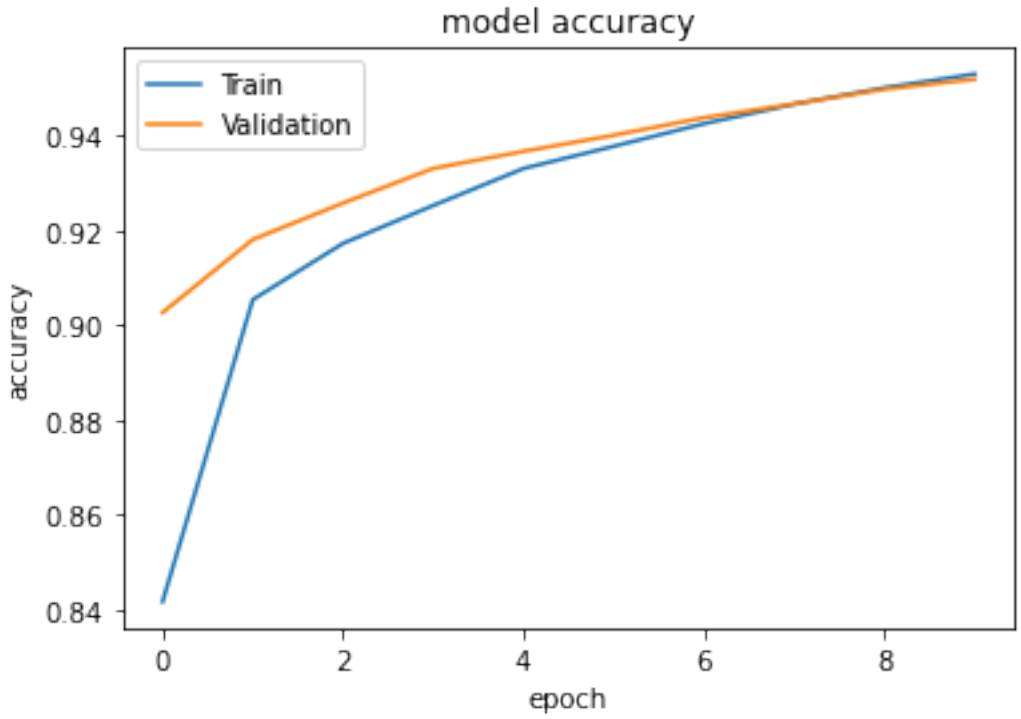
9 Plot graph for Accuracy and Loss

[20]:
```
history.history??
```

[21]:
```
history.history.keys()
```

[21]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

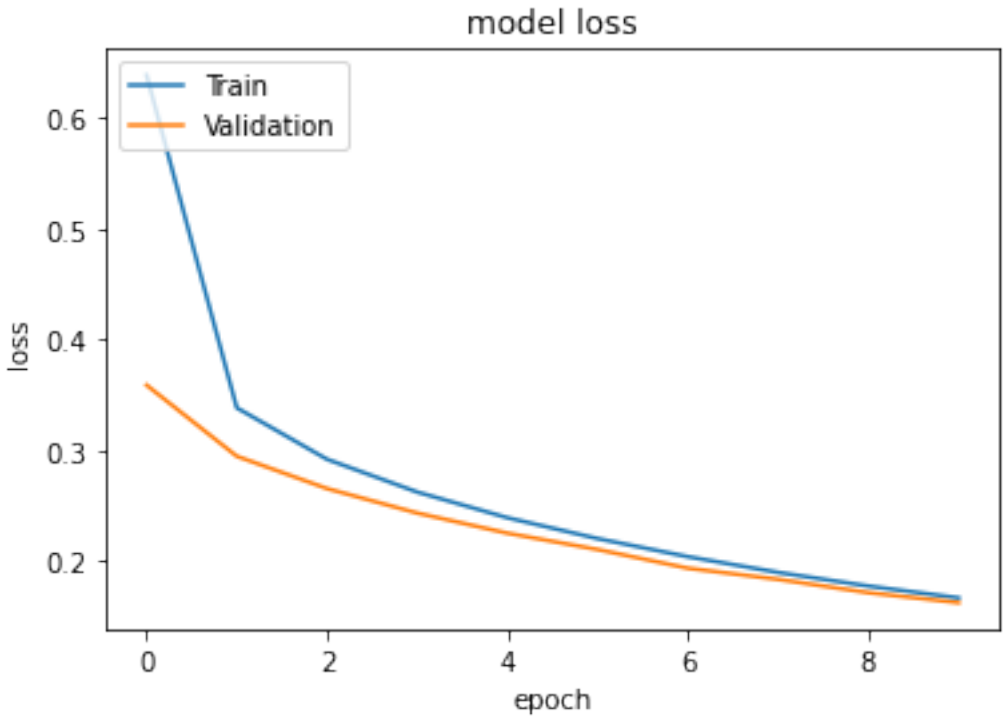
[22]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



graph representing the model's accuracy

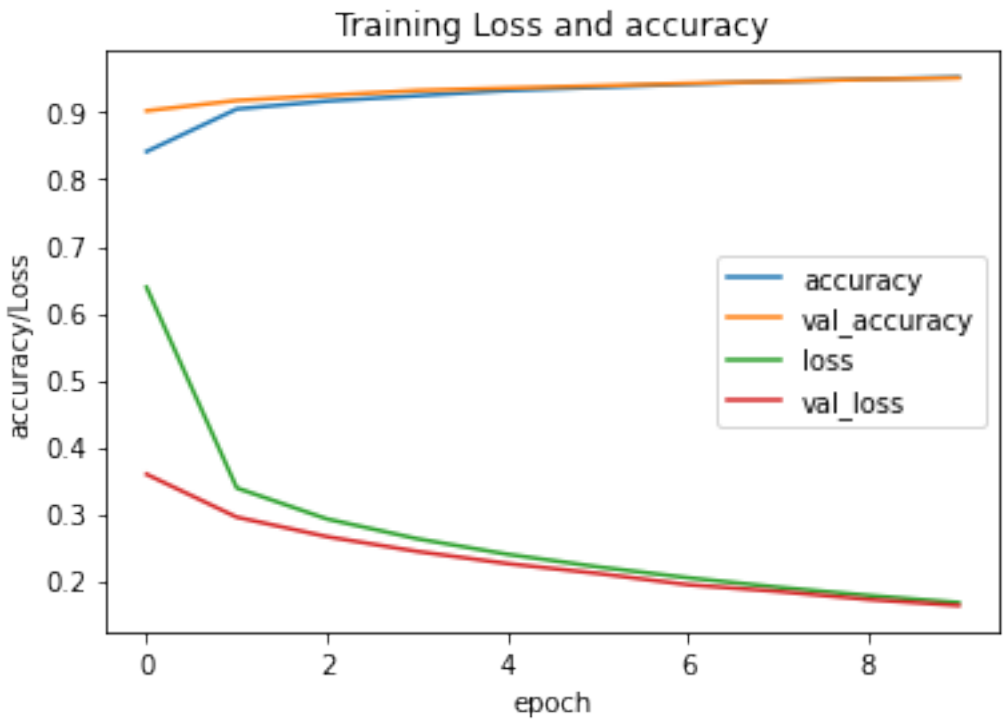
[23]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



graph represents the model's loss

[24]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss and accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy','loss','val_loss'])
plt.show()
```



Conclusion: With above code We can see, that throughout the epochs, our model accuracy increases and our model loss decreases,that is good since our model gains confidence with its predictions.

1. The two losses (loss and val_loss) are decreasing and the accuracy (accuracy and val_accuracy)are increasing.
   So this indicates the model is trained in a good way.

2. The val_accuracy is the measure of how good the predictions of your model are. So In this case, it looks like the model is well trained after 10 epochs

[50]:
```
#pwd
```

[50]:
```
'C:\\Users\\admin'
```

10 Save the model

[25]:
```
keras_model_path='C:\\Users\\admin'
model.save(keras_model_path)
```

```
INFO:tensorflow:Assets written to: C:\Users\admin\assets
```

[26]:
```
#use the save model
restored_keras_model = tf.keras.models.load_model(keras_model_path)
```

[ ]: