```
import keras
from keras.models import Sequential
from keras.layers import Dense,Flatten
```

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

```
X_train.shape
```

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

```
y_train.shape
```

```
    (60000,)
```

```
import numpy as np

# Assuming y_train is your target labels
num_classes = len(np.unique(y_train))
print("Number of classes:", num_classes)
```

```
    Number of classes: 10
```

```
X_test.shape
```

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

```
y_test.shape
```

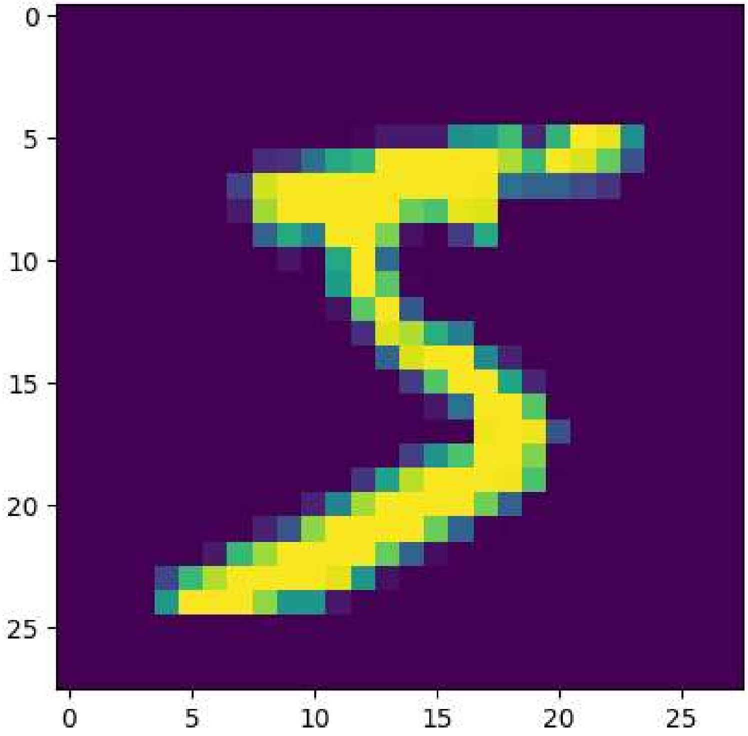
```
    (10000,)
```

```
y_train
```

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

```
import matplotlib.pyplot as plt
plt.imshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x780465c38370>
```



```
X_train = X_train/255
X_test = X_test/255
```

```
X_train[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.        ],
```

```
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'))
```

```
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)
_________________________________________________________________
```

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
history = model.fit(X_train,y_train,epochs=3,validation_split=0.2)
```

```
=====] - 15s 9ms/step - loss: 0.2815 - accuracy: 0.9180 - val_loss: 0.1436 - val_accuracy: 0.958

=====] - 13s 9ms/step - loss: 0.1191 - accuracy: 0.9646 - val_loss: 0.1087 - val_accuracy: 0.968

=====] - 7s 5ms/step - loss: 0.0812 - accuracy: 0.9758 - val_loss: 0.1116 - val_accuracy: 0.9660
```

```
y_prob = model.predict(X_test)
print(y_prob)
```

```
313/313 [==============================] - 1s 2ms/step
[[1.78507761e-07 7.68193968e-06 2.08849157e-03 ... 9.69996631e-01
  5.16456703e-06 1.70557689e-06]
 [2.00629877e-08 1.01916601e-04 9.99888361e-01 ... 1.94382981e-08
  1.04407832e-06 2.08265002e-12]
 [1.23298193e-07 9.98178065e-01 9.25001630e-04 ... 2.92041164e-04
  4.86784731e-04 4.96727080e-06]
 ...
 [1.12880734e-07 2.58517730e-07 5.74004311e-08 ... 4.59586363e-06
  3.79294623e-04 1.04841871e-04]
 [1.39115537e-08 2.31078593e-06 2.37699318e-08 ... 2.74862284e-08
  1.72401225e-04 1.26447148e-07]
 [7.57534391e-09 6.16657800e-08 6.83047176e-07 ... 2.95458970e-11
  1.00674635e-09 2.11102774e-10]]
```

```
y_pred = y_prob.argmax(axis=1)
```

```
print(y_pred)
```

```
[7 2 1 ... 4 5 6]
```

```
# Assuming you have a list or array of class labels
class_labels = ['class_0', 'class_1', 'class_2','class_3','class_4', 'class_5','class_6','class_7',

# Mapping indices to class labels
y_pred_classes = [class_labels[i] for i in y_pred]
```

```
y_pred_classes
```

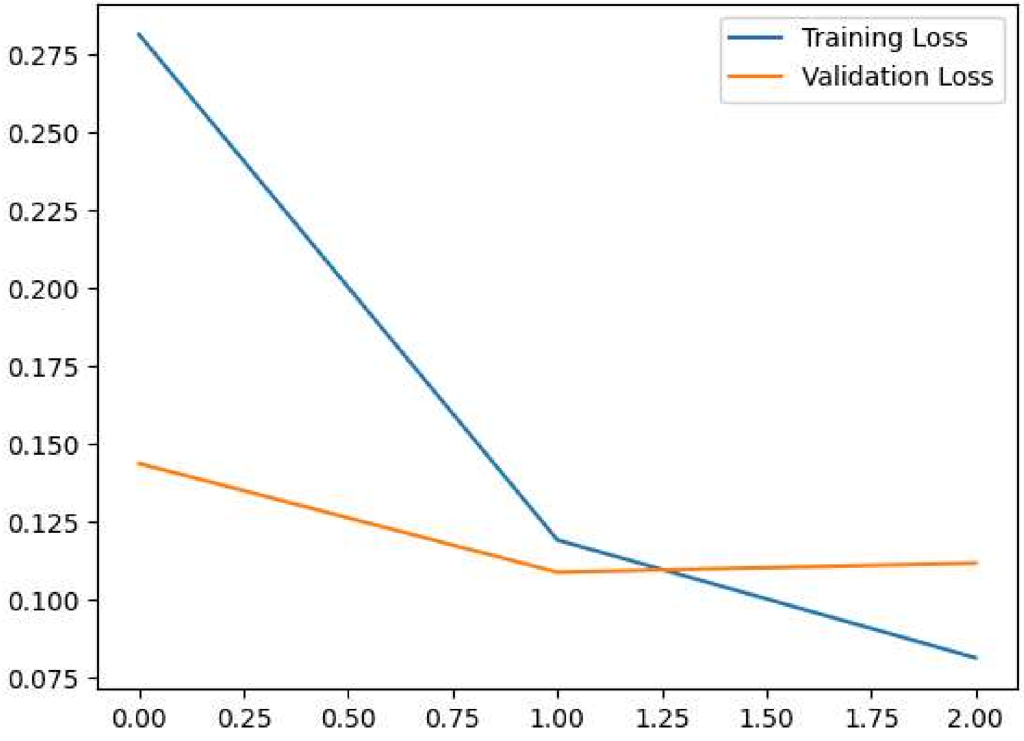
```
      'class_9',
      'class_7',
      'class_9',
      'class_4',
      'class_4',
      'class_9',
      'class_2',
      'class_5',
      'class_4',
      'class_7',
      'class_6',
      'class_4',
      'class_9',
      'class_0',
      'class_5',
      'class_8',
      'class_5',
      'class_6',
      'class_6',
      'class_5',
      'class_7',
      'class_8',
      'class_1',
      'class_0',
      'class_1',
      'class_6',
      'class_4',
      'class_6',
      'class_7',
      'class_3',
      'class_1',
      'class_7',
      'class_1',
      'class_8',
      'class_2',
      'class_0',
      'class_2',
      'class_9',
      'class_8',
      'class_5',
      'class_5',
      'class_1',
      'class_5',
      'class_6',
      'class_0',
      'class_3',
      'class_4',
      'class_4',
      'class_6'
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

```
0.9693
```

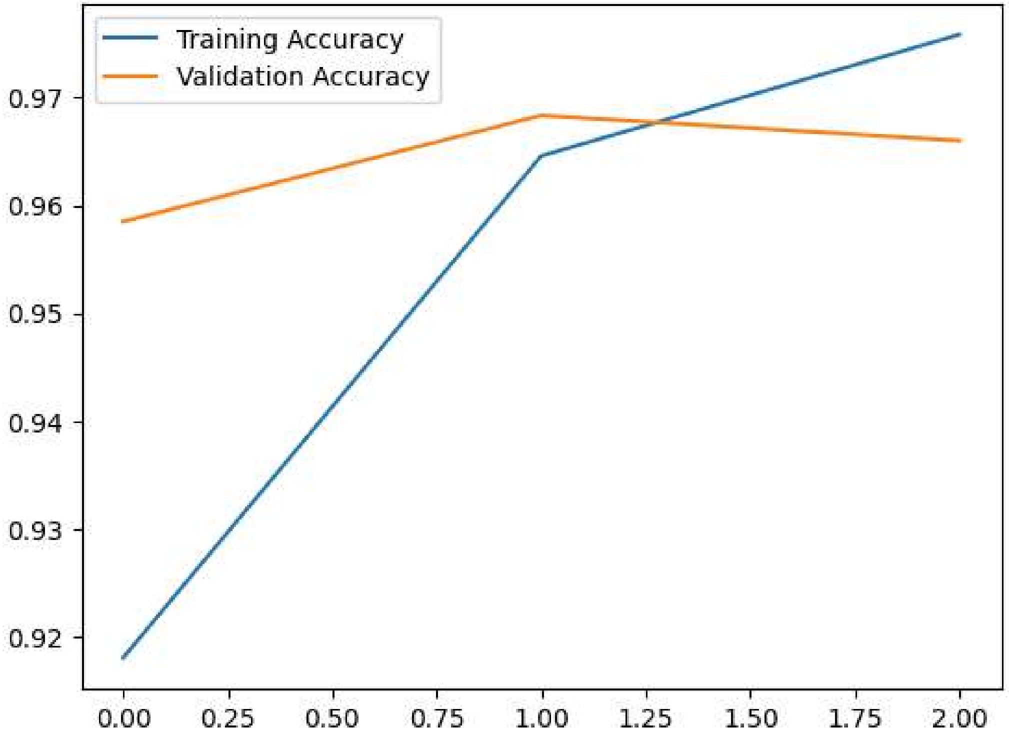
```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x780440c5cd90>
```



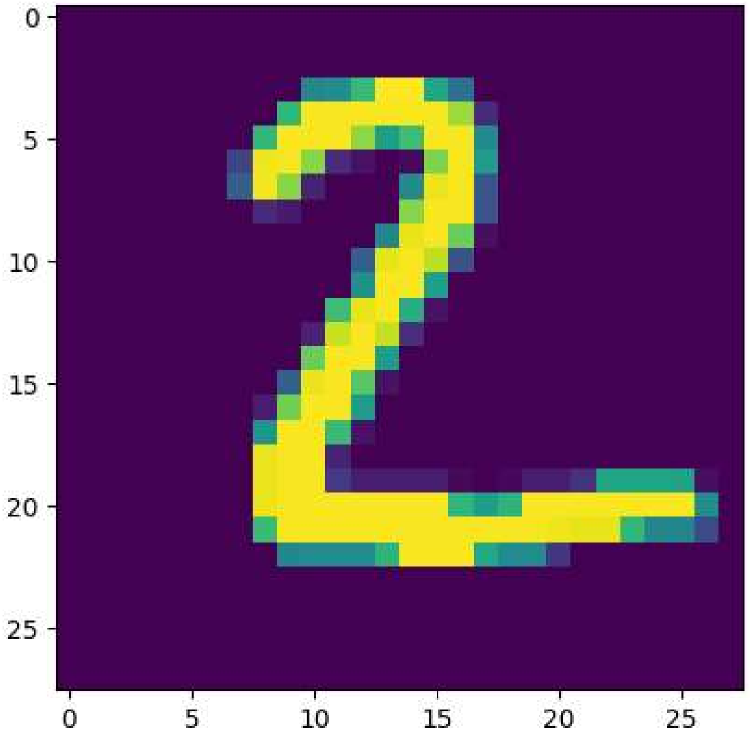
```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'],label='Validation Accuracy')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7804408f29b0>
```



```
plt.imshow(X_test[1])
model.predict(X_test[1].reshape(1,28,28)).argmax(axis=1)
model.predict(X_test[1].reshape(1,28,28)).argmax(axis=1)
```

```
1/1 [==============================] - 0s 21ms/step
array([2])
```


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