

Traffic Sign Detection and Recognition Using Deep Learning

Inchara Budanur M¹, Lakshmi A², Nikitha Prasad³, Rajath A N⁴

UG Students, Department of CSE, GSSSIETW, Mysuru, India^{1,2,3}

Assistant Professor, Department of CSE, GSSSIETW, Mysuru, India⁴

Abstract: Deep learning is a type of machine learning and Artificial Intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modelling. It is extremely beneficial when we have to collect, analyse and interpret large amounts of data; deep learning makes this process faster and easier. At its simplest, deep learning can be thought of as a way to automate predictive analytics. In today's world, almost everything we do has been simplified by automated tasks. In an attempt to focus on the road while driving, drivers often miss out on signs on the side of the road, which could be dangerous for them and the people around them. This problem can be avoided if there was an efficient way to notify the driver without having them shift their focus. Traffic Sign Detection and Recognition (TSDR) plays an important role here by detecting and recognizing a sign, thus notifying the driver of any upcoming signs. This not only ensures road safety but also allows the driver to be at a little more ease while driving on tricky or new roads. Another commonly faced problem is not being able to understand the meaning of the sign. In this project image processing for the detection of a sign and an ensemble of Convolutional Neural Networks (CNN) for the recognition of the sign are used. The driver can see the upcoming sign board and the meaning of that sign with the help of this Advanced Driver Assistance Systems (ADAS) application, drivers will no longer face the problem of understanding what the sign says. This is a very useful project wherein the driver can see the sign board displayed on the screen with its meaning in the form of a text alert message.

Keywords: TSDR, Python, Image Processing, Deep Learning, Convolutional Neural Network.

I. INTRODUCTION

A. Overview

Vehicles are the primary means of transportation in our day-to-day life. Due to the increase in the number of vehicles drivers are experiencing several risks while driving and this may also lead to accidents. A vast number of accidents are happening every year all over the world. These accidents are mainly because of the driver's inability to process all the visual information that is available while driving.

According to the 'World Road Statistics' report published by the International Road Federation (IRF), Geneva, India has recorded the second highest number of road accident deaths in the world in the year 2015. A lot of research is being carried out to alert the driver about the road conditions. The purpose of the advanced driver assistance system is the automation of the vehicle systems for safety and better driving. The applications of ADAS are blind spot detection, speed limit recognition, emergency brake assist, traffic sign recognition, lane departure warning etc.

Nowadays, most of the vehicles are built with ADAS for example, BMW 7 series, BMW 5 series, Ford Focus, Ford Edge etc. In Ford Focus, the images of the road signs are captured using the camera fixed in front of the vehicle. A character recognition algorithm recognizes the speed limit sign and displays it on the dashboard of the vehicle.

Traffic signs are placed beside the roads to warn about the dangerous road conditions ahead and to provide necessary information to the driver. Sometimes, heavy traffic, weather conditions or divided attention of drivers causes a chance of missing a sign and it might lead to accidents. So, it is necessary to detect and recognize these traffic signs automatically and alert the driver about the situation.

There are three kinds of traffic signs are there in India namely,

- **Mandatory Signs:** provide information about certain laws and regulations to be followed by the driver. Generally, these signs will be circular except "STOP" and "GIVE WAY" signs.
- **Cautionary Signs:** warn the drivers about hazardous road conditions such as humps, narrow bridges, gaps in median etc. The Cautionary signs will be triangular with a red outer border and the black symbol on the white background
- **Informatory/Danger Signs:** provide the information about the route directions, destination names and distances and help the driver along the routes as shown in Figure 1




Category	Shape	Color	Example
Prohibitory	Circular	Red, Blue, White & Black	
Mandatory	Rectangular & Circular	Blue, White & Black	
Danger	Triangular	Red, White & Black	

Figure 1: Traffic signs categories according to GTSDDB

As an application of ADAS, traffic sign detection and recognition has become a challenging task since real-time applications require a fast and accurate system.

B. Proposed System

Figure 2 shows the architecture of the proposed system. The architecture shows how the system detects and classifies traffic signs using a trained model using a database. The database mainly contains the data of all existing traffic signs. The detailed working of the proposed system is explained in the subsequent section.

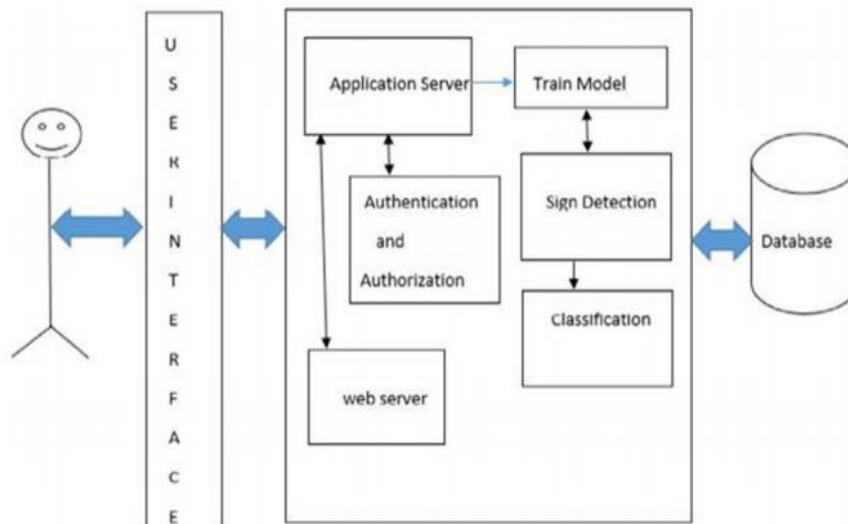


Figure 2: Proposed System Architecture

I.Data Set: A data set used for this system is a collection of traffic-related discrete items. These are all traffic signs which are used to implement this system.

II.Sign Detection: In this phase, the image obtained from the camera in the vehicle is pre-processed before the process of detection starts. General preprocessing steps involve converting the obtained RGB image into an HSV image. Once the HSV image is obtained, the next is to detect objects based on their colour followed by finding out their shape and validating the object to be a traffic sign. The first and most important thing we notice in a sign is the colour. Once we see the colour red, we realize that the board on the side of the road is a traffic sign. The second part is shape detection; this is done using the number of edges and the area. The main types of areas identified are triangle, circle and rectangle. Once the area is identified; the sign is validated.

III. Classification: The proposed system uses a convolutional neural network for the classification of signs. A convolutional neural network can have many layers, the first always being the input layer and the last the output layer anything else in between is called a hidden layer. Convolution neural network assorts traffic signs. It is trained to decode aggregation signs from natural intellectual images using the Traffic Sign Dataset. This data is prepared to maximize the model performance. Convolutional neural network scans two main parameters: stride and padding. The picture process of the convolution gives a set of new frames each frame contains information about one feature and its presence in the scanned image. The resulting frame has larger values in places where a feature is strongly visible and lower values where there are no or few such features. Afterwards, the process is repeated for each of the obtained frames a chosen number of times. The code of this system is written in Python. After the traffic signs are classified, they are converted into text format and text message is further converted to voice alert message. The final output of the system traffic signs is the voice alert message given to the driver so that he/she will be assisted in driving.

II. IMPLEMENTATION

This chapter illustrates the implementation details of the project. The goal of the implementation phase is to translate the design of the system produced during the design phase, into coded form in a given programming language. It can then be executed by a computer performing the computation specified by the design. The coding phase effects both testing and maintenance profoundly well-written code can reduce the testing and maintenance cost. Implementation is a stage in which the design is converted into working system.

Implementation part consists of several stages which contains choosing the programming language, develop environment, using various programming methods and techniques to fulfil the requirement, deciding on which library could be used within the system and how they should be integrated to it.

A. Usage of Tools

I. Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

II. Image Processing

Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modelled in the form of multidimensional systems. The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.

III. Deep Learning

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels. Importantly, a deep learning process can learn which features to optimally place in which level on its own. This does not completely eliminate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.

IV. Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural network, most commonly

applied to analyse visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are only equivariant, as opposed to invariant, to translation. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks makes them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

B. Data set

In the detection stage, the publicly available German traffic sign detection benchmark (GTSDDB) dataset was adopted for performance evaluation. In the GTSDDB, there are three classes of traffic signs: Prohibitory, mandatory and danger, and the shapes of the traffic signs are circular and triangle.

The GTSDDB dataset included 900 high-resolution natural scene images of the road environment in Germany. The sizes of those images were 1360×800 pixels, and the size of the traffic signs varied from 16×16 pixels to 128×128 pixels. These 900 pictures were divided into two parts: 600 of which were used for training, and 300 were used for testing. There were 1213 traffic signs in the GTSDDB scene pictures, and zero to six traffic signs in one traffic scene. In the test dataset of GTSDDB, there were 161 prohibitory traffic signs, 63 danger signs, and 49 mandatory signs, respectively as shown in the figure 3.

German traffic sign detection benchmark (GTSDDB) is an image classification dataset. The images are photos of traffic signs.

This benchmark has the following properties:

- Single-image, multi-class classification problem.
- More than 40 classes.
- More than 50,000 images in total.
- Large, life like database

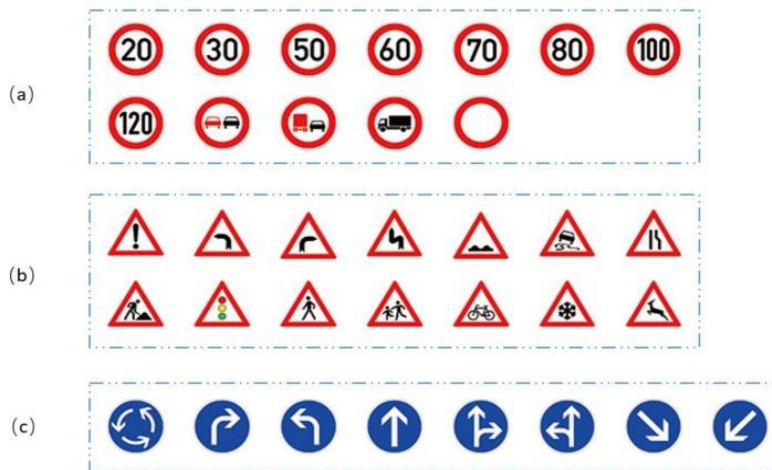


Fig 3: Examples of the subclasses of the German traffic sign detection benchmark (GTSDDB):
(a) prohibitory traffic signs (b) danger traffic signs (c) mandatory traffic signs.

Our project traffic sign recognition was based on the Germany traffic sign recognition benchmark (GTSRB), and the sizes of the traffic signs varied from 15×15 pixels to 250×250 pixels. The total number of traffic signs in GTSRB was 51,839: 12,630 images for testing and 39,209 images for training. Consistent with the GTSDDB, the traffic signs of GTSRB were divided into three main categories, including 43 subclasses.

'Extended annotations including class ids' file for the test set. Then we must organize these files in our directory for further accessing in code.

Training of images: Since the representative image consist of noises and vary a lot in illumination and size we need to equalize and normalize these images. Hence, we use functions of histogram equalization in HSV colour space and resize the images to standard size.

Once the above process is done then we pre-process all the training images and store into NumPy arrays. We also get labels of images from the paths as described in the directory structure. We also convert targets to one-hot form as is required by keras

C. *Implementation Details*

We will implement our CNN in Keras. Keras is a deep learning library written in python and allow us to do quick experimentation. After training and testing the model, we would finally do performance comparison based on accuracy with given number of parameters the model is evaluated.

To implement our system, a computer with 3.7 GHz Intel core i3 with 8GB 1600 MHz DDR3 memory is employed. We here are using anaconda python with Theano and keras and other libraries installed for running our code and getting output.

1. Dataset Acquisition: We need to download 'Images and annotation' for training and test set from the GTSRB website and extract them into a folder. Also, we need to download 'Extended annotations including class ids' file for the test set. Then we must organize these files in our directory for further accessing in code.

2. Training of images: Since the representative image consist of noises and vary a lot in illumination and size we need to equalize and normalize these images. Hence, we use functions of histogram equalization in HSV colour space and resize the images to standard size.

Once the above process is done then we pre-process all the training images and store into NumPy arrays. We also get labels of images from the paths as described in the directory structure. We also convert targets to one-hot form as is required by keras.



Figure 4: Input Image to pre-process the image and Processed Image

3. Defining Models: After the above steps being completed, we define our models. We used feed-forward network with 6 convolutional layers followed by a fully connected hidden layer. We also used dropout layers in between because Dropout regularizes the network, i.e. it prevents the network from overfitting.

All layers have ReLU activations except the Output layer. Output layer uses SoftMax activation as it has to output the probability for each of the classes.

Sequential is a Keras container for a linear stack of layers. Each of the layers in the model needs to know the input shape it should expect, but it is enough to specify input shape for the first layer of the Sequential model. Rest of the layers do automatic shape inference.

To attach a fully connected layer (aka dense layer) to a convolutional layer, we reshape/flatten the output of the conv layer. This we achieved by Flatten layer. Before training the model, we configured the model, the learning algorithm and compile it. We optimized the loss function and used optimizer and metric for optimizing and finding the accuracy of our model.

4. Training: After the above process we trained our designed model. During the training, our model will iterate over batches of the training set, each of size batch_size. For each batch, gradients will be computed and updates will be made

to the weights of the network automatically. One iteration over all the training set is referred to as an epoch. Training is usually run until the loss converges to a constant.

5. Loading test data: After the model is completely trained, we load the test data and evaluate our model on it.

6. Testing with the real environment: After being satisfied with the result we now use the above system with live camera footage and use a beep sound to alert the driver in case a traffic sign is detected.

III. EXPERIMENTAL RESULTS

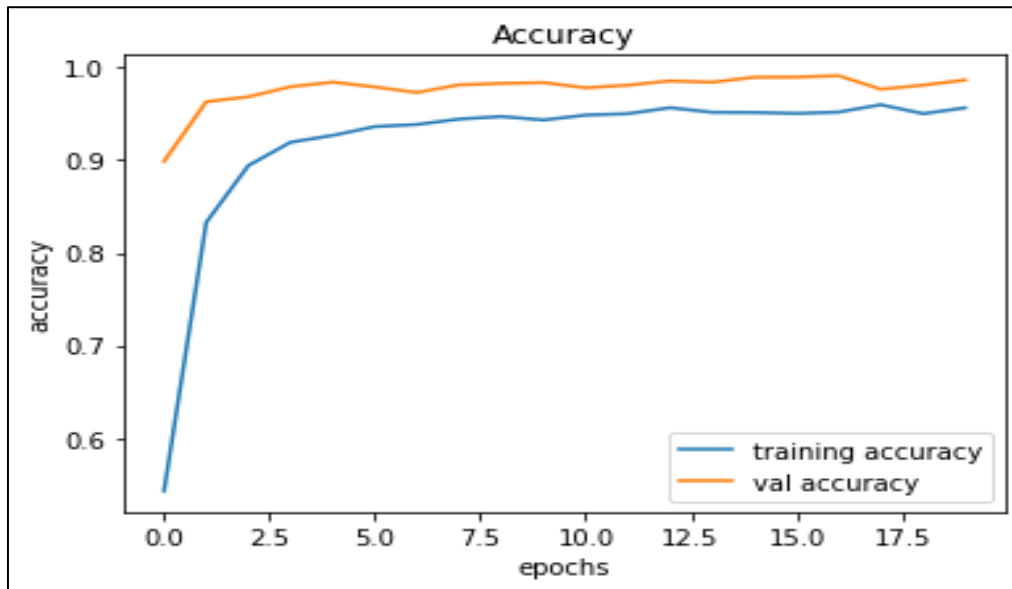


Figure 5: The accuracy variation graph shows, that the validation curve is above the training curve in accuracy variation graph, signifying that the model considered is not overfitting.

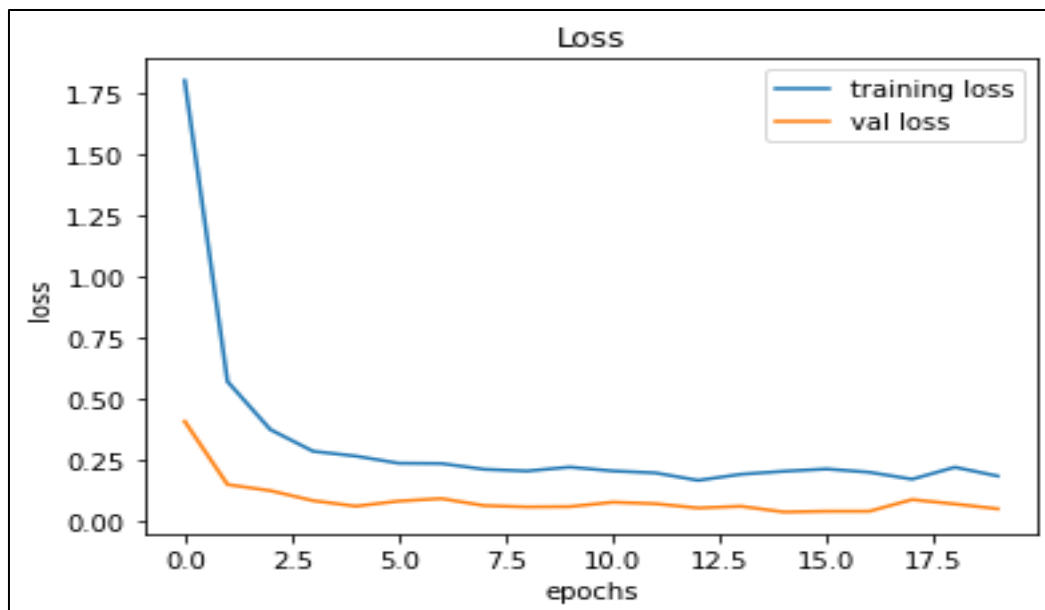


Figure 6: Loss variation graph helps show the decline in bad predictions as the epoch count increases, further solidifying the parameters used in model



Figure 7: Graphical User Interface to Know the Traffic sign

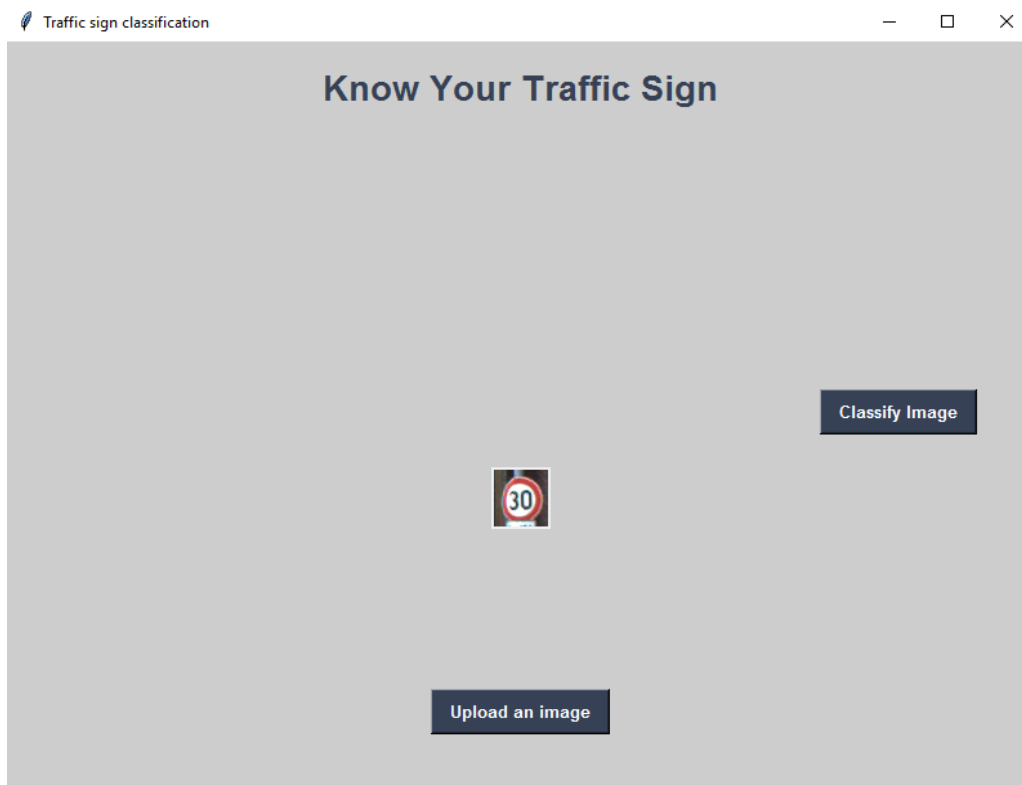


Figure 8: Uploading the Image of the Traffic Sign

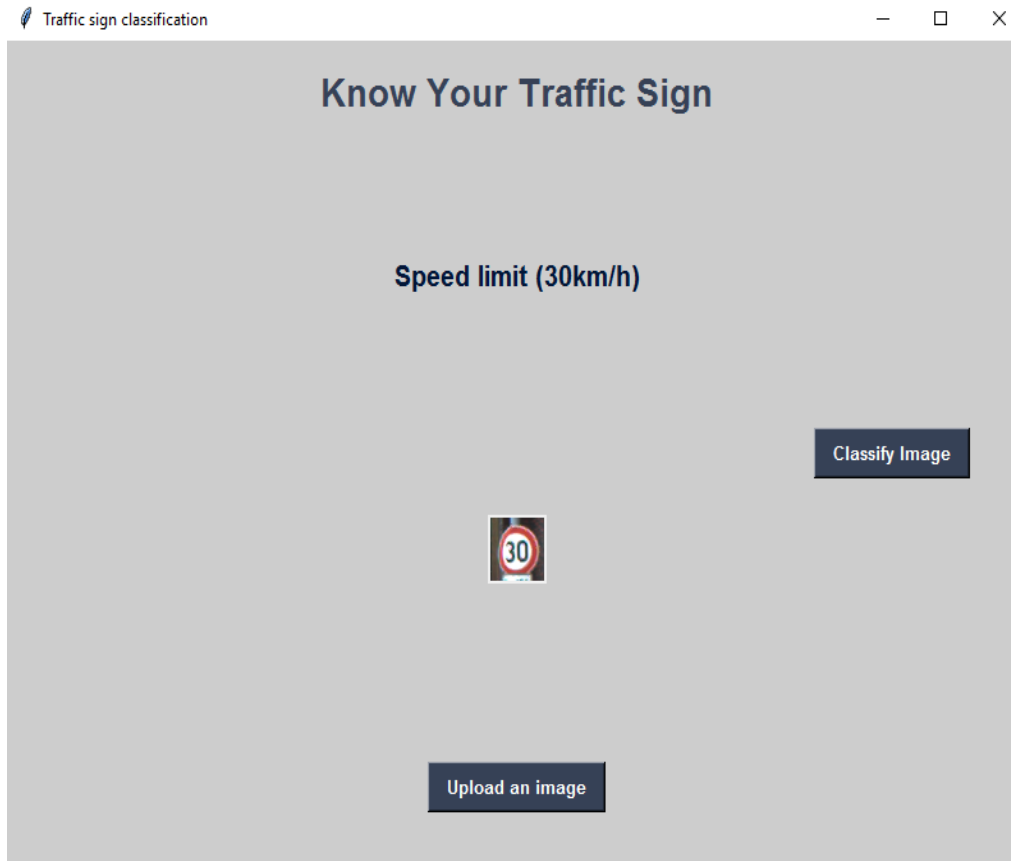


Figure 9. The meaning of the Traffic Sign is Displayed in the Text Format

IV.CONCLUSION

- In this work, we have addressed the problem of detecting and recognizing a large number of traffic-sign categories for the main purpose of automating traffic-sign inventory management.
- Due to a large number of categories with a small inter-class but a high intra-class variability, we proposed detection and recognition utilizing an approach based on the CNN detector.
- After Implementing the CNN model, we obtained an accuracy rate of 95%.
- While the model proposed in this project does bring us a step closer to achieving the ideal Advanced Driver Assistance System or even a completely driverless system in real-time, there is a lot that can be improved.
- Another important issue to consider is detection at night. If the camera is not able to capture the environment at night due to the darkness, the sign cannot be detected and classified.
- A text-to-speech module can also be added to this application. In the current application, the driver would have to read the text printed on the classified sign, but with the help of a voice module, more comfort is guaranteed.
- The overall performance could also be improved and customized with the help of more datasets from different countries.

REFERENCES

- [1] "Traffic Sign Detection and Recognition using a CNN Ensemble", Aashrith Vennelakanti, Smriti Shreya, Resmi Rajendran, Debasis Sarkar, Deepak Muddegowda, Phanish Hanagal, 2019-IEEE International Conference on Consumer Electronics (ICCE).
- [2] "Traffic Sign Detection- A New Approach and Recognition Using Convolution Neural Network", Prashengit Dhar, Md. Zainal Abedin, Tonoy Biswas, Anish Datta, 2017-Humanitarian Technology Conference (R10-HTC).
- [3] "Attention-based Neural Network for Traffic Sign Detection", Jing Zhang, Le Hui, Jianfeng Lu, Yuhua Zhu, 2018-International Conference on Pattern Recognition (ICPR).

- [4] "Traffic sign detection and recognition based on convolutional neural networks", Md Tarequl Islam, 2020- 2019 International Conference on Advances in Computing, Communication and Control (ICAC3).
- [5] "Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN", Danyah A. Alghmghama, Ghazanfar Latif, Jaafar Alghazo a, Loay Alzubaidi, 2019- Published by Elsevier B.V.
- [6] "Traffic Sign Detection and Recognition Based on Convolutional Neural Network", Ying Sun, Pingshu Ge, Dequan Liu, 2019-Chinese Automation Congress (CAC).
- [7] "Deep Learning for Large-Scale Traffic-Sign Detection and Recognition", Domen Tabernik and Danijel Sko, 2019-IEEE Transactions on Intelligent Transportation Systems.
- [8] "Automatic Traffic Sign Detection and Recognition Using SegU-Net and a Modified Tversky Loss Function with L1-Constraint", 2019-IEEE Transactions on Intelligent Transportation Systems.
- [9] "Traffic Sign Detection- A New Approach and Recognition Using Convolution Neural Network" Prashengit Dhar, Md. Zainal Abedin, Tonoy Biswas, Anish Datta, 2017 IEEE Region 10 Humanitarian Technology Conference.
- [10] "Automatic Traffic Sign Detection and Recognition Using SegU-Net and a Modified Tversky Loss Function with L1-Constraint" Uday Kamal, Thamidul Islam Tonmoy, Sowmitra Das, and Md. Kamrul Hasan, 2019-IEEE Transactions on Intelligent Transportation System.
- [11] "Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives" Chunsheng Liu, Shuang Li, Faliang Chang, Yinhai Wang, 2018-The National Key R&D Program of China under Grant.
- [12] "Automatic Traffic Sign Detection and Recognition: A Review" Swathi M, K. V. Suresh, 2017.
- [13] "Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning" Wang Canyong, 2018-International Conference on Robots & Intelligent Systems.
- [14] "Traffic signs recognition with deep learning" Djebbara Yasmina, Rebai Karima, Azouaoui Ouahiba, 2018-International Conference on Applied Smart Systems.
- [15] "Computer Vision-Based Traffic Sign Detection and Extraction: A Hybrid Approach Using GIS And Machine Learning" Zihao Wu, 2019-Electronic Theses and Dissertations.
- [16] "Traffic Sign Classification Using Computationally Efficient Convolutional Neural Networks" Carl Ekman, 2019-LiTH-ISY-EX--19/5216--SE.
- [17] "Traffic sign detection using computer vision Explorations for a driver support system" Andreas Møgelmoose 2012-Master thesis: Vision, Graphics, and Interactive Systems.