OUTPUTS

Output 1:

Data which is converted into a numpy array i.e. data and labels

```
[8]
# STEP 4: To feed the data to the model, we have to convert the list into Numpy array
data = np.array(data)
main_label = np.array(main_label)
print(data)
print(data.shape, main_label.shape)
# the shape is (X, 30, 30, 3) as (total number of images, 30 X 30, with RGB value(3) )

[[[[ 75  78  80]
   [ 74  76  78]
   [ 85  86  84]
   ...
   [ 68  75  74]
   [ 65  69  68]
   [ 66  67  66]]

  [[ 83  84  86]
   [ 80  80  82]
   [ 88  88  83]
   ...
   [ 73  77  78]
   [ 76  78  75]
   [ 80  80  78]]

  [[ 78  78  80]
   [ 86  85  86]
   [ 90  89  90]
   ...
   [ 71  74  71]
   [ 73  74  69]
   [ 78  78  74]]

  ...
```

Output 2:

Shape of the dataset -> the list which contains the data and the labels that specifies the total number of images in the format (a,b,c,d) ->(shape of data,30x30 pixels, RGB colour)

```
   [ 10   9  10]
   ...
   [ 10   9  11]
   [ 10   9  11]
   [  9   9  11]]

  [[ 11  11  13]
   [ 11  10  12]
   [  9   9  10]
   ...
   [ 10   9  11]
   [  9   9  11]
   [ 10   9  11]]

  [[ 11  11  12]
   [ 10  10  11]
   [  9   9  10]
   ...
   [ 10   9  12]
   [ 10  10  11]
   [ 10   9  11]]]]
(39209, 30, 30, 3) (39209,)
```

Output 3:

Shape of training data and testing data, split in the ratio of 75:25 with random state of 43

```
File Edit Selection View Go Run Terminal Help                main.ipynb* - pycol - Visual Studio Code
problemstmt.txt ●   guiMainn.ipynb   main.ipynb* ×

    ...
    [ 10   9  12]
    [ 10  10  11]
    [ 10   9  11]]]]
(39209, 30, 30, 3) (39209,)

[9]
# step 5 : spliting the train and test data using sklearn

X_train, X_test, y_train, y_test = train_test_split(data,main_label,test_size=0.25,random_state=43)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
# catagorical is used to convert the labels present in y_train,y_test as ONE-HOT ENCODING
y_train = to_categorical(y_train,43)
y_test = to_categorical(y_test,43)

(29406, 30, 30, 3) (9803, 30, 30, 3) (29406,) (9803,)
```

Output 4:

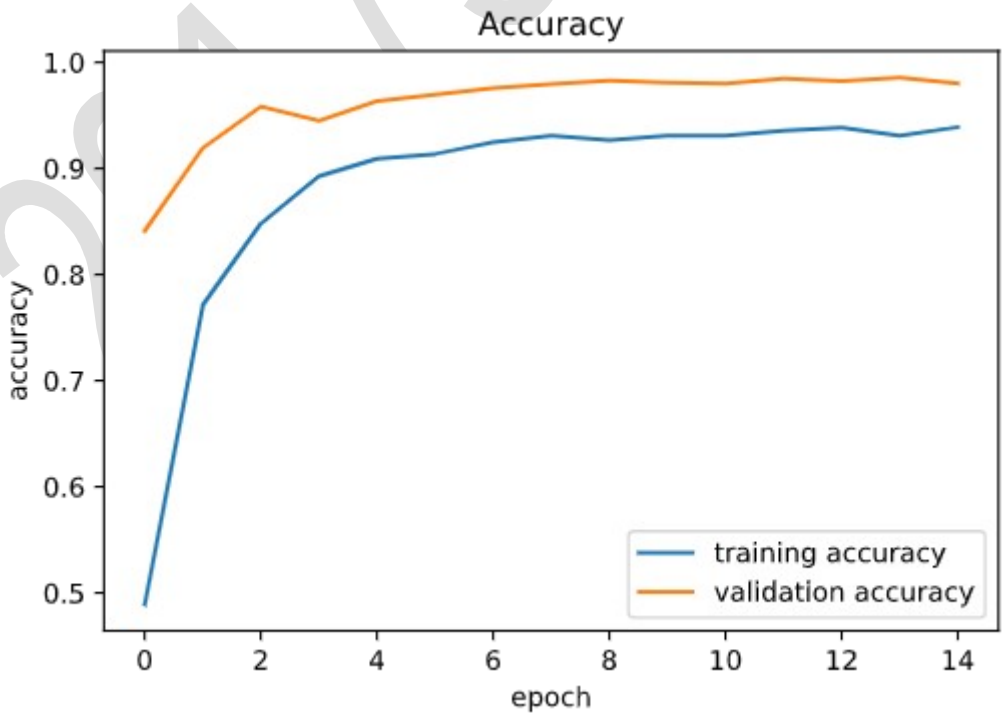
fitting the training dataset using Keras

```
Train on 29406 samples, validate on 9803 samples
Epoch 1/15
29406/29406 [==============================] - 120s 4ms/step - loss: 1.8868 - accuracy: 0.5108 - val_loss: 0.4678 - val_accuracy: 0.8755
Epoch 2/15
29406/29406 [==============================] - 119s 4ms/step - loss: 0.6809 - accuracy: 0.8019 - val_loss: 0.2470 - val_accuracy: 0.9312
Epoch 3/15
29406/29406 [==============================] - 128s 4ms/step - loss: 0.4268 - accuracy: 0.8767 - val_loss: 0.1615 - val_accuracy: 0.9558
Epoch 4/15
29406/29406 [==============================] - 112s 4ms/step - loss: 0.3442 - accuracy: 0.9017 - val_loss: 0.1359 - val_accuracy: 0.9662
Epoch 5/15
29406/29406 [==============================] - 99s 3ms/step - loss: 0.2860 - accuracy: 0.9187 - val_loss: 0.1200 - val_accuracy: 0.9708
Epoch 6/15
29406/29406 [==============================] - 100s 3ms/step - loss: 0.2840 - accuracy: 0.9238 - val_loss: 0.1131 - val_accuracy: 0.9718
Epoch 7/15
29406/29406 [==============================] - 100s 3ms/step - loss: 0.2819 - accuracy: 0.9240 - val_loss: 0.1051 - val_accuracy: 0.9733
Epoch 8/15
29406/29406 [==============================] - 99s 3ms/step - loss: 0.2532 - accuracy: 0.9305 - val_loss: 0.0670 - val_accuracy: 0.9805
Epoch 9/15
29406/29406 [==============================] - 103s 3ms/step - loss: 0.2669 - accuracy: 0.9320 - val_loss: 0.0763 - val_accuracy: 0.9783
Epoch 10/15
29406/29406 [==============================] - 105s 4ms/step - loss: 0.2576 - accuracy: 0.9333 - val_loss: 0.0746 - val_accuracy: 0.9809
Epoch 11/15
29406/29406 [==============================] - 106s 4ms/step - loss: 0.2396 - accuracy: 0.9353 - val_loss: 0.0694 - val_accuracy: 0.9810
Epoch 12/15
29406/29406 [==============================] - 107s 4ms/step - loss: 0.2434 - accuracy: 0.9387 - val_loss: 0.0630 - val_accuracy: 0.9824
```

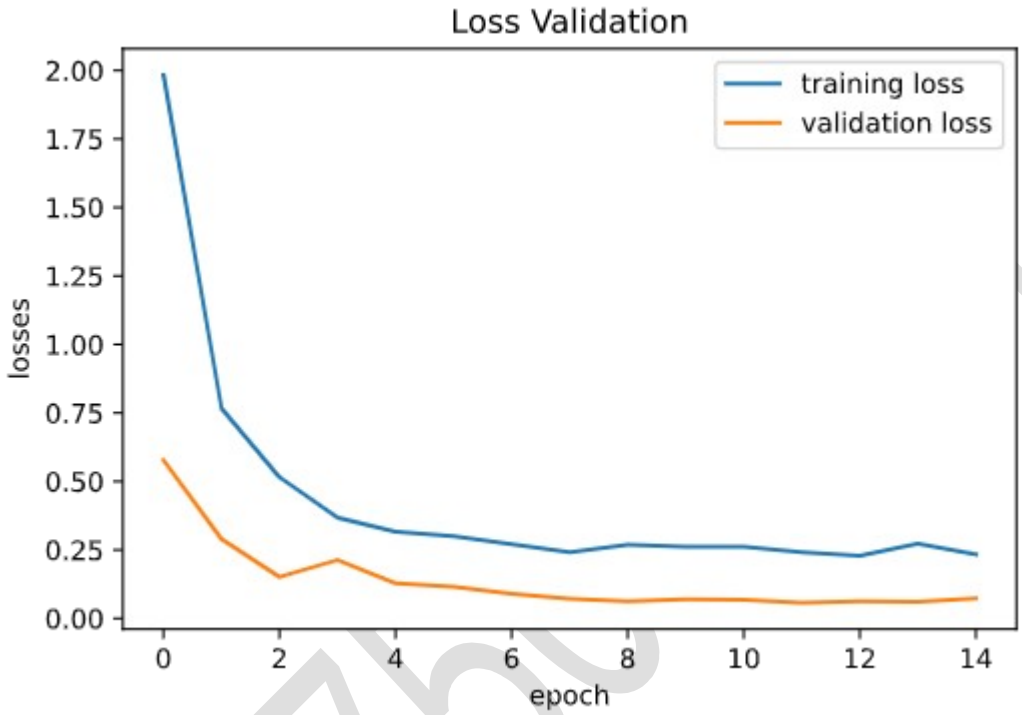
```
29406/29406 [==============================] - 128s 4ms/step - loss: 0.4268 - accuracy: 0.8767 - val_loss: 0.1615 - val_accuracy: 0.9558
Epoch 4/15
29406/29406 [==============================] - 112s 4ms/step - loss: 0.3442 - accuracy: 0.9017 - val_loss: 0.1359 - val_accuracy: 0.9662
Epoch 5/15
29406/29406 [==============================] - 99s 3ms/step - loss: 0.2860 - accuracy: 0.9187 - val_loss: 0.1200 - val_accuracy: 0.9708
Epoch 6/15
29406/29406 [==============================] - 100s 3ms/step - loss: 0.2840 - accuracy: 0.9238 - val_loss: 0.1131 - val_accuracy: 0.9718
Epoch 7/15
29406/29406 [==============================] - 100s 3ms/step - loss: 0.2819 - accuracy: 0.9240 - val_loss: 0.1051 - val_accuracy: 0.9733
Epoch 8/15
29406/29406 [==============================] - 99s 3ms/step - loss: 0.2532 - accuracy: 0.9305 - val_loss: 0.0670 - val_accuracy: 0.9805
Epoch 9/15
29406/29406 [==============================] - 103s 3ms/step - loss: 0.2669 - accuracy: 0.9320 - val_loss: 0.0763 - val_accuracy: 0.9783
Epoch 10/15
29406/29406 [==============================] - 105s 4ms/step - loss: 0.2576 - accuracy: 0.9333 - val_loss: 0.0746 - val_accuracy: 0.9809
Epoch 11/15
29406/29406 [==============================] - 106s 4ms/step - loss: 0.2396 - accuracy: 0.9353 - val_loss: 0.0694 - val_accuracy: 0.9810
Epoch 12/15
29406/29406 [==============================] - 107s 4ms/step - loss: 0.2434 - accuracy: 0.9387 - val_loss: 0.0630 - val_accuracy: 0.9824
Epoch 13/15
29406/29406 [==============================] - 107s 4ms/step - loss: 0.2294 - accuracy: 0.9437 - val_loss: 0.0603 - val_accuracy: 0.9848
Epoch 14/15
29406/29406 [==============================] - 108s 4ms/step - loss: 0.2151 - accuracy: 0.9459 - val_loss: 0.0512 - val_accuracy: 0.9867
Epoch 15/15
29406/29406 [==============================] - 73s 2ms/step - loss: 0.2195 - accuracy: 0.9447 - val_loss: 0.0801 - val_accuracy: 0.9791
```

Output 5:

Accuracy Graph

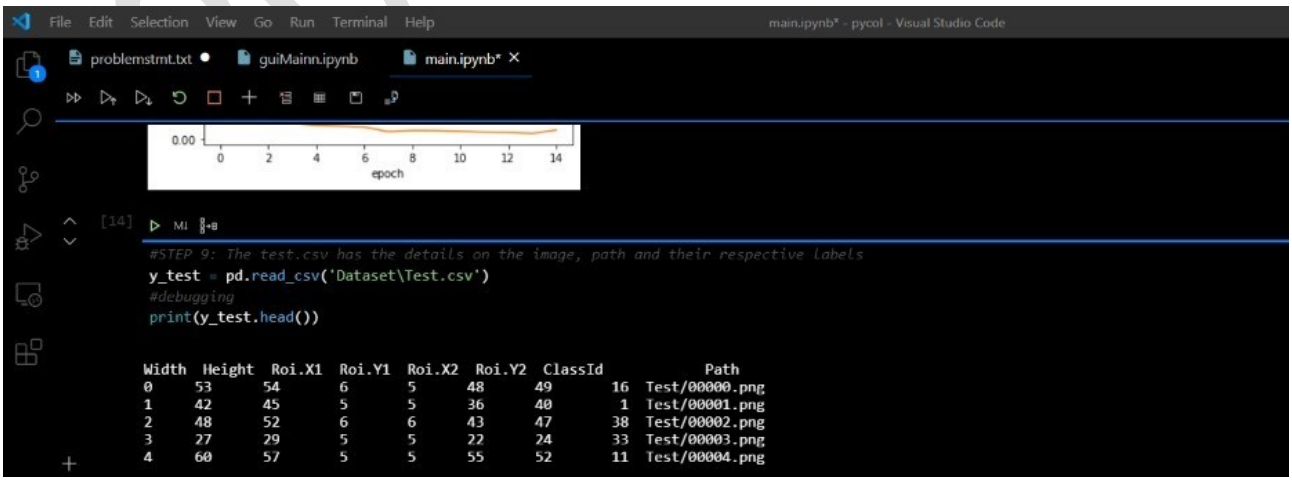


Loss Graph



Output 6:

head of the test CSV dataset



Output 7:

accuracy

```
#accuracy
print(accuracy_score(labels, pred))

0.9468725257323832
```

Output 8:

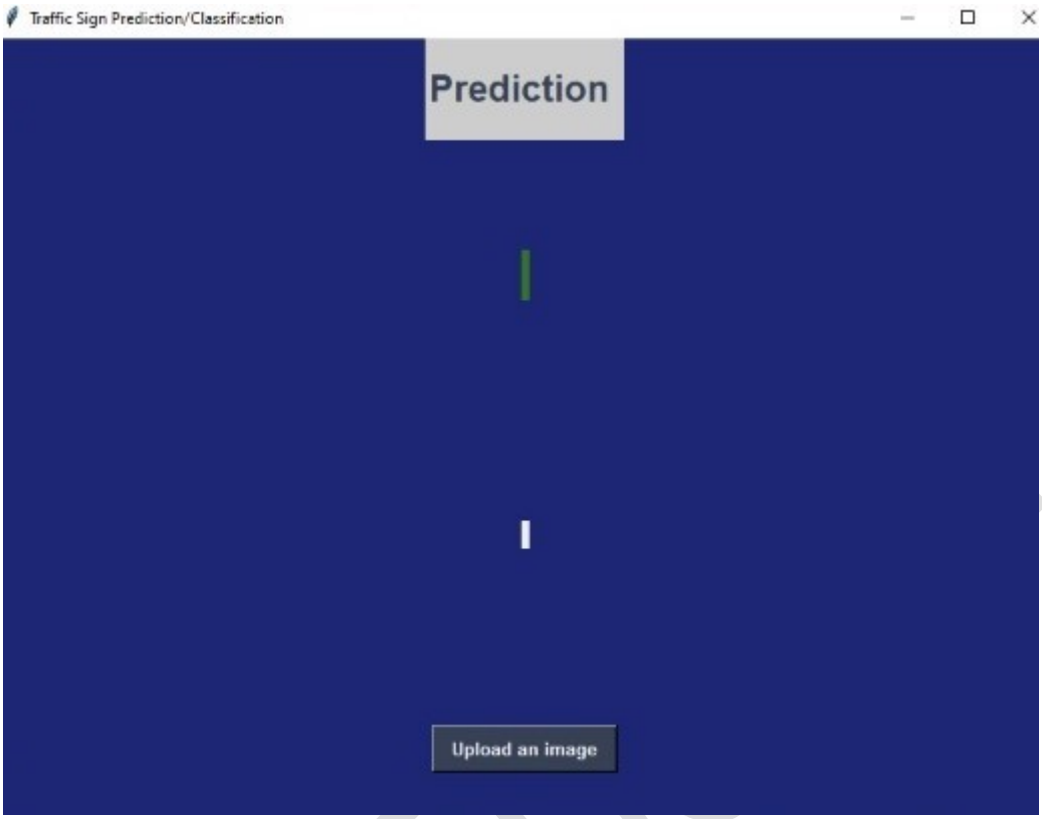
classification report

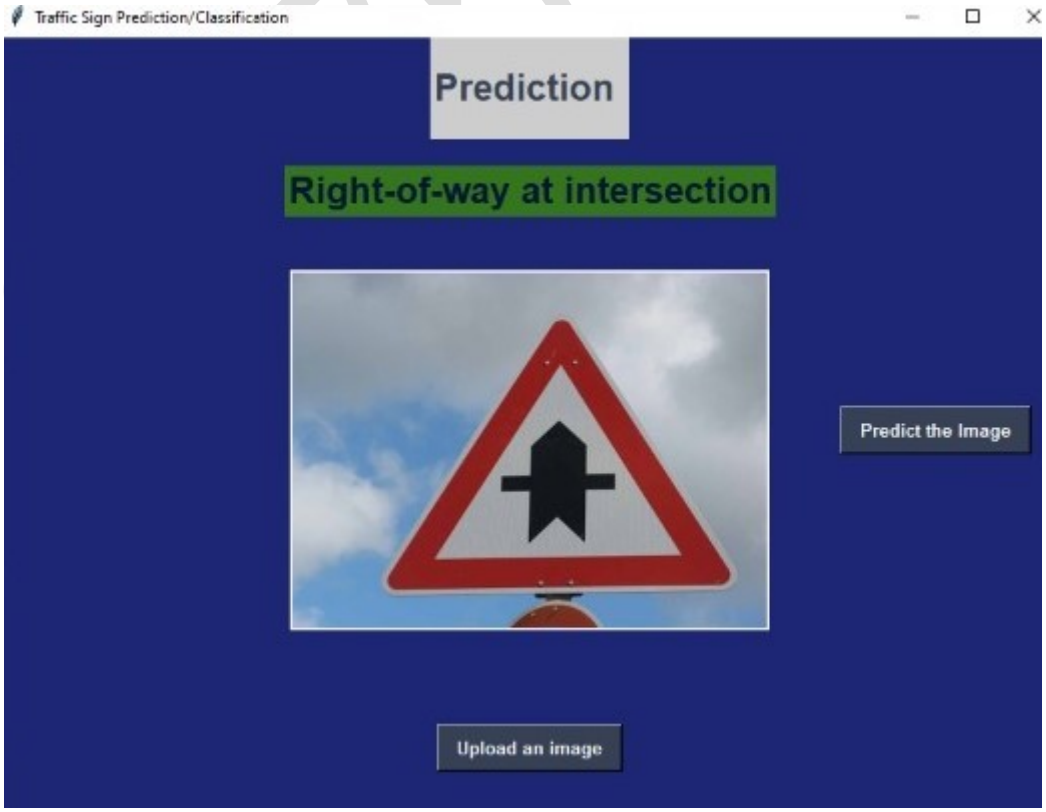
```
print(classification_report(labels,pred))

  precision    recall  f1-score   support

           0       1.00      0.98      0.99        60
           1       0.99      0.99      0.99       720
           2       0.99      0.97      0.98       750
           3       0.91      0.88      0.90       450
           4       0.99      0.96      0.98       660
           5       0.81      0.97      0.88       630
           6       0.94      0.79      0.86       150
           7       0.95      0.94      0.95       450
           8       0.97      0.92      0.94       450
           9       0.99      0.97      0.98       480
          10       0.97      0.97      0.97       660
          11       0.86      0.91      0.88       420
          12       0.98      0.93      0.96       690
          13       0.99      1.00      0.99       720
          14       0.99      0.98      0.99       270
          15       0.91      0.99      0.95       210
          16       0.89      0.99      0.93       150
          17       0.96      0.99      0.98       360
          18       0.98      0.87      0.92       390
          19       0.97      0.93      0.95        60
          20       0.96      0.94      0.95        90
          21       0.95      0.90      0.93        90
          22       1.00      0.96      0.98       120
```

```
print(classification_report(labels,pred))
          23       0.96      0.92      0.94       150
          24       0.99      0.88      0.93        90
          25       0.97      0.91      0.94       480
          26       0.87      0.84      0.86       180
          27       0.97      0.48      0.64        60
          28       0.98      0.93      0.96       150
          29       0.92      0.91      0.92        90
          30       0.71      0.80      0.75       150
          31       0.86      0.99      0.92       270
          32       0.88      0.97      0.92        60
          33       0.97      0.99      0.98       210
          34       0.98      0.99      0.99       120
          35       0.99      0.97      0.98       390
          36       0.98      0.97      0.98       120
          37       0.97      0.98      0.98        60
          38       0.95      0.99      0.97       690
          39       0.94      0.83      0.88        90
          40       0.98      0.98      0.98        90
          41       0.91      0.72      0.80        60
          42       0.80      0.96      0.87        90

    accuracy                           0.95     12630
   macro avg       0.94      0.92      0.93     12630
weighted avg       0.95      0.95      0.95     12630
```

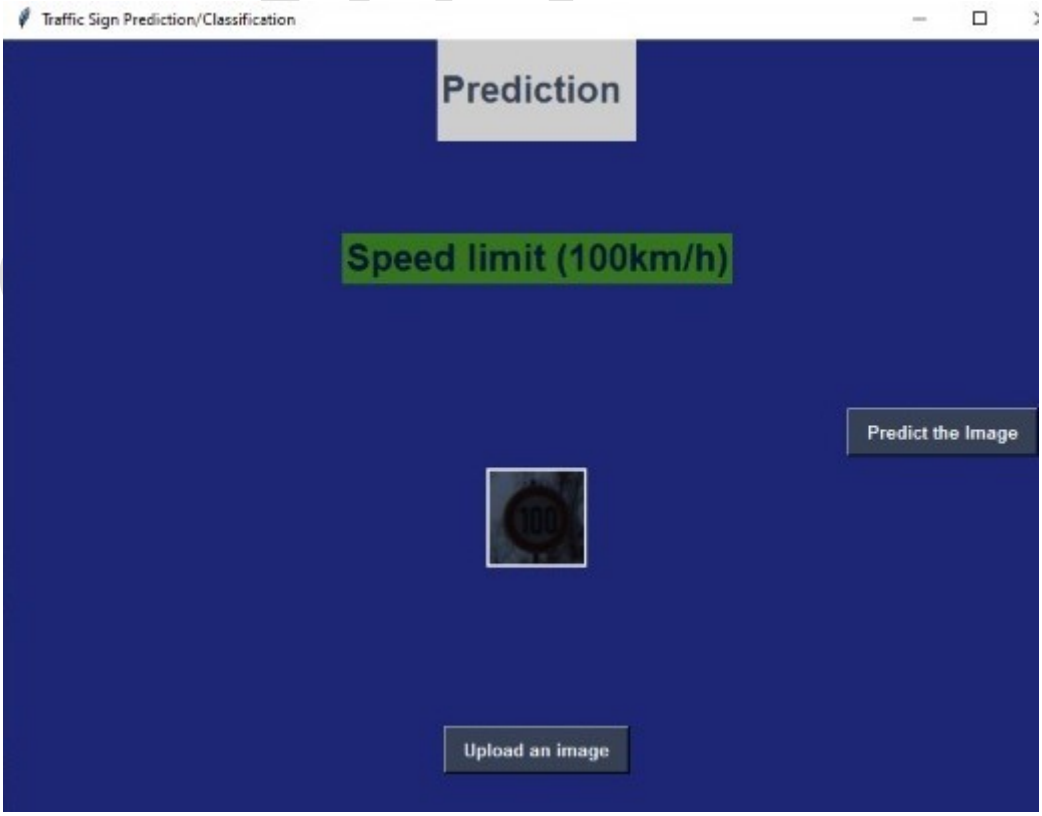
Final accuracy: 94%

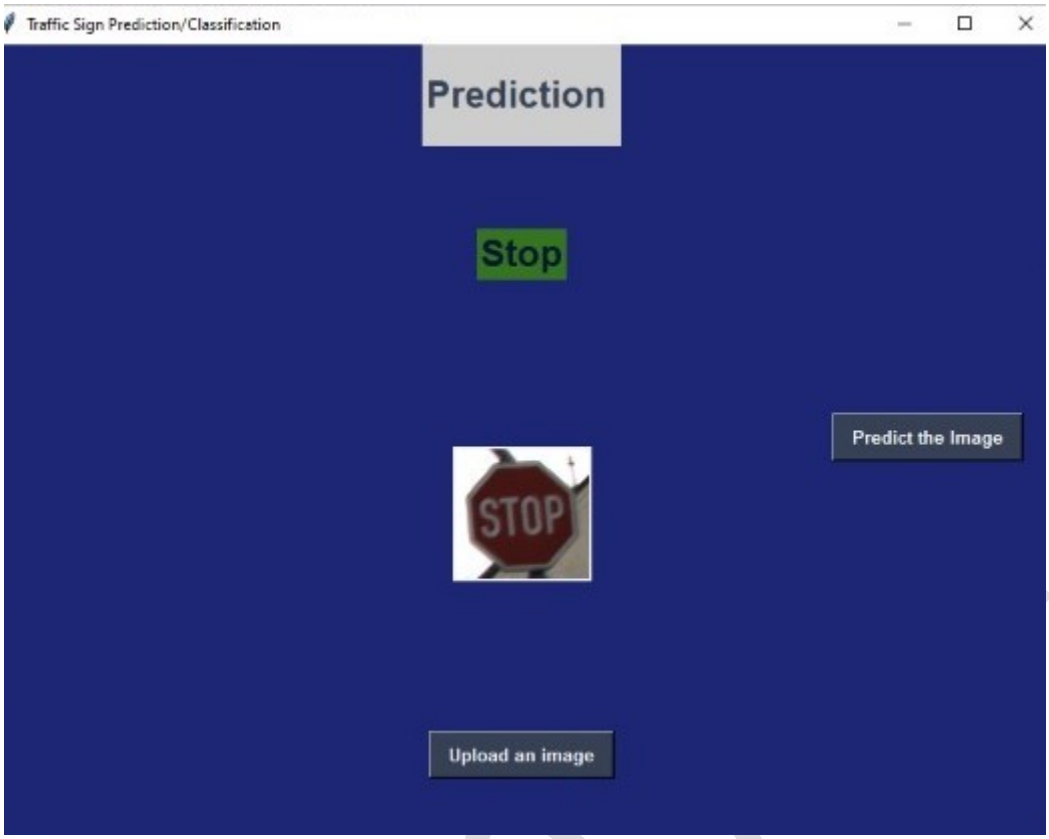
Final Output:

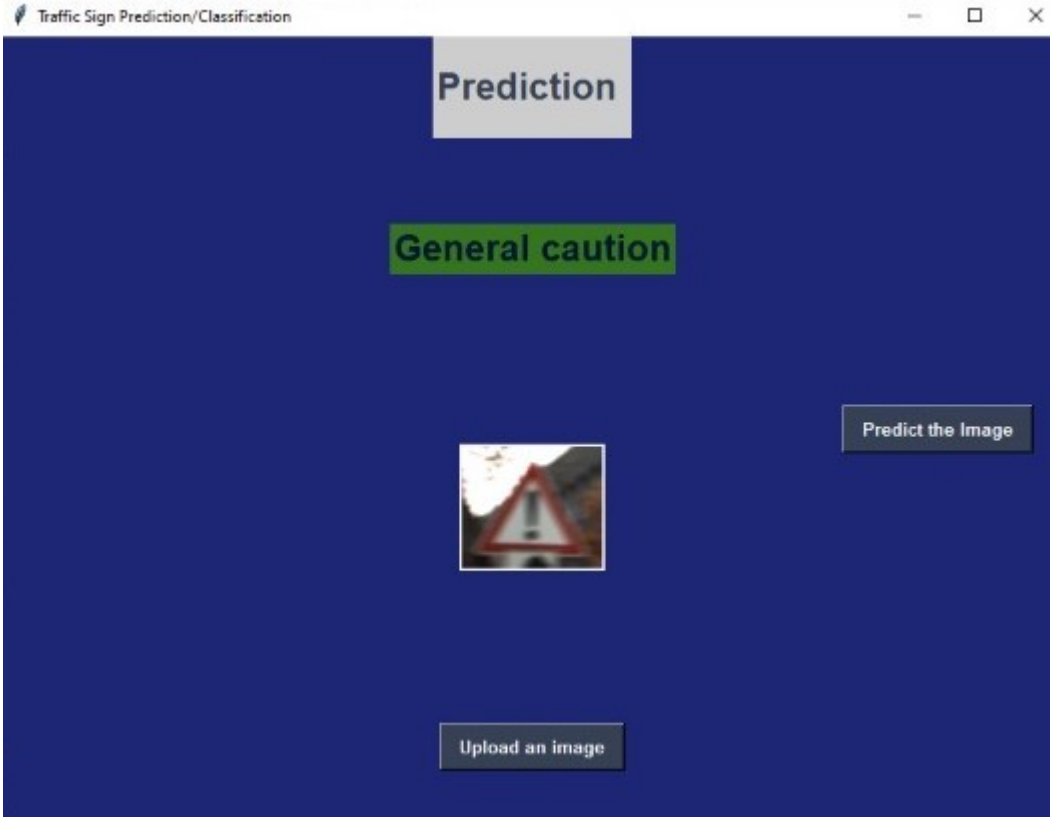


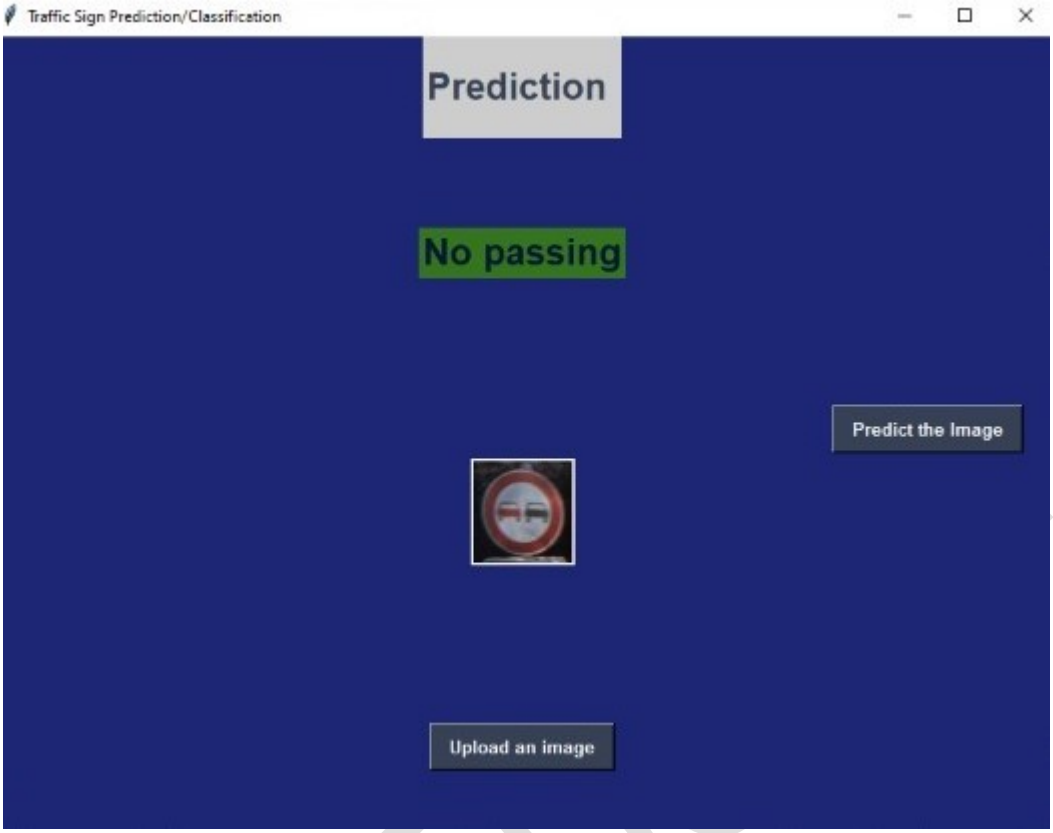



Traffic Sign Prediction/Classification
Prediction
Road work
Predict the Image
Upload an image


Traffic Sign Prediction/Classification
Prediction
Speed limit (100km/h)
Predict the Image
Upload an image


Traffic Sign Prediction/Classification
Prediction
Stop
Predict the Image
STOP
Upload an image


Traffic Sign Prediction/Classification
Prediction
General caution
Predict the Image
Upload an image


Traffic Sign Prediction/Classification
Prediction
No passing
Predict the Image
Upload an image