In [1]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Flatten
```

In [2]:
```
(X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 3s 0us/step
```

In [3]:
```
X_test.shape
```

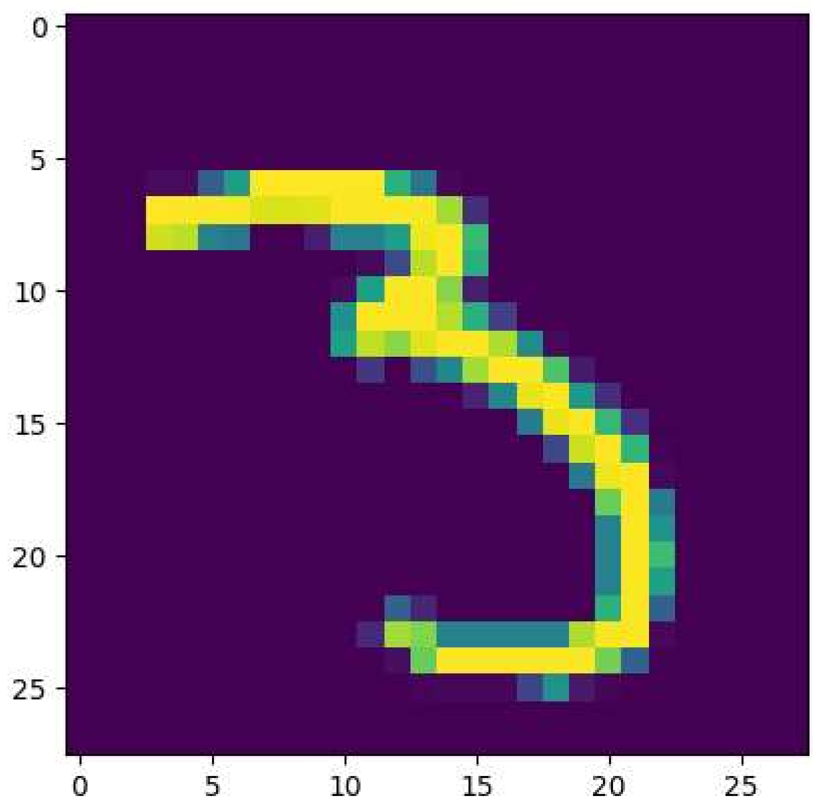
Out[3]:
```
(10000, 28, 28)
```

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

Out[4]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [6]:
```
import matplotlib.pyplot as plt
plt.imshow(X_train[86])
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x1fef98026d0>
```



In [7]:
```
X_train = X_train/255
X_test = X_test/255
```

In [8]:
```
X_train[0]
```

Out[8]:
```
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.        ,
```

In [9]:
```
model = Sequential()

model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

In [10]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten (Flatten)           (None, 784)               0         
                                                                 
 dense (Dense)               (None, 128)               100480    
                                                                 
 dense_1 (Dense)             (None, 32)                4128      
                                                                 
 dense_2 (Dense)             (None, 10)                330       
                                                                 
=================================================================
Total params: 104938 (409.91 KB)
Trainable params: 104938 (409.91 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [11]:
```
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=
```

In [12]:
```
history = model.fit(X_train,y_train,epochs=25,validation_split=0.2)
```

```
Epoch 1/25
1500/1500 [==============================] - 4s 2ms/step - loss: 0.2805 - acc
uracy: 0.9184 - val_loss: 0.1554 - val_accuracy: 0.9538
Epoch 2/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.1234 - acc
uracy: 0.9635 - val_loss: 0.1187 - val_accuracy: 0.9663
Epoch 3/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0828 - acc
uracy: 0.9749 - val_loss: 0.1036 - val_accuracy: 0.9684
Epoch 4/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0631 - acc
uracy: 0.9807 - val_loss: 0.1042 - val_accuracy: 0.9706
Epoch 5/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0484 - acc
uracy: 0.9844 - val_loss: 0.0922 - val_accuracy: 0.9744
Epoch 6/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0376 - acc
uracy: 0.9883 - val_loss: 0.1008 - val_accuracy: 0.9733
Epoch 7/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0306 - acc
uracy: 0.9901 - val_loss: 0.0999 - val_accuracy: 0.9743
Epoch 8/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0262 - acc
uracy: 0.9915 - val_loss: 0.1003 - val_accuracy: 0.9750
Epoch 9/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0215 - acc
uracy: 0.9929 - val_loss: 0.1142 - val_accuracy: 0.9735
Epoch 10/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0200 - acc
uracy: 0.9933 - val_loss: 0.1017 - val_accuracy: 0.9767
Epoch 11/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0187 - acc
uracy: 0.9938 - val_loss: 0.1217 - val_accuracy: 0.9723
Epoch 12/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0150 - acc
uracy: 0.9948 - val_loss: 0.1179 - val_accuracy: 0.9747
Epoch 13/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0123 - acc
uracy: 0.9961 - val_loss: 0.1255 - val_accuracy: 0.9743
Epoch 14/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0143 - acc
uracy: 0.9952 - val_loss: 0.1216 - val_accuracy: 0.9735
Epoch 15/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0107 - acc
uracy: 0.9964 - val_loss: 0.1391 - val_accuracy: 0.9722
Epoch 16/25
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0115 - acc
uracy: 0.9963 - val_loss: 0.1260 - val_accuracy: 0.9763
Epoch 17/25
1500/1500 [==============================] - 4s 2ms/step - loss: 0.0118 - acc
uracy: 0.9966 - val_loss: 0.1494 - val_accuracy: 0.9750
Epoch 18/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0101 - acc
uracy: 0.9968 - val_loss: 0.1470 - val_accuracy: 0.9712
Epoch 19/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0089 - acc
uracy: 0.9970 - val_loss: 0.1575 - val_accuracy: 0.9722
```

```
Epoch 20/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0117 - acc
uracy: 0.9963 - val_loss: 0.1610 - val_accuracy: 0.9743
Epoch 21/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0081 - acc
uracy: 0.9971 - val_loss: 0.1429 - val_accuracy: 0.9754
Epoch 22/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0075 - acc
uracy: 0.9975 - val_loss: 0.1495 - val_accuracy: 0.9746
Epoch 23/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0093 - acc
uracy: 0.9969 - val_loss: 0.1652 - val_accuracy: 0.9733
Epoch 24/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0058 - acc
uracy: 0.9981 - val_loss: 0.1512 - val_accuracy: 0.9755
Epoch 25/25
1500/1500 [==============================] - 3s 2ms/step - loss: 0.0094 - acc
uracy: 0.9970 - val_loss: 0.1666 - val_accuracy: 0.9732
```

In [13]:
```
y_prob = model.predict(X_test)
```

```
313/313 [==============================] - 0s 1ms/step
```

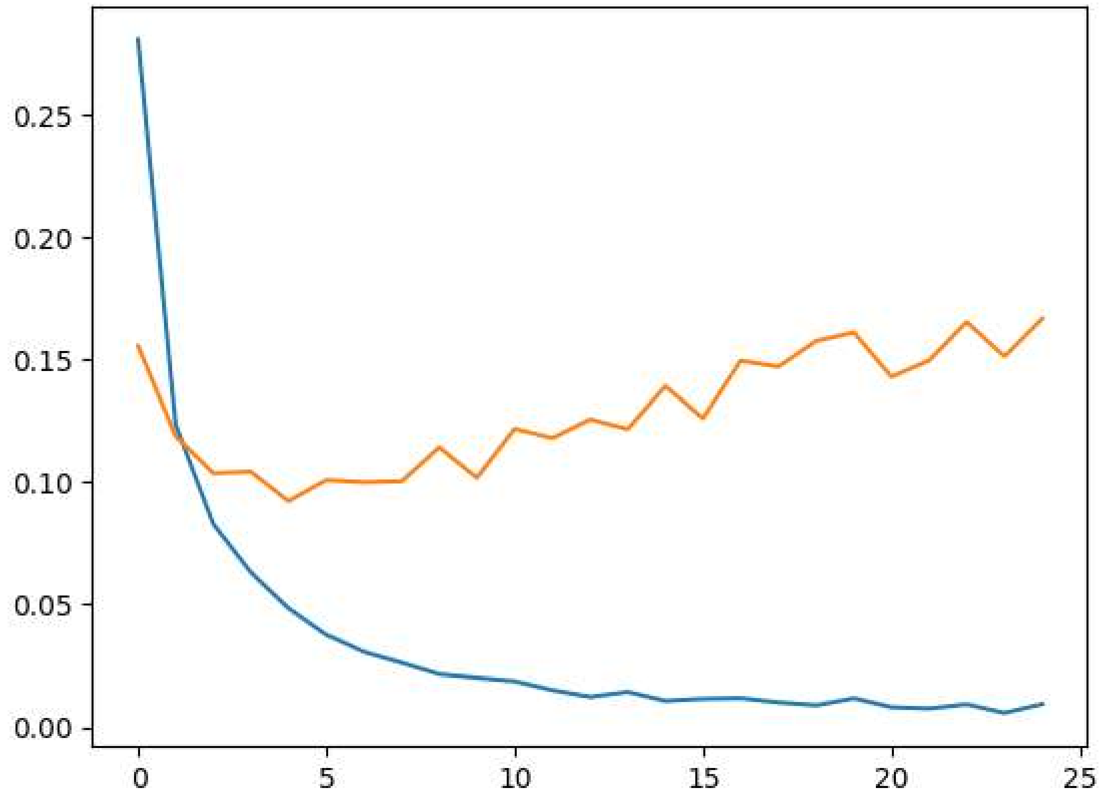
In [14]:
```
y_pred = y_prob.argmax(axis=1)
```

In [15]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[15]:
```
0.9759
```

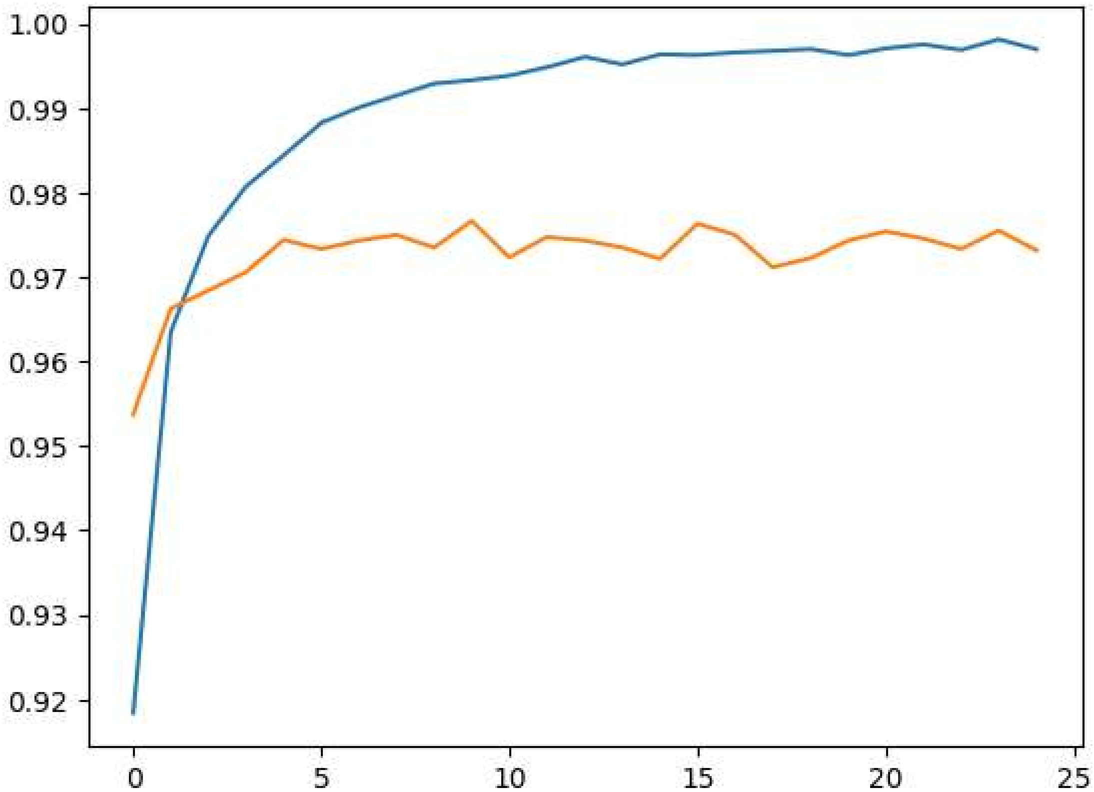
In [16]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x1fefce0d850>]
```



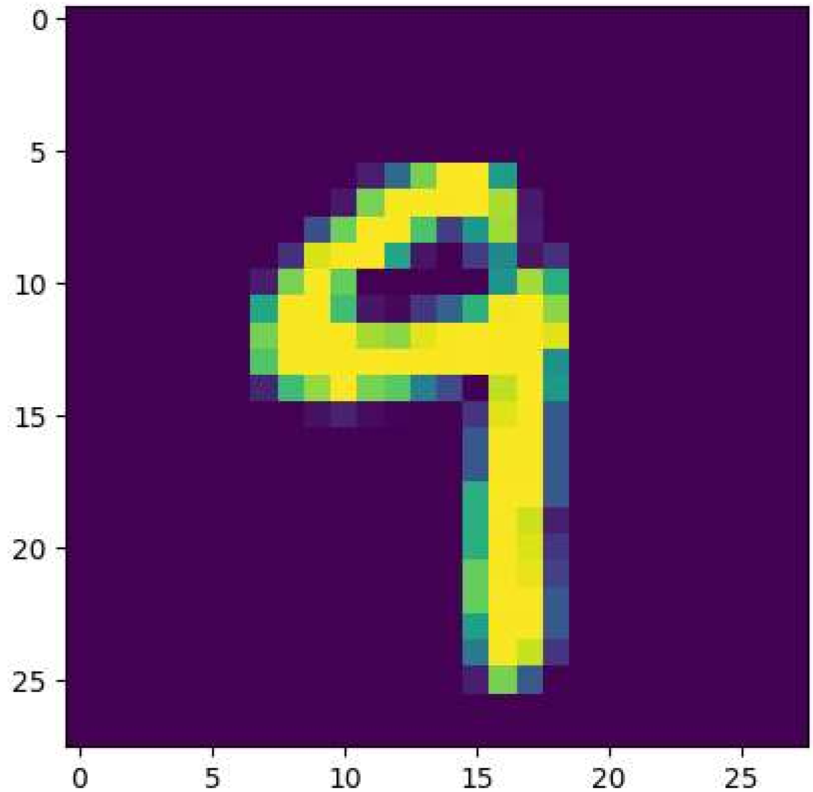
In [17]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

Out[17]:
```
[<matplotlib.lines.Line2D at 0x1fefd054350>]
```



In [20]:
```
plt.imshow(X_test[108])
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x1fefd0f26d0>
```



In [22]:
```
model.predict(X_test[108].reshape(1,28,28)).argmax(axis=1)
```

```
1/1 [==============================] - 0s 20ms/step
```

Out[22]:
```
array([9], dtype=int64)
```

In [ ]: