Traffic Signs Classification Using Machine Learning

Ankita Hora
Student, Department of Computer Science and Engineering (CSE AI-ML)
222210001@nitdelhi.ac.in
National Institute of Technology, Delhi
www.nitdelhi.ac.in

Abstract— These days, autonomous self-driving vehicles are getting much importance and are growing at a very higher pace, so with that comes the major role of traffic signs recognition and classification. Here, we will be discussing a few approaches and use cases to identify traffic signs. We will be building an ml model using Random Forest Classifier algorithm which will classify the traffic signs present in the image into different categories. With this model, we can read and understand the traffic signs which is a very important task for all autonomous vehicles. Finally, we are deploying this traffic sign classifier model using the Flask web framework in python.

Keywords—Random Forest Classifier Algorithm, Flask.

I. INTRODUCTION

Recently the number of road vehicles has increased enormously thanks to the technological achievements in the motor industry and very precisely the availability of low rates. With this remarkable growth, the number of accidents is as well an infinite raise year after year, due to different causes, which the ignorance of traffic signs is considered as a major cause of these lasts. Developing automated traffic sign recognition systems helps assist the driver in different ways to guarantee his /her safety, which preserves as well the safety of other drivers and pedestrians. These systems have one main goal: detecting and recognizing traffic signs during the driving process. With these functionalities, the system can guide and alert drivers to prevent danger. The benefits are generally focused on driver convenience, which for many is great news. As mentioned above, it should take the stress off trying to keep up with speed signs when driving somewhere new.

Also, for those who spend a lot of time on the motorway these variable speed limits are becoming all the more frequent, many systems, such as Ford's traffic sign recognition software, will identify speed limits displayed on motorway gantries.

II. LITERATURE REVIEW

Victor Ciuntu et.al in "Real-Time Traffic Sign Detection and Classification Using Machine Learning and Optical Character Recognition" proposed a method for recognizing and classifying traffic signs using a convolutional neural network and optical character recognition algorithm, especially for the speed limit signs. Here, they captured the traffic signs from real-time video taken from German, Belgian, and Illinois United States where they showed after their analysis that the average frame rate of 5FPS when running on GTX 1070 is very much sufficient for real-time image processing. With that frame rate they showed that the vehicle driving on a highway with 70mph, the neural network gets 15 chances to detect and correctly classify the traffic signs. And when driving at 25-30mph, the neural network gets 30-40 chances to perform detection and classification. Also, they achieved an accuracy of 90.76 for the classification of images which is very good [1].

AL Sangal et.al in "ICSCCC paper on Traffic Signs Classification using Convolutional Neural Networks" used CNN for image classification and SVM, segmentation for image detection. They showed that LeNet-5 architecture works well and they used the dropout feature in the neural network very effectively to reduce overfitting [2].

III. METHOD AND MATERIAL

A. Dataset

The dataset for this project is downloaded from the site named Kaggle. The dataset contains more than 50,000 images of different traffic signs. It is further classified into 58 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images.

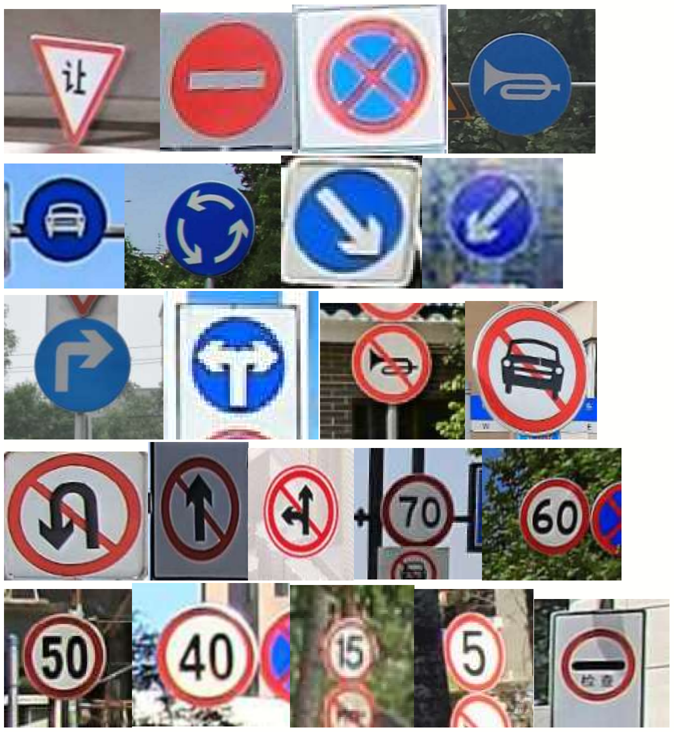
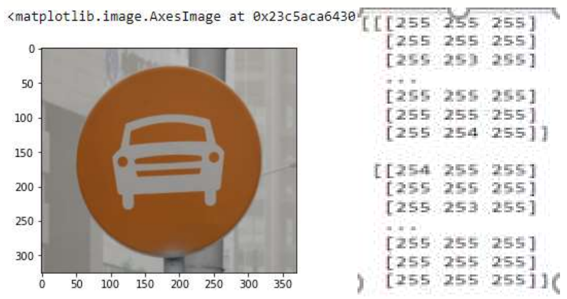


Figure 1. Images of Dataset (Different types of traffic signs.)

B. Image preprocessing

Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are images. Computers are able to perform computations on numbers and are unable to interpret images in the way that we do. We have to somehow convert the images to numbers for the computer to understand as shown in figure 2.



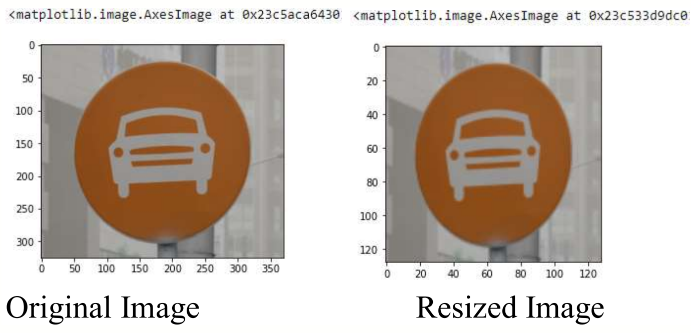
```
<matplotlib.image.AxesImage at 0x23c5aca6430
[[[255 255 255]
  [255 255 255]
  [255 253 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 254 255]]

 [[254 255 255]
  [255 255 255]
  [255 253 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]](
```

Figure 2. Image Preprocessing

The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing. The two preprocessing that we are using are: resize and flatten.

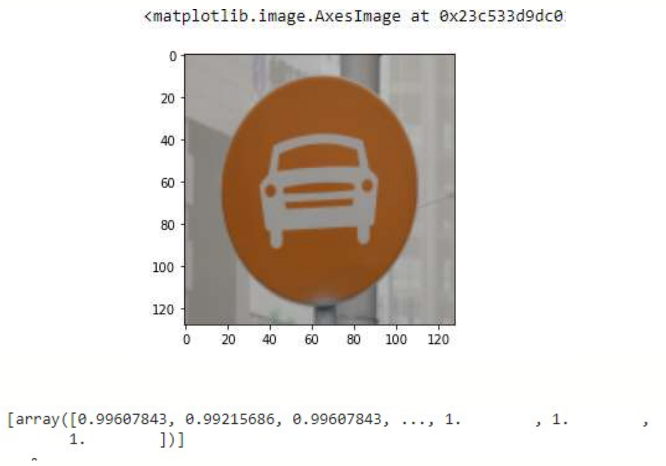
- **Resize image**
  Some images captured by a camera when fed to our model vary in size, therefore, we should establish a base size for all images by resizing them as shown in figure 3.



```
<matplotlib.image.AxesImage at 0x23c5aca6430    <matplotlib.image.AxesImage at 0x23c533d9dc0
```

Figure 3. Resizing the image

- **Flatten**
  Flattening is a technique that is used to convert multi-dimensional arrays into a 1-D array as shown in figure 4.



```
<matplotlib.image.AxesImage at 0x23c533d9dc0

[array([0.99607843, 0.99215686, 0.99607843, ..., 1.        , 1.        ,
        1.        ])]
```

Figure 4. Flattening of image

We flatten the Image Array before processing/feeding the information to our model as we will be dealing with a dataset which contain large amount of images thus flattening helps in decreasing the memory as well as reducing the time to train the model.

C. Random Forest Classifier

Random forest is a supervised learning algorithm. The "forest" it builds is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

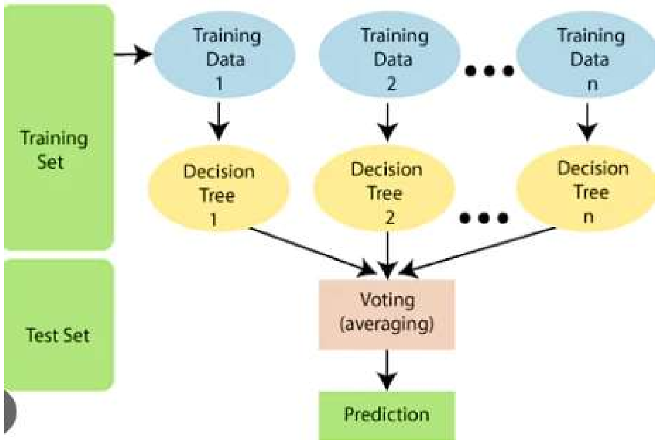


Figure 5. Random Forest Classifier

There are a lot of benefits to using the Random Forest Algorithm, but one of the main advantages is that it reduces the risk of overfitting and the required training time. Additionally, it offers a high level of accuracy. Random Forest algorithm runs efficiently in large databases and produces highly accurate predictions by estimating missing data.
The following hyperparameters are used to enhance the performance and model's predictive power or make the model faster: -

- n_estimators: Number of trees built by the algorithm before averaging the products.
- max_features: Maximum number of features random forest uses before considering splitting a node.
- mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

IV Existing System and it's solution

Existing system problem: -
Despite the advantages of traffic sign recognition, there are drawbacks. For starters, in most rural areas it is fairly common to see a traffic sign which has been engulfed by a hedge, which could mean you and your car plow into a 30mph zone way above the speed limit, endangering other motorists and pedestrians.

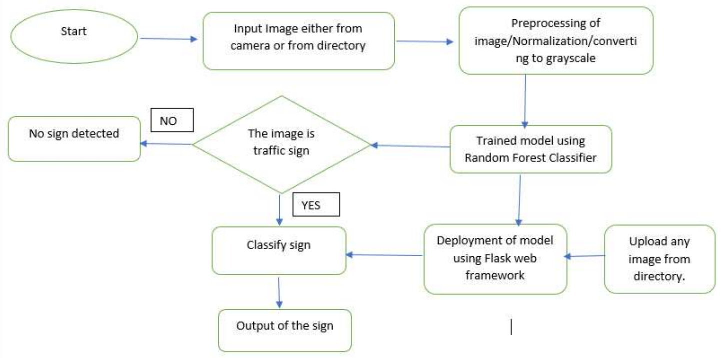

Start
Input Image either from camera or from directory
Preprocessing of image/Normalization/converting to grayscale
No sign detected
NO
The image is traffic sign
Trained model using Random Forest Classifier
YES
Classify sign
Deployment of model using Flask web framework
Upload any image from directory.
Output of the sign

Figure 6. Flow Chart

The traffic signs are categorized into 58 classes. Preprocessing on the image is done by using resizing and flattening the image. All methods were trained splitting the dataset into 80% train data and 20% test data and all data was stratified to ensure a balance between the classes. A model is created using a Random Forest classifier. Now, this model is used to classify the traffic sign when the image is given and then the accuracy is compared by doing parameter tuning. Once the model of greater accuracy is found then deploy the model which is then used to classify the traffic signs.

V. RESULT

The proposed model with different parameters is used and see that when n_estimators (no. of trees used) is equal to 1400 and min_samples_split is equal to 10 , max_depth is equal to 10 also the criterion is set to "entropy" it gives high accuracy. The graph in figure 7 represents the model having different f1_score having different n_estimators with criterion='entropy'.

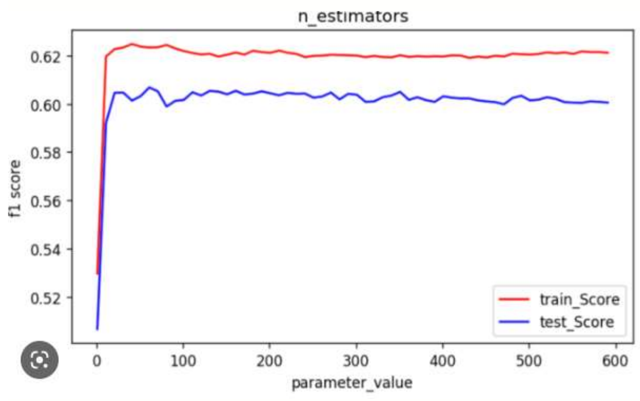
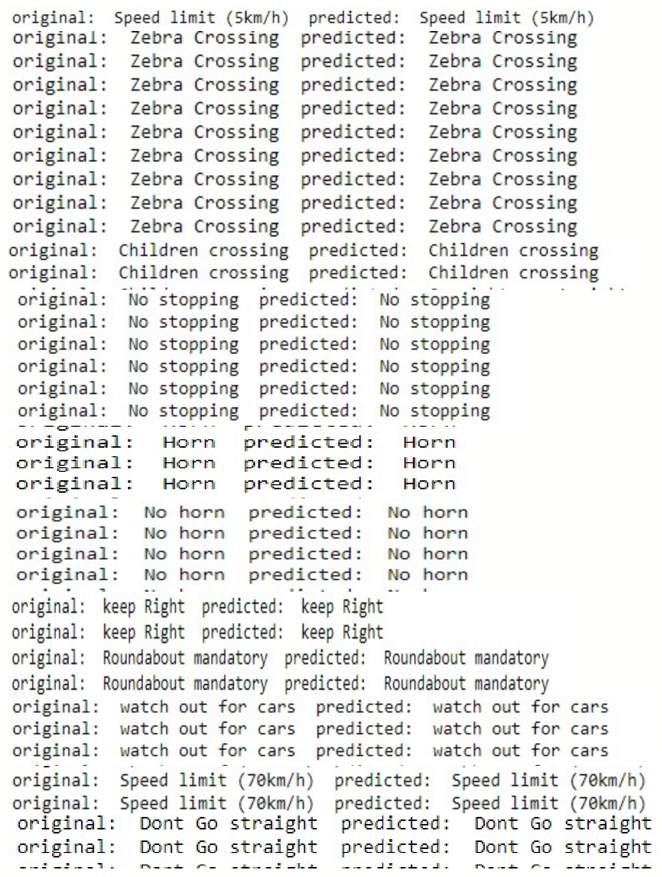


Figure 7. F1_score vs n_estimators

Proposed Solution:
As the technology becomes more widespread it’s capabilities will increase too, so it would be natural for the signs a car can identify to increase. There are also app-based traffic sign recognition systems for your Android, which could bring the technology to older vehicles

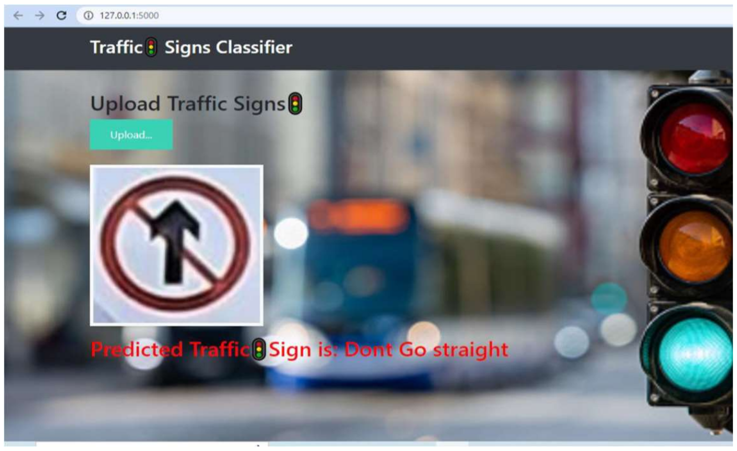
Figure 8 shows how the prediction is done when the path of the image is given to the model.



```
original:  Speed limit (5km/h)  predicted:  Speed limit (5km/h)
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Zebra Crossing  predicted:  Zebra Crossing
original:  Children crossing  predicted:  Children crossing
original:  Children crossing  predicted:  Children crossing
original:  No stopping  predicted:  No stopping
original:  No stopping  predicted:  No stopping
original:  No stopping  predicted:  No stopping
original:  No stopping  predicted:  No stopping
original:  No stopping  predicted:  No stopping
original:  No stopping  predicted:  No stopping
original:  Horn  predicted:  Horn
original:  Horn  predicted:  Horn
original:  Horn  predicted:  Horn
original:  No horn  predicted:  No horn
original:  No horn  predicted:  No horn
original:  No horn  predicted:  No horn
original:  No horn  predicted:  No horn
original:  keep Right  predicted:  keep Right
original:  keep Right  predicted:  keep Right
original:  Roundabout mandatory  predicted:  Roundabout mandatory
original:  Roundabout mandatory  predicted:  Roundabout mandatory
original:  watch out for cars  predicted:  watch out for cars
original:  watch out for cars  predicted:  watch out for cars
original:  watch out for cars  predicted:  watch out for cars
original:  Speed limit (70km/h)  predicted:  Speed limit (70km/h)
original:  Speed limit (70km/h)  predicted:  Speed limit (70km/h)
original:  Dont Go straight  predicted:  Dont Go straight
original:  Dont Go straight  predicted:  Dont Go straight
```

Figure 8. Prediction of image

After deploying the model using flask we get the output as shown in figure 9.



Predicted Traffic Sign is: Dont Go straight

Figure 9. Deployment

VI. CONCLUSION

This paper considers the implementation of the classification algorithm for the traffic sign recognition task. Combined with preprocessing and localization steps from previous works, the proposed method for traffic sign classifications shows very good results: 97% of correctly classified images of the random set of distinct traffic signs images. The proposed classification solution is implemented using the computer vision library, which focuses on extracting information from the input images or video to properly understand them and predict the visual input. It contains various image processing methods like Hidden Markov models etc.

VII. FUTURE SCOPE

Another direction for further research is to develop a real-time traffic sign recognition system that captures a video by a camera mounted on a vehicle, detects and recognizes the traffic signs in real-time and gives the result to the driver within a sufficient time frame to take the right action. The crucial issue in real-time applications is the time spent recognizing the traffic sign. This should be reduced to the 169 minimums by choosing the proper techniques for real-time applications and by optimizing the code. The methods presented in this report can be modified to fit real-time requirements.

REFERENCES

[1]. Real-Time Traffic Sign Detection and Classification Using Machine Learning and Optical Character Recognition by authors: Victor Ciuntu and Hasan Ferdowsi

[2]. ICSCCC paper on Traffic Signs Classification using Convolutional Neural Networks by Authors: Palak and AL Sangal.