In [1]:

```
from keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
```

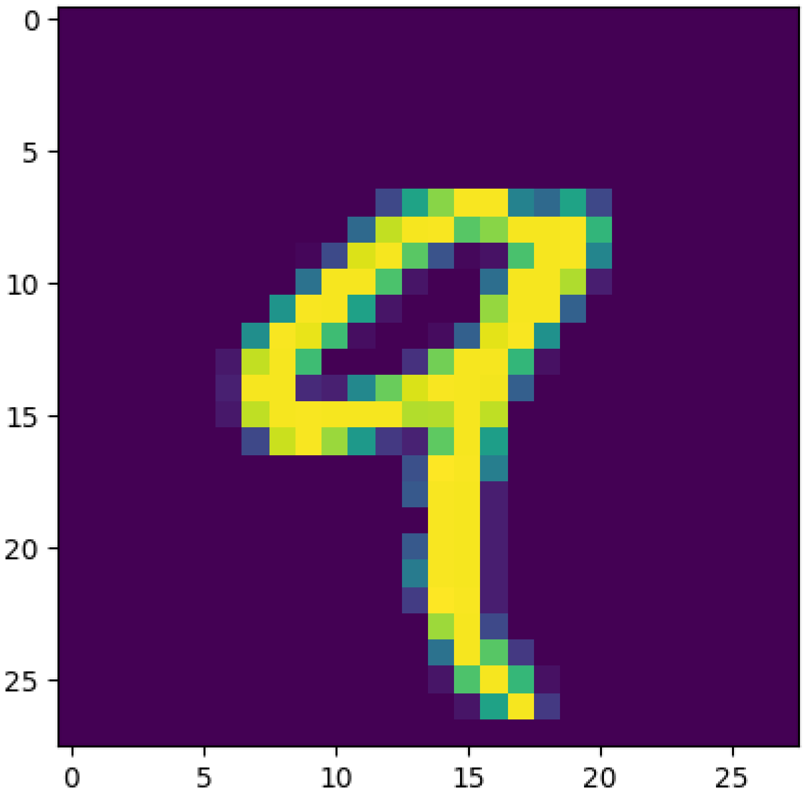
In [2]:

```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
```

In [3]:

```
print("Training data:")
plt.imshow(x_train[4])
plt.show()
print("Label of this image is",y_train[4])
```

```
Training data:
```



```
Label of this image is 9
```

In [4]:

```
#reshaping data as needed by the model
x_train=np.reshape(x_train,(-1,28,28,1))
x_test = np.reshape(x_test,(-1,28,28,1))
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[4]:

```
((60000, 28, 28, 1), (10000, 28, 28, 1), (60000,), (10000,))
```

In [5]:

```
#normalising
x_train = x_train / 255
x_test = x_test / 255
```

In [6]:

```
#implementing one hot encoding
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
```

In [7]:

```
#importing the model
from keras.models import Sequential
```

In [8]:

```
#creating model object
model = Sequential()
```

In [9]:

```
#importing layers
from keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

In [10]:

```
model = Sequential([
    Input(shape=(28, 28, 1)),  # Add this Input layer
    Conv2D(32, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

In [11]:

```
#compiling
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["acc
```

In [12]:

```
#training the model
model.fit(x_train, y_train, batch_size=100, epochs=5, validation_data=(x_test,
```

```
Epoch 1/5
600/600 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.8777 - loss: 0.4318
- val_accuracy: 0.9774 - val_loss: 0.0741
Epoch 2/5
600/600 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.9801 - loss: 0.0699
- val_accuracy: 0.9816 - val_loss: 0.0555
Epoch 3/5
600/600 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.9872 - loss: 0.0419
- val_accuracy: 0.9809 - val_loss: 0.0605
Epoch 4/5
600/600 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.9908 - loss: 0.0307
- val_accuracy: 0.9857 - val_loss: 0.0458
Epoch 5/5
600/600 ━━━━━━━━━━━━━━━━━━━━ 12s 21ms/step - accuracy: 0.9937 - loss: 0.0214
- val_accuracy: 0.9850 - val_loss: 0.0417
```

Out[12]:

```
<keras.src.callbacks.history.History at 0x20d20e431f0>
```

In [14]:

```
#model train and test scores
model.evaluate(x_train, y_train)
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9958 - loss: 0.0158
```

Out[14]:

```
[0.015368202701210976, 0.9960333108901978]
```

In [15]:

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9812 - loss: 0.0516
```

Out[15]:

```
[0.041743360459804535, 0.9850000143051147]
```

In [ ]: