DEEP LEARNING ASSIGNMENT

SUPRATIM NAG (CSE-AIML/22/57)

Q-10:Write a python code to implement Deep Learning Techniques – Shallow and Deep Neural Network.(CNN)

In [1]:
```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten
from keras.datasets import mnist
from keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
print(x_train.shape)
print(x_test.shape)
print(y_train)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
ist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
(60000, 28, 28)
(10000, 28, 28)
[5 0 4 ... 5 6 8]
```

In [3]:
```
x_train[0]
```

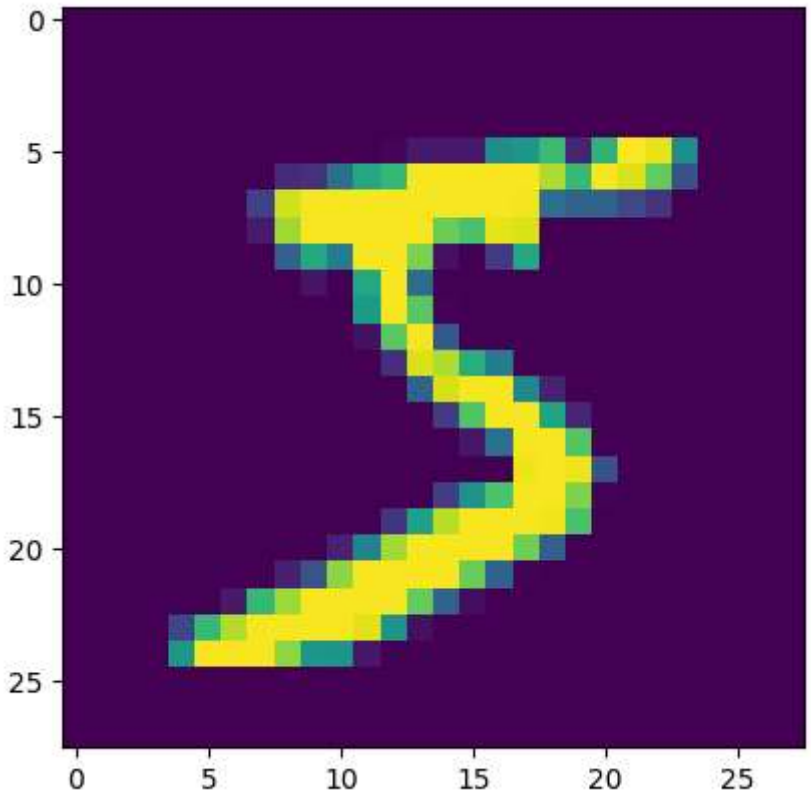
Out[3]: ndarray (28, 28) show data

In [4]:
```
y_train[0]
```

Out[4]: 5

In [5]:
```
plt.imshow(x_train[0])
```

Out[5]: <matplotlib.image.AxesImage at 0x7a3ecf220130>



In [6]:
```
#reshape the data to fit the model
x_train=x_train.reshape(60000,28,28,1)
x_test=x_test.reshape(10000,28,28,1)
```

In [7]:
```
#one-Hot encoding
y_train_one_hot=to_categorical(y_train)
y_test_one_hot=to_categorical(y_test)
#print the new label
print(y_train_one_hot[0])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

In [8]:
```
#build the CNN model
model=Sequential()
#add model layers
model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(Flatten())
model.add(Dense(10,activation='softmax'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:
107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [9]:
```
#compile the model
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']
```

In [10]:
```
#train the model
hist=model.fit(x_train,y_train_one_hot,validation_data=(x_test,y_test_one_hot),epoc
```

```
Epoch 1/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 184s 97ms/step - accuracy: 0.9114 - loss: 0.7167 - va
l_accuracy: 0.9734 - val_loss: 0.0799
Epoch 2/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 204s 99ms/step - accuracy: 0.9806 - loss: 0.0622 - va
l_accuracy: 0.9774 - val_loss: 0.0736
Epoch 3/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 206s 101ms/step - accuracy: 0.9873 - loss: 0.0401 - v
al_accuracy: 0.9769 - val_loss: 0.0824
```

In [11]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	
conv2d (Conv2D)	(None, 26, 26, 64)	
conv2d_1 (Conv2D)	(None, 24, 24, 32)	
flatten (Flatten)	(None, 18432)	
dense (Dense)	(None, 10)	

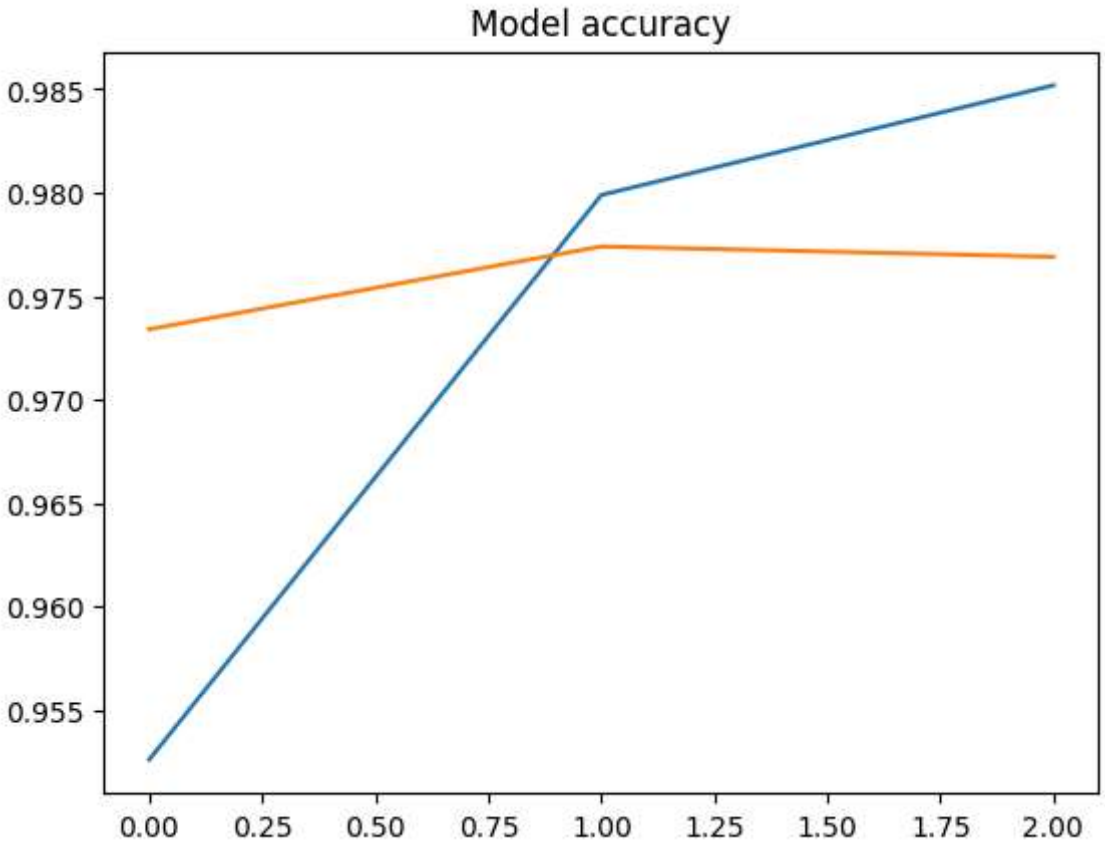
Total params: 610,304 (2.33 MB)

Trainable params: 203,434 (794.66 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 406,870 (1.55 MB)

In [12]:
```
#visualize the model accuracy
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
#plt.y_label('accuracy')
#plt.x_label('epochs')
#plt.legend(['train','val'],loc='upper left')
plt.show()
```



In [13]:
```
#show predictions for first 4 images on the test dataset
predictions=model.predict(x_test[:4])
predictions
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
```

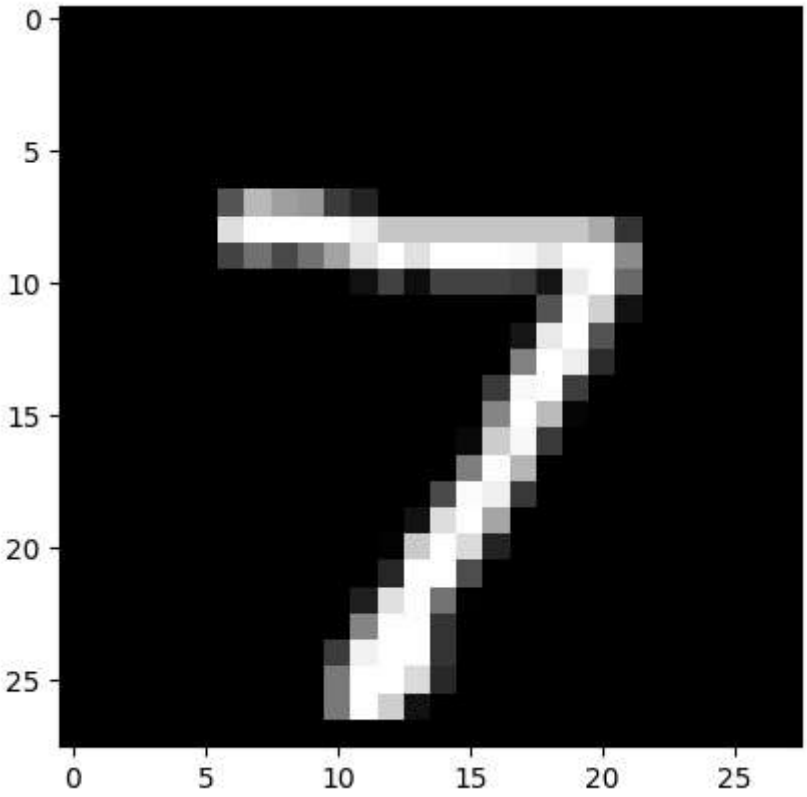
Out[13]:
```
array([[1.5148733e-09, 3.0295711e-15, 1.0905964e-08, 1.0532311e-07,
        2.6494740e-15, 3.5598130e-12, 1.2279894e-17, 9.9999976e-01,
        7.4414686e-08, 7.5158485e-10],
       [1.6115918e-07, 3.0229479e-09, 9.9999964e-01, 6.6490875e-11,
        1.5607125e-11, 6.2739032e-15, 2.5254786e-07, 7.4964410e-12,
        1.6842344e-10, 5.0604713e-15],
       [2.8732391e-06, 9.9611390e-01, 2.8093948e-05, 6.1378211e-07,
        3.5669554e-03, 1.8014359e-04, 1.9959425e-06, 4.2722405e-07,
        1.0497536e-04, 3.5680163e-08],
       [9.9999988e-01, 9.9179071e-15, 1.1003821e-07, 3.3295795e-12,
        4.1094276e-12, 5.0829133e-12, 1.7887742e-09, 1.3808325e-12,
        2.6626353e-08, 1.4247548e-08]], dtype=float32)
```

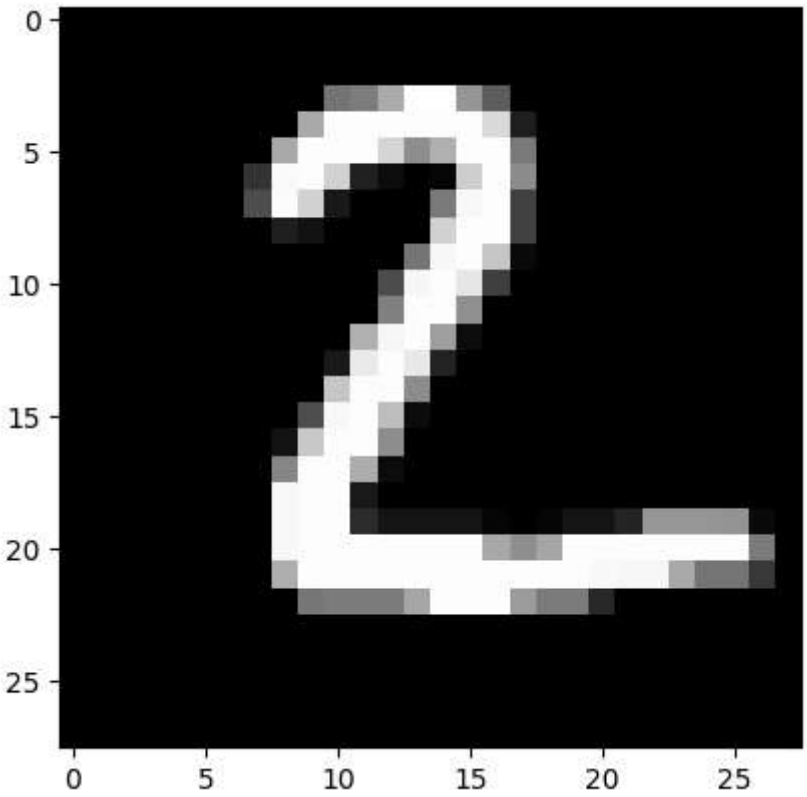
In [14]:
```
#print the prediction as number label for first 4 images
print(np.argmax(predictions,axis=1))
#print actual labels
print(y_test[:4])
```

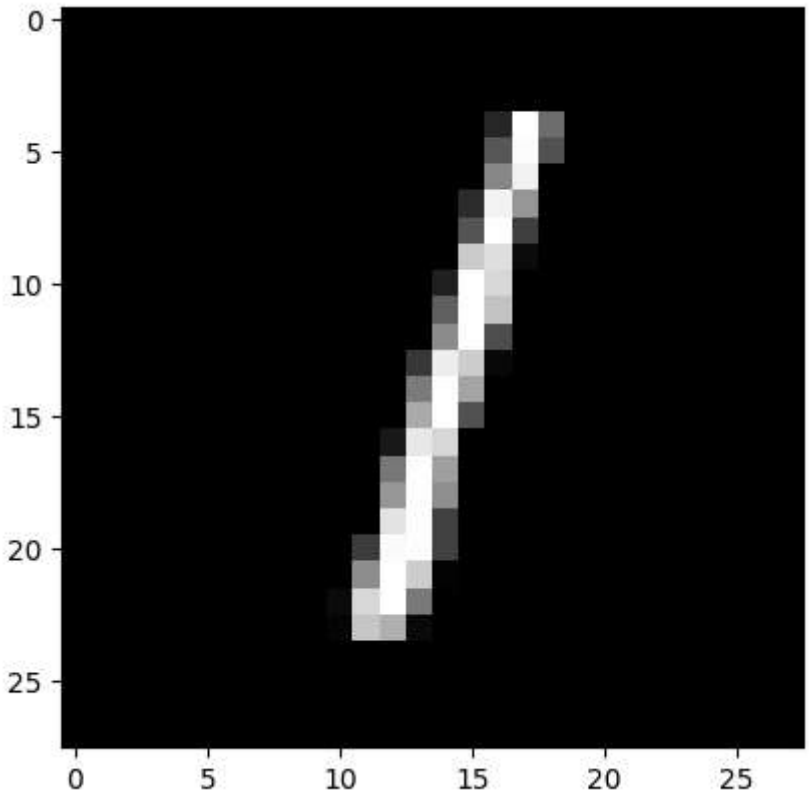
```
[7 2 1 0]
[7 2 1 0]
```

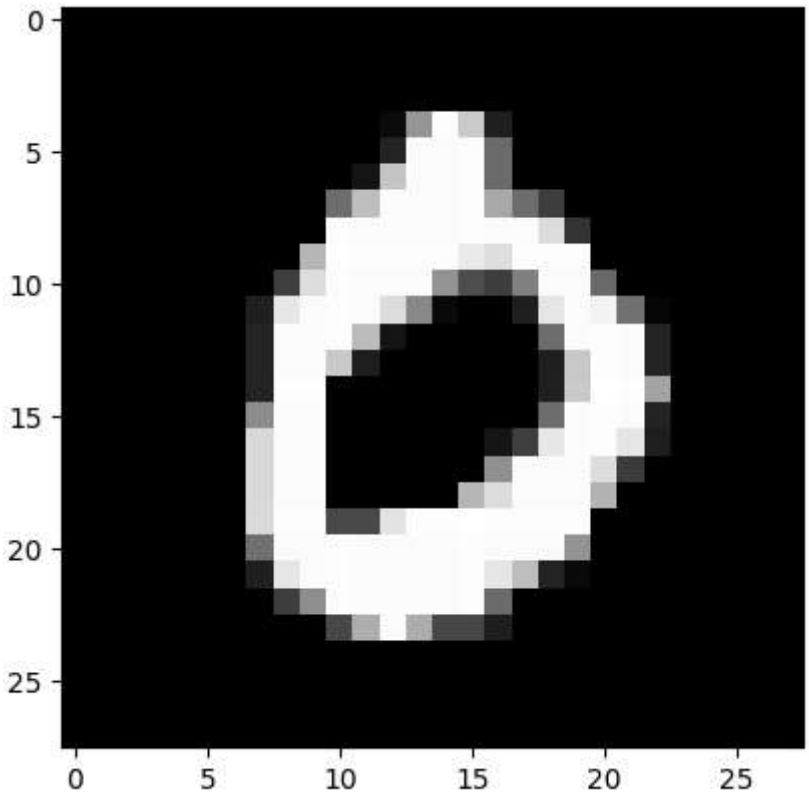
In [15]:
```
#show the first 4 images as pictures
for i in range(0,4):
  image=x_test[i]
  image=np.array(image,dtype='float')
```

```
pixels=image.reshape((28,28))
plt.imshow(pixels,cmap='gray')
plt.show()
```






0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25