HANDWRITTEN DIGIT CLASSIFICATION (MNIST)

Project Goal:

Train a machine learning model to recognize handwritten digits from images.

Task : Image Classification

Objective :

Develop a model that can accurately classify handwritten digits into their corresponding categories (0-9).

About the dataset :

The MNIST(Mixed National Institute of Standards and Technology) dataset is collection of :

-- 70,000 images of handwriiten digits(0-9)
-- 28 x 28 pixels,grayscale
-- 60,000 training images
-- 10,000 testing images

This dataset is available in many python libraries include TensorFlow, Keras, Numpy, Scipy and we can also download the MNIST dataset directly from kaggle,UCI Machine Learning Repository.

Project Requirements :

1.Load and Preprocess the Dataset :

--- Load the data from the keras library and for preprocessing normalize the pixel values (0-255) to [0,1] or [-1,1]
--- Resize images to 28 X 28 pixels.

For normalizing we divide the pixel value with the 255 to get the values between 0 and 1.

```
# scaling our data (converting our data values between 0 and 1 by dividing with 255)
X_train = X_train/255
X_test = X_test/255
X_train[0]
```

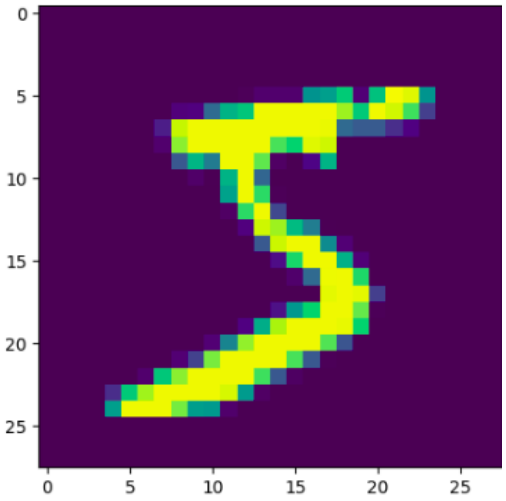
2. Splitting the Data :

--- Split the data into training (60,000) and testing sets (10,000)

```
(X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
```

Here we split our data into training sets and testing sets and our data is in the form of images. Lets see our first image of our dataset :

```
# this is our train and test data in the dataset
plt.imshow(X_train[0])
y_train[0]
```



3. Model Selection and Training:

We have to choose a suitable algorithm for this project. The best algorithm for this project is CNN(Convolutional Neural Network) which is specifically used for image classification.

--- We have to define the CNN model architecture

```
model = Sequential()
```

```
model.add(Flatten(input_shape =(28,28)))
model.add(Dense(128,activation = 'relu'))
model.add(Dense(10,activation = 'softmax'))
```

```
model.summary()
```

--- Next we have to compile our module for better understanding of how the model learns from data.

--- It also measures the diffreneces between predictions and actual values.

--- It evaluates model performances during training.

```
model.compile(optimizer = 'Adam',loss ='sparse_categorical_crossentropy',metrics = ['accuracy'])
history = model.fit(X_train,y_train,batch_size = 64,epochs = 20,verbose = 1,validation_split = 0.2)
```

Here we train our model by using the model.fit( ) method

4. Model Evaluation :

Evaluate the model's performance on the testing data.

```
model.evaluate(X_test,y_test)
```

--- By evaluating our data it will estimate model accuracy.
--- It will identify whether our model is overfitting or underfitting.
--- It will cmpare the models and gives the best model and best accuracy score
--- It is a classification based problem so we check the evaluation by following:
1. Accuracy
2. Precision
3.Confusion Matrix
4.F1-Score
Here we use the accuracy for evaluating our model.

--- If our project is on the regression type then for evaluation we use :
1.Mean Squared Error
2.Mean Absolute Error
3.R-Squared

5. Model Improvement :
For model improvement firts we have to check our data is correct or not.

```
predictions = model.predict(X_test)
predictions
```

For more improvement we have to apply data augmentation, regularization or ensemble methods.

In this way here we create a model that can accurately classify the handwritten digits into their corresponding categories.(0-9)

--- We have to try with different algorithms and hyperparameters techniques to see which model is best for this project. But I use CNN, which is specifically designed for image classification so,it gives the best accuracy.

*** For the project handwritten digit classification the whole code is given below which is available in the notebook

handwritten digit classification

Possible Algorithms used :
1. Logistic Regression
2.Decision Trees
3. Random Forest
4. Support Vector Machines
5.Convolutional Neural Network (CNN)
6. Recurrent Neural Networks (RNN)

Tools and Libraries used :

1. Python
2. Tensor Flow
3. Keras
4. Numpy
5. Scipy

Python Libraries :

This MNIST dataset is available in many libraries like :

1.TensorFlow :
tf.keras.datasets.mnist

2. Keras :
keras.datasets.mnist

3.Numpy :
numpy.datasets.mnist

4.Scipy :
scipy.datasets.mnist

5.pytorch :
torchvision.datasets.mnist

6.Scikit-learn :
sklearn.datasets.fetch_mldata('MNIST original')

Conclusion :

CNN effectively recognizes handwritten digits.

In [1]:
```
#import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten,Dense
```

```
2024-10-18 21:34:02.692602: W tensorflow/stream_executor/platform/default/
dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dler
ror: libcudart.so.11.0: cannot open shared object file: No such file or di
rectory
2024-10-18 21:34:02.692655: I tensorflow/stream_executor/cuda/cudart_stub.
cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your
machine.
```

In [3]:
```
(X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
```

In [4]:
```
# check the shape of the data
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

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

In [5]:
```
X_train
```

Out[5]:
```
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, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [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)
```

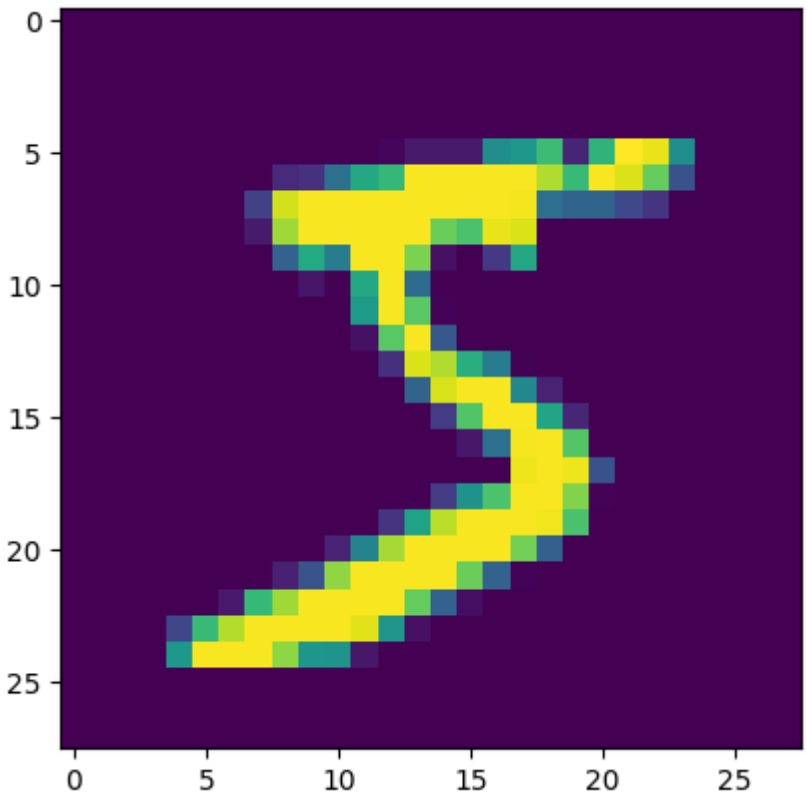
In [6]:
```
X_train[0]
```

```
Out[6]:  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)
```

In [10]:
```
# this is our train and test data in the dataset
plt.imshow(X_train[0])
y_train[0]
```

Out[10]: 5



In [35]:
```
# scaling our data (converting our data values between 0 and 1 by dividin
X_train = X_train/255
X_test = X_test/255
X_train[0]
```

```
Out[35]: 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.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ]])
```

In [48]:
```
model = Sequential()
```

In [49]:
```
model.add(Flatten(input_shape =(28,28)))
model.add(Dense(128,activation = 'relu'))
model.add(Dense(10,activation = 'softmax'))
```

In [50]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 flatten_2 (Flatten)          (None, 784)               0

 dense_4 (Dense)              (None, 128)               100480

 dense_5 (Dense)              (None, 10)                1290

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

In [51]:
```
model.compile(optimizer = 'Adam',loss ='sparse_categorical_crossentropy',
history = model.fit(X_train,y_train,batch_size = 64,epochs = 20,verbose =
```

```
Epoch 1/20
750/750 [==============================] - 5s 5ms/step - loss: 0.3336 - ac
curacy: 0.9066 - val_loss: 0.1809 - val_accuracy: 0.9492
Epoch 2/20
750/750 [==============================] - 4s 5ms/step - loss: 0.1496 - ac
curacy: 0.9571 - val_loss: 0.1282 - val_accuracy: 0.9626
Epoch 3/20
750/750 [==============================] - 4s 5ms/step - loss: 0.1035 - ac
curacy: 0.9697 - val_loss: 0.1140 - val_accuracy: 0.9650
Epoch 4/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0780 - ac
curacy: 0.9774 - val_loss: 0.0997 - val_accuracy: 0.9688
Epoch 5/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0615 - ac
curacy: 0.9818 - val_loss: 0.0862 - val_accuracy: 0.9739
Epoch 6/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0493 - ac
curacy: 0.9856 - val_loss: 0.0933 - val_accuracy: 0.9709
Epoch 7/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0412 - ac
curacy: 0.9879 - val_loss: 0.0891 - val_accuracy: 0.9728
Epoch 8/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0331 - ac
curacy: 0.9907 - val_loss: 0.0812 - val_accuracy: 0.9737
Epoch 9/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0269 - ac
curacy: 0.9923 - val_loss: 0.0802 - val_accuracy: 0.9754
Epoch 10/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0212 - ac
curacy: 0.9946 - val_loss: 0.0871 - val_accuracy: 0.9737
Epoch 11/20
750/750 [==============================] - 4s 6ms/step - loss: 0.0181 - ac
curacy: 0.9953 - val_loss: 0.0871 - val_accuracy: 0.9740
Epoch 12/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0147 - ac
curacy: 0.9961 - val_loss: 0.0868 - val_accuracy: 0.9758
Epoch 13/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0121 - ac
curacy: 0.9973 - val_loss: 0.0970 - val_accuracy: 0.9735
Epoch 14/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0092 - ac
curacy: 0.9982 - val_loss: 0.0907 - val_accuracy: 0.9761
Epoch 15/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0099 - ac
curacy: 0.9974 - val_loss: 0.1054 - val_accuracy: 0.9728
Epoch 16/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0070 - ac
curacy: 0.9987 - val_loss: 0.0952 - val_accuracy: 0.9747
Epoch 17/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0064 - ac
curacy: 0.9987 - val_loss: 0.0995 - val_accuracy: 0.9758
Epoch 18/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0063 - ac
curacy: 0.9985 - val_loss: 0.1081 - val_accuracy: 0.9743
Epoch 19/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0058 - ac
curacy: 0.9987 - val_loss: 0.1019 - val_accuracy: 0.9737
Epoch 20/20
750/750 [==============================] - 4s 5ms/step - loss: 0.0065 - ac
curacy: 0.9981 - val_loss: 0.1108 - val_accuracy: 0.9736
```

In [52]:
```
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1052 - ac
curacy: 0.9765
```

Out[52]:
```
[0.10519644618034363, 0.9764999747276306]
```

In [53]:
```
predictions = model.predict(X_test)
predictions
```

```
313/313 [==============================] - 1s 2ms/step
```

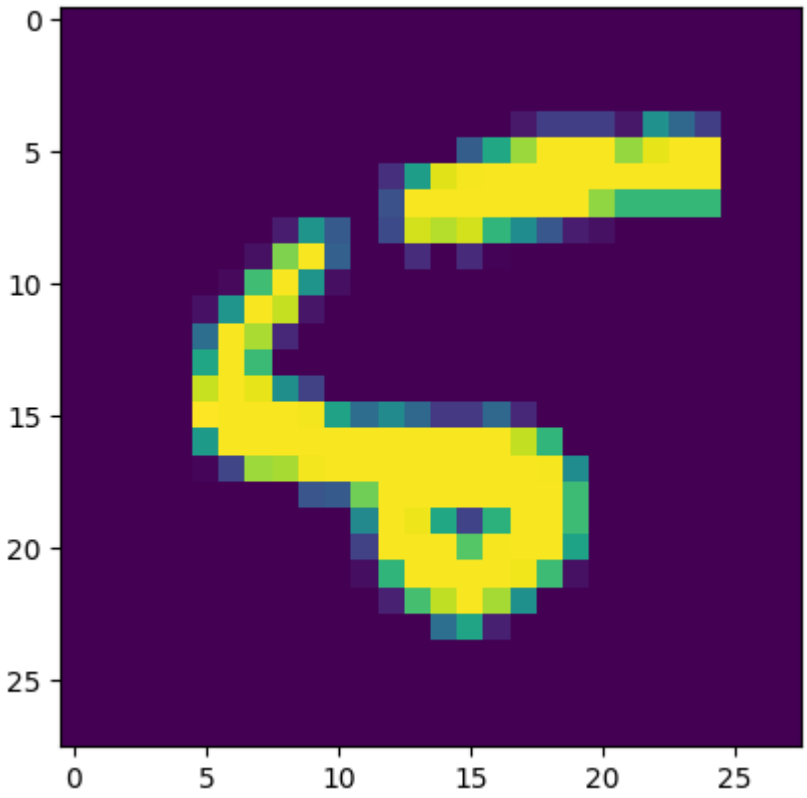
Out[53]:
```
array([[3.4312400e-10, 8.5000361e-13, 8.0243607e-09, ..., 9.9968475e-01,
        3.7059618e-09, 1.3984014e-07],
       [9.6464221e-13, 6.5860768e-08, 9.9999982e-01, ..., 3.4865960e-19,
        2.3181022e-09, 6.3745878e-19],
       [5.9042082e-09, 9.9998754e-01, 8.8219122e-07, ..., 5.5886312e-06,
        1.2419707e-06, 3.4368156e-08],
       ...,
       [1.6758108e-21, 6.9423375e-20, 1.2955888e-21, ..., 1.2005312e-11,
        2.1202732e-11, 2.2209376e-08],
       [1.7731031e-09, 1.5442420e-11, 1.2791945e-14, ..., 5.7403411e-09,
        9.8023440e-07, 3.7632328e-11],
       [5.9475249e-13, 2.4356166e-18, 1.5295274e-12, ..., 1.8506255e-16,
        2.3058054e-15, 1.0851202e-13]], dtype=float32)
```

In [54]:
```
predicted_classes = predictions.argmax(axis = -1)
predicted_classes
```

Out[54]:
```
array([7, 2, 1, ..., 4, 5, 6])
```

In [55]:
```
plt.imshow(X_test[8])
```

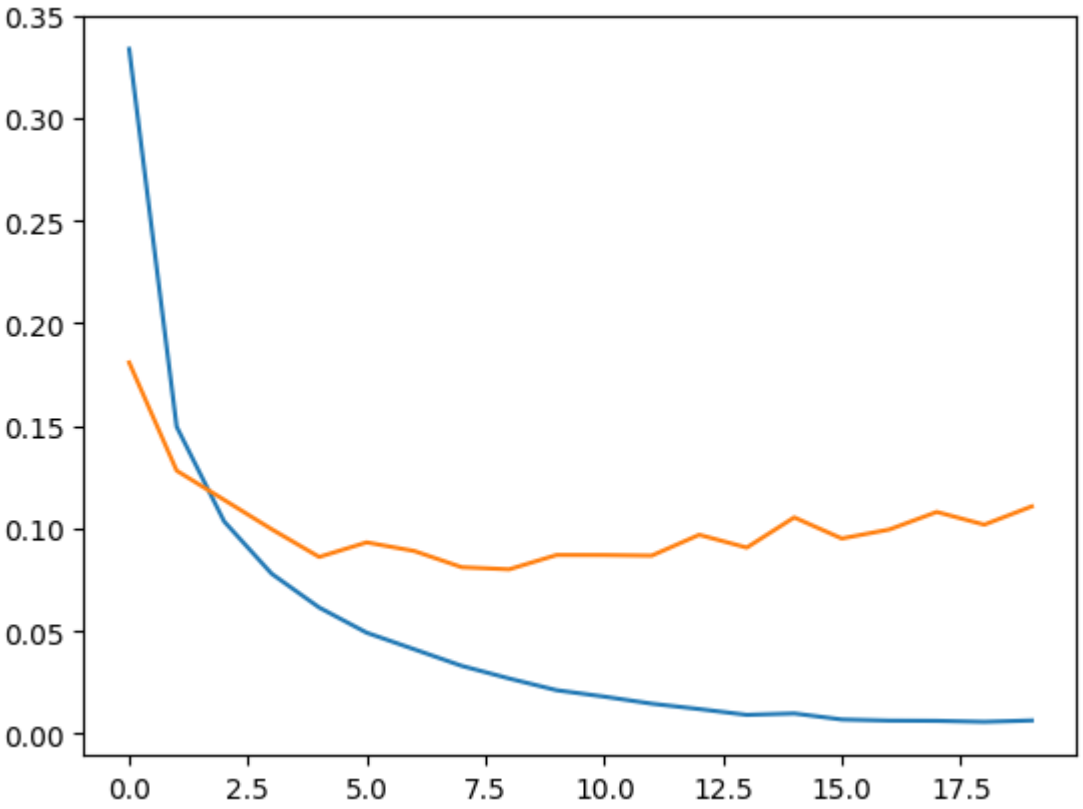
Out[55]:
```
<matplotlib.image.AxesImage at 0x7ff642aedf10>
```



In [57]:
```
plt.plot(history.history['loss'])
```

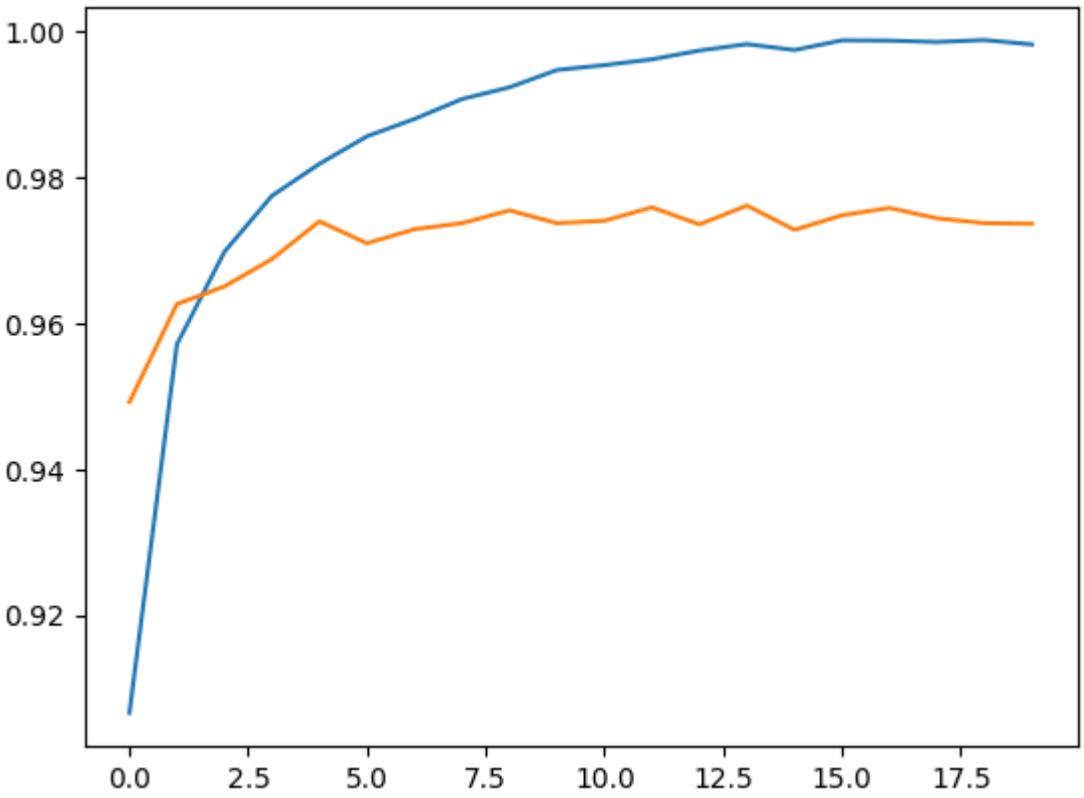
```
plt.plot(history.history['val_loss'])
```

Out[57]: [<matplotlib.lines.Line2D at 0x7ff642a5ed90>]



In [58]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

Out[58]: [<matplotlib.lines.Line2D at 0x7ff62aa9c940>]



In [ ]: