

SIGN BOARD CLASSIFICATION

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KYATHAM REVANTH REDDY (Reg. No. 37110399)

MARIDU KUSHAL KUMAR RAJA (Reg. No. 37110438)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI – 600 119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **K. REVANTH REDDY (Reg. No. 37110399) and M. KUSHAL KUMAR RAJA (Reg. No. 37110438)** who carried out the project entitled "**SIGN BOARD CLASSIFICATION**" under my supervision from **August 2020** to **March 2021**.

Internal Guide
Dr. V. Nirmalrani,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Kyatham Revanth Reddy** hereby declare that the Project Report entitled "**SIGN BOARD CLASSIFICATION**" is done by me under the guidance of **Dr. V. Nirmalrani**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. V. Nirmalrani, Associate Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Sign board classification system plays an important role and could guarantee the safety of human life on the road since it feedbacks road information to the driver in time. It's used the most in self-driving cars which are the future. This paper proposes a framework that may discover and classify differing types of traffic signs from pictures provided. The technique consists of two main modules: road sign detection, and classification and recognition. In the first step, images are sorted out through various classes and are resized based on our need. If present in predefined classes, normalized in size and then its classified. we tend to propose a Convolutional Neural Network (CNN)-based approach for detection and classifying traffic signs that is powerful against such difficult environmental conditions. Convolution Neural network is used for classification purposes. For evaluation purpose, forty-three type traffic signs such as Stop Sign, No Entry Sign, Give Way Sign, and Speed Limit Sign and so on are used. Altogether 39209 images, 31367 images are used for training purposes. 7842 images are used testing. The experimental results show the detection rate is above 90% and the accuracy of recognition is more than 97%.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OUTLINE OF PROJECT	2
	1.3 AIM AND PURPOSE	3
2	LITERATURE SURVEY	5
3	AIM AND SCOPE OF PRESENT INVESTIGATION	7
	3.1 AIM OF THE PROJECT	7
	3.1.1 OBJECTIVES	7
	3.2 SCOPE	8
	3.3 ADVANTAGES	10
	3.4 PYTHON PACKAGES	11
4	EXPERIMENTAL OR MATERIAL AND METHODS; ALGORITHMS USED	14
	4.1 SYSTEM ARCHITECTURE	14
	4.1.1 GATHERING DATA	15
	4.1.2 DATA PRE-PROCESSING	15
	4.1.3 DATA LOADING	16
	4.1.4 BUILDING A MODEL	16
	4.1.4.1 WORKING	17
	4.1.5 POOLING LAYER	18

	4.1.6 FULLY CONNECTED LAYER	18
	4.1.7 TRAINING THE MODEL	19
	4.1.8 EVALUATING THE MODEL	20
	4.1.9 PREDICTING THE RESULT	20
5	RESULT AND DISCUSSION	22
6	CONCLUSION AND FUTURE WORK	27
	6.1 CONCLUSION	27
	6.2 FUTURE WORK	27
	REFERENCES	28
	APPENDICES	30
	A. SOURCE CODE	30
	B. SCREEN SHOTS	40

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	System Architecture	14
4.2	Flow Chart	20
5.1	Splitting Classes	22
5.2	Accuracy and Detection Rate	23
5.3	Analysis	24
5.4	Model Summary	24
5.5	Plots of Loss and Accuracy	25
5.6	Sign Board Prediction	26

CHAPTER 1

INTRODUCTION

N

1.1 INTRODUCTION

Sign board Classification is a field of computer vision which is very important aspect for autonomous driver support system. The traffic signs define a visual language realized by drivers. Traffic signs carry much information necessary for successful driving - they describe up to date traffic situation, define right-of-way, prohibit or permit certain actions or directions, warn about risky factors etc. Road signs also help drivers with routing the vehicle by identifying the road-sign. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs. Naturally, autonomous vehicles must also abide by road legislation and therefore recognize and understand traffic signs.

Traffic signs have several unique features that may be used for their detection and identification. Colours and shapes are the most important attributes that assist and get better driving conditions. Moreover, the colour used in traffic sign are synchronized by each country and are often nearly simple primary colours (red, green or blue) with the exception a secondary colour of yellow. Therefore, colours can be used to classify traffic sign into specific groups. Many methods for traffic-sign recognition have been developed in the literature. However, classification of traffic signs invariantly with respect to various natural viewing conditions still remains a challenging task and still is an open research problem in traffic engineering

The detection of the traffic signs is achieved by three main modules: pre-processing, detection, and recognition. Within the pre-processing modules, the input images are pre-processed to removed noise, enhance the image. Within the detection phase, probable road sign are generated from the image. The image is segmented on the basis of pre-defined classes. The output of this stage is a

segmented image containing Regions of Interests (ROIs) which could be recognized as potential road signs.

The segmented potential regions are extracted as input in recognition stage. In the recognition stage, classification and recognition of detected signs is done by a Convolution Neural Network (CNN).

1.2 OUTLINE OF PROJECT

As we know, the proposed model for accurately predicting the traffic sign boards. Traffic sign classification is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs, etc. Being able to automatically recognize traffic signs enables us to build “smarter cars”. Self-driving cars need traffic sign recognition in order to properly parse and understand the roadway. Similarly, “driver alert” systems inside cars need to understand the roadway around them to help aid and protect drivers.

Traffic sign recognition is just one of the problems that computer vision and deep learning can solve. Like many others, I rely on Google Maps quite a bit these days to take me from point A to point B. While it helps me reach my destination with dynamic traffic updates and alternate route suggestions, I couldn't help notice how much we rely on the traffic sign infrastructure as well. Imagine a few decades ahead when all vehicles are level 5 autonomous and fully connected so they can communicate between themselves, and with the transportation infrastructure. In such a scenario, there may not be a need for many of these traffic signs. Until then however, we still have to rely upon them as long as autonomous vehicles with different levels of autonomy co-exist harmoniously with human drivers on the road. To accomplish such a task, it would mean that autonomous vehicles should have the ability to read and understand the traffic signs, and to determine the appropriate action. This project focuses on the former: To develop a neural network that reads traffic signs and classifies them correctly.

Every country has some standards set for the design of different traffic signs like U-turn, Left-turn, Right-turn, No-entry, etc. Traffic sign recognition is the process of automatically identifying which of the following class the sign belongs to. The earlier Computer Vision techniques required lots of hard work in data processing and it took a lot of time to manually extract the features of the image.

Now, deep learning techniques have come to the rescue and today we will see how to build a traffic recognition system for autonomous vehicles. The recognition of traffic signs has received increasing attention in recent years; it is even considered as a highly important feature of intelligent vehicles. Traffic signs carry substantial useful information that might be disregarded by drivers due to driving fatigue or searching for an address reasons. These drivers are also likely to pay less attention to traffic signs on driving in threatening weather. Therefore, making enhancement initiatives, like increasing driving safety along with improving automatic detection and road sign recognition system, is becoming indispensable to help decrease road death toll. These enhancements, however beneficial they may seem, meet several external non-technical challenges such as lighting variations, scale and weather conditions changes, occlusions and rotations, which may eventually decrease the traffic sign recognition systems performance.

1.3 AIM AND PURPOSE

The given datasets should be processed to enhance performance, which is accomplished by identifying the necessary features by applying one of the selection methods to eliminate the unwanted variables since each step has its unique features that help to predict the sign. A comprehensive study of detecting traffic-signs in the presence of challenging conditions that a car may face on the road is yet to be undertaken. Besides, the challenges faced during real-time image-capture on the roads may be multifarious. Even the reported models, which are able to handle illumination-variation efficiently, have not been tested against other types of challenges, such as, blur, decolorization, etc. as they are not likely to produce satisfactory results in these cases. Thus, designing a unified model that can handle all of these problems is quite challenging. In order to address these problems, we present a novel method of detecting traffic-signs completely based on deep Convolutional Neural Networks (CNNs).

The motivation for our model is twofold. First of all, due to the variety of the types and level of challenges present in the dataset, any algorithmic or statistical approach to denoising, segmentation and classification proves to be quite computationally intensive. As a result, this approach would be unsuitable for real-

time applications.

Thus, one of the reasons to consider a neural-network based detection and classification scheme is its computational efficiency during implementation. Secondly, CNNs have achieved phenomenal accuracy and efficiency in image-classification in recent years. An analysis of the data set reveals that, there are only a finite number of challenge-types and sign-types to be considered, and, the type of noise in each image is distinct and unique. Therefore, a neural-network based classification of the noise is likely to yield good results.

Advanced driver assistance systems (ADAS) are one of the fastest-growing fields in automotive electronics. ADAS technology can be based upon vision systems, active sensors technology, car data networks, etc. These devices can be utilized to extract various kinds of data from the driving environments. One of the most important difficulties that ADAS face is the understanding of the environment and guidance of the vehicles in real outdoor scenes.

Traffic signs are installed to guide, warn, and regulate traffic. They supply information to help drivers. In the real world, drivers may not always notice road signs. At night or in bad weather, traffic signs are harder to recognize correctly and the drivers are easily affected by headlights of oncoming vehicles. These situations may lead to traffic accidents and serious injuries.

A vision-based road sign detection and recognition system is thus desirable to catch the attention of a driver to avoid traffic hazards. These systems are important tasks not only for ADAS, but also for other real-world applications including urban scene understanding, automated driving, or even sign monitoring for maintenance. It can enhance safety by informing the drivers about the current state of traffic signs on the road and giving valuable information about precaution. However, many factors make the road sign recognition problem difficult such as lighting condition changes, occlusion of signs due to obstacles, deformation of signs, motion blur in video images, etc.

CHAPTER 2

LITERATURE SURVEY

Paper 1: Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning

Author: Wang, C.

Outcome of the Project:

Several methods have been practiced for the detection of traffic signs. various algorithms are developed to detect traffic signs. Wang, C in 2018 has proposed a method for the traffic sign detection. Traffic sign detection methods mainly includes color and shape-based methods. During the detection phase, the collected images are preprocessed, enhanced and segmented according to basic attributes of color and shape. It uses a VGG-16 as a frontend network structure of the SSD algorithm to detect and identify traffic signs. The VGG-16 network consists mainly of five stacked convolutional layers, three fully connected layers and a SoftMax.

Paper 2: Traffic Signs Recognition and Classification based on Deep Feature Learning

Author: Yan Lai, Nanxin Wang, Yusi Yang and Lan Lin

Outcome of the Project:

It uses a novel approach for traffic signs recognition and classification that is it uses Convolutional Neural Network and Support Vector Machine (CNN-SVM). In this video-based CNNSVM recognition framework, by introducing the YCbCr color space, we first divide the color channels, secondly employ CNN deep network for deep feature extraction.

Paper 3: Vision-based traffic sign compliance evaluation using convolutional neural network

Author: Edison A. Roxas, Joshua N. Acilo, Ryan Rhay P. Vicerra, Elmer P.

Dadios, Argel A. Bandala

Outcome of the Project:

This paper focus on the research is to implement a vision – based traffic sign

compliance analysis system that may be used in producing the first traffic sign

database. This paper presents a spatial measures and interpretations of some of the most commonly used traffic signs. It uses Alex net architecture for classification of traffic signs.

Paper 4: Detection of traffic signs in real-world images: The German traffic sign detection benchmark

Author: Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipf, and Christian Igel.

Outcome of the Project:

The pictures were cut to 1360×800 pixels as the lower part predominantly shows the front cover and is in this way not task-relevant. All images in the dataset were converted to RGB color space employing an edge-adaptive, constant-hue demosaicing method and were stored in raw PPM file format. All relevant traffic sign that are visible in the images were labeled manually. It uses mathematical approach for image processing.

Paper 5: Goal evaluation of segmentation algorithms for traffic sign recognition

Author: H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo

Outcome of the Project:

This paper presents a quantitative comparison of several segmentation methods (including new ones) that have successfully been used in traffic sign recognition. The methods presented can be classified into color-space thresholding, edge detection, and chromatic/achromatic decomposition. Although it is not possible to identify one method as being the best in all the cases, our main conclusion is that a color-space-threshold method incorporating illumination normalization constitutes a good choice. In addition, the use of an LUT with some loss of information improves speed and implies that some more lengthy methods could be used with good results. Achromatic decomposition has been a good choice since most problems of color classification arise when treating achromatic pixels due to instabilities. Moreover, white identification in signs has been improved using this decomposition. Consequently, the results presented here should be

considered as being relative between methods and in no way absolute.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM OF PROJECT

The primary aim of this proposed system i.e. “SBC” (Sign Board Classification) is to assist the driver in such a way that they can get all the necessary and detailed information of the road sign that is located in a particular area which will help the autonomous vehicle to the direction in a much faster way. The proposed system also provides information or status of the traffic sign present in the pre-defined classes.

3.1.1 Objectives

To understand the properties of road and traffic signs and their implications for image processing for the recognition task. To develop a recognizer that is invariant to in-plane transformations such as translation, rotation, and scaling based on invariant shape measures. To identify the most appropriate approach for feature extraction from road signs. To develop an appropriate road sign classification algorithm. To evaluate the performance of the aforementioned methods for robustness under different conditions of weather, lighting geometry, and sign.

The main issue of the problem in the traffic sign recognition system is not how to detect or recognize, with high recall, a traffic sign in a fixed image. It is rather about how to obtain a high precision in videos big data. To illustrate the problem of false alarms, a traffic sign recognition system installed in a smart phone, with 30 shot frames pers, was considered. If we suppose that in every 4minwe detect a sign and that—with reference to the speed of acer every sign span 2 s, this means that in the course of 1 hew will find a total of 15 signs and that every sign will displaying 60 frames and 1 true positive and 68 false alarms in 4 min by consequent. This may eventually lead to most users disabling their applications.

3.2 SCOPE

This model will help in building the strategies to prevent road accidents which will be helpful for the driver, especially for autonomous vehicle. SBC will surely help in bringing down the accident rates and gradually in future it will make the whole country free from accidents. The purpose of traffic sign detection is to find the locations and sizes of traffic signs in natural scene images. The well-defined colors and shapes are two main cues for traffic sign detection. Thus, we can divide the detection methods into two categories: color-based and shape-based. Color-based methods are usually fast and invariant to translation, rotation and scaling. As color can be easily affected by the lighting condition, the main difficulty of color-based methods is how to be invariant to different lighting conditions. These methods tend to follow a common scheme: the image is transformed into a color space and then thresholded. Some authors perform this thresholding directly in RGB (Red Green Blue) space, even if it is very sensitive to illumination changes.

To overcome this, simple formulas relating red, green and blue components are employed. For example, Scalora et al. in used different relations between the R, G and B components to segment the desired color. In the difference between R and G, and the difference between R and B channels are employed to form two stable features in traffic sign detection. Rota et al. in, used the color enhancement to extract red, blue and yellow blobs. This transform emphasizes the pixels where the given color channel is dominant over the other two in the RGB color space. In addition to RGB space, other color spaces such as YUV and HSI are also used. For example, The YUV system is considered in to detect blue rectangular signs. In a segmentation method in both Lab and HSI color spaces is used to extract candidate blobs for chromatic signs. At the same time, white signs are detected with the help of an achromatic decomposition.

Then a post-processing step is performed in order to discard non-interest regions, to connect fragmented signs, and to separate signs located at the same post. In the other hand, shape-based methods employ either hear-like features in frameworks inspired by the popular Viola-Jones detector or the orientation and

intensity of image gradients in frameworks inspired by the Generalized Hough

Transform. Many recent approaches use gradient orientation information in the detection phase, for example, Edge Orientation Histograms are computed over shape-specific subregions of the image. Gao et al. classify the candidate traffic signs by comparing their local edge orientations at arbitrary fixation points with those of the templates.

In the Regions of Interest (ROI) obtained from color-based segmentation are classified using the HOG feature. To integrate color information in the HOG descriptor, Creusen et al. Concatenate the HOG descriptors calculated on each of the color channels. The advantages of this feature are its scale invariance, the local contrast normalization, the coarse spatial sampling and the fine weighted orientation binning. The purpose of traffic sign recognition is to classify the detected traffic signs to their specific sub-classes. Regarding the recognition problem, it is common to use some features with machine learning algorithms. Maldonado et al. in utilized different one-vs-all SVMs with Gaussian kernel for each color and shape classification to recognize signs.

In SVMs are used with HOG features to carry out classification on candidate regions provided by the interest region detectors. It with stand great appearance variations thanks to the robustness of local features, which typically occur in outdoor data, especially dramatic illumination and scale changes. Zaklouta uses different sized HOG features, and adopts random forest-based classification to achieve high detection accuracy. Tang proposes an efficient method of traffic sign recognition using complementary features to reduce the computation complexity of traffic sign detection, and then uses the SVM to implement the traffic sign classification. The complementary features used in Tang include HOG and LBP. Convolutional Neural Network (CNN) is another method used for traffic sign classification. It is proved in that the performance of CNN on traffic sign classification outperforms the human performance. In, a CNN together with a Multi-Layer Perception (MLP), which is trained on HOG features, was used. In, a CNN with multi-scale features by using layer-skipping connection is presented. In, the authors suggest a hinge loss stochastic gradient descent method to train convolutional neural networks. The method yields to high accuracy rates. However, a high computing cost is paid to train the data when using CNNs. In general, the

quality of the results obtained by any study on TSR varies from one research group

to another. It is very difficult to decide which approach gives better overall results, mainly due to the lack of a standard database of road images.

It is not possible to know, for example, how well the systems respond to changes in illumination of the images since in the different studies it is usually not specified whether images with low illumination have been used in the experiments. Another disadvantage of the lack of a standardized database of road images is that some studies are based on a small set of images since the compilation of a set of road scene images is a very time-consuming task. The problem with working with such small data sets is that it is difficult to evaluate the reliability of the results.

3.3 ADVANTAGES

This method has several advantages over existing state of the art methods in terms of performance, scalability and memory requirement. Recent high performed method Committee of CNNs have used 25 networks with 3 convolutional layers and 2 fully connected layers along with manual data augmentation. On original dataset they have modified each image using translation, rotation etc., to get five modified version of that image. Committee of CNNs end up with total around 90. Million parameters whereas, in our method, we have around 10.5 Million parameters. Overall accuracy comparisons with different high performing approaches are shown. We have also reported the accuracy of our deep networks with the Google inception module, which is slightly lower than the accuracy obtained using our modified inception module. GTSRB dataset composed of mainly 6 high-level groups. Classification of images belong to different groups is easier than that of same group images. We have reported the accuracy obtained for each group. Also, comparisons with other state-of-the-arts methods are presented.

- Time saving as it is detected before the traffic sign occurs.
- This system helps the driver to recognize traffic sign even in blind spots while driving.
- It provides accurate information even in non-friendly environment conditions.
- It is user friendly as the sign is displayed on the dashboard.

3.4 PYTHON PACKAGES

NUMPY

NumPy (pronounced (NUM-Py) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy targets the Python reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to matlab since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, matlab boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more matlab-like functionality and Matplotlib is a plotting package that provides matlab-like plotting functionality. Internally, both matlab and numpy rely on blas and LAPACK for efficient linear algebra computations.

PANDAS

Pandas is a Python library. Pandas is used to analyze data. Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008. Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable

and relevant. Relevant data is very important in data science. Pandas gives you

answers about the data. Like: Is there a correlation between two or more columns? What is average value Max value Min value Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

SCIKIT-LEARN

scikit-learn is largely written in python, and uses numpy extensively for high-performance linear algebra and array operations. furthermore, some core algorithms are written in python to improve performance. support vector machines are implemented by a python wrapper around libsvm; logistic regression and linear support vector machines by a similar wrapper around liblinear. in such cases, extending these methods with python may not be possible. scikit-learn integrates well with many other python libraries, such as matplotlib and plotly for plotting, numpy for array vectorization, panda's data frames, scipy, and many more. scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the python programming language. it features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the python numerical and scientific libraries numpy and scipy.

MATPLOTLIB

Matplotlib is a plotting library for the python programming language and its numerical mathematics extension numpy. it provides an object-oriented api for embedding plots into applications using general-purpose gui toolkits like tkinter, wxpython, qt, or gtk+. there is also a procedural "pylab" interface based on a state machine (like openpy), designed to closely resemble that of matlab, though its use is discouraged. scipy makes use of matplotlib 2.0.x supports python versions 2.7 through 3.6. python 3 support started with matplotlib

1.2. matplotlib 1.4 is the last version to support python 2.6. matplotlib has pledged not to support python 2 past 2020 by signing the python 3 statement pyplot is a matplotlib module which provides a matlab-like interface. matplotlib is designed to be as usable as matlab, with the ability to use python, and the advantage of being

free and open-source.

KERAS

Keras is a high-level neural networks API, capable of running on top of TensorFlow, Theano, and CNTK. It enables fast experimentation through a high level, user-friendly, modular and extensible API. Keras can also be run on both CPU and GPU. Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.” Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

CV2

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

PIL

Pillow is an image processing library that has been forked from PIL (Python Image Library). Although advanced image processing (face recognition, optical flow, etc.) like OpenCV cannot be performed, simple image processing such as resizing (scaling), rotation, and trimming (partial cutout) can be performed. Because Pillow is simpler and easier to understand than OpenCV, it is better to use it depending on the purpose. Use the Python Imaging Library (PIL) to add image processing capabilities to your Vizard scripts. You can process an image and then load it as a texture within Vizard. Cutting, pasting, and merging images.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 SYSTEM ARCHITECTURE

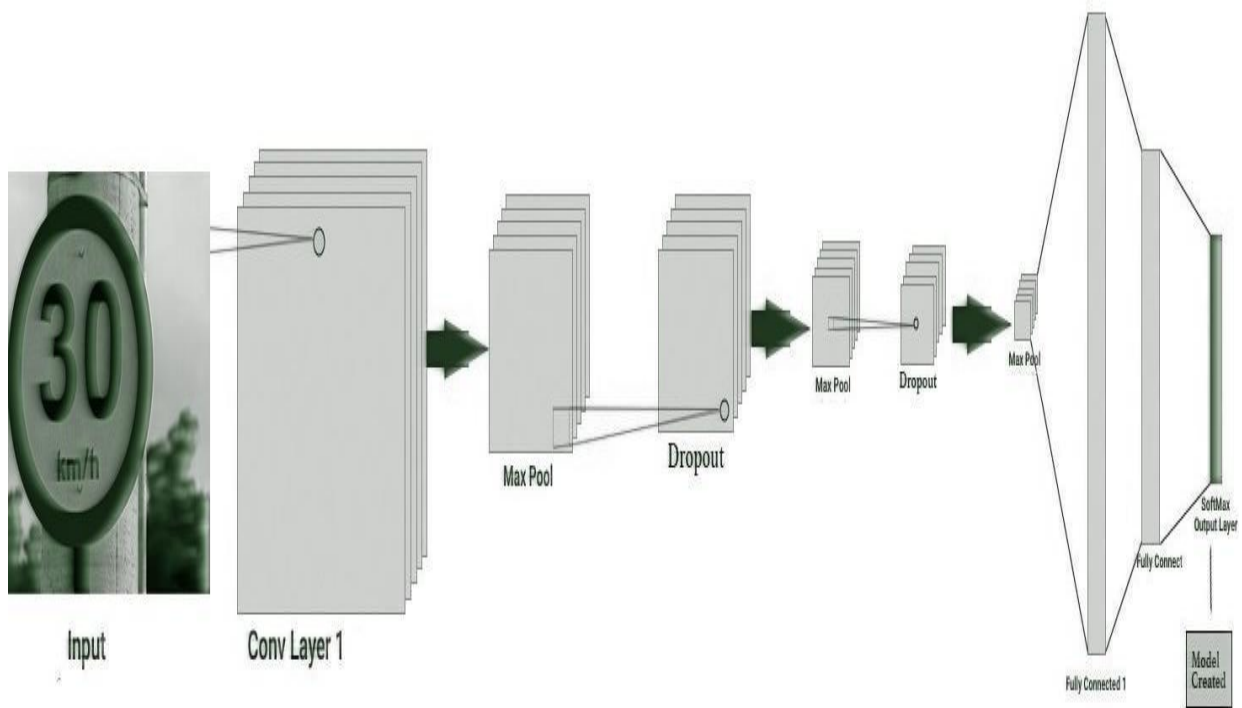


Fig 4.1 System Architecture

Traffic signs may be divided into different categories according to function, and in each category, they may be further divided into subclasses with similar generic shape and appearance but different details. This suggests traffic-sign recognition should be carried out as a two-phase task: detection followed by classification. The detection step uses shared information to suggest bounding boxes that may contain traffic-signs in a specific category, while the classification step uses differences to determine which specific kind of sign is present (if any). (We note that the words 'detection' and 'classification' have different meanings in the general object recognition community where, as exemplified by the Image Net competition, classification means giving an image a label rather than an object, and detection means finding the bounding box of an object in a specific category.

4.1.1 Gather Data

The German traffic signs detection dataset is used here. The dataset consists of 39209 images with 43 different classes. The images are distributed unevenly between those classes and hence the model may predict some classes more accurately than other classes.

We can populate the dataset with various image modifying techniques such as rotation, colour distortion or blurring the image. We will be training the model on the original dataset and will see the accuracy of the model. Then we'll be adding more data and making each class even and check the model's accuracy.

4.1.2 Data Pre-Processing

One of the limitations of the CNN model is that they cannot be trained on a different dimension of images. So, it is mandatory to have same dimension images in the dataset. We'll check the dimension of all the images of the dataset so that we can process the images into having similar dimensions. In this dataset, the images have a very dynamic range of dimensions from $16 \times 16 \times 3$ to $128 \times 128 \times 3$ hence cannot be passed directly to the Conv Net model. We need to compress or interpolate the images to a single dimension. Not, to compress much of the data and not to stretch the image too much we need to decide the dimension which is in between and keep the image data mostly accurate. I've decided to use dimension $64 \times 64 \times 3$. We will transform the image into the given dimension using OpenCV package. cv2 is a package of OpenCV. resize method transforms the image into the given dimension. Here, we're transforming an image into the 64×64 dimension. Interpolation will define what type of technique you want to use for stretching or for compressing the images. OpenCV provides 5 types of interpolation techniques based on the method they use to evaluate the pixel values of the resulting image. The techniques are INTER_AREA, INTER_NEAREST, INTER_LINEAR, INTER_CUBIC, INTER_LANCZOS4. We'll be using INTER_AREA interpolation technique it's more preferred for image decimation but for extrapolation technique it's similar as INTER_NEAREST. We could have used INTER_CUBIC but it requires high computation power so will be not using it.

4.1.3 Data Loading

Above we learned how we'll pre-process the images. Now, we'll load the dataset along with converting them in the decided dimension.

The dataset consists of 43 classes total. In other words, 43 different types of traffic signs are present in that dataset and each sign has its own folder consisting of images in different sizes and clarity. Total 39209 number of images are present in the dataset.

We can plot the histogram for number of images present for different traffic signs. `ClassId` is the unique id given for each unique traffic sign.

As, we can see from the graph that the dataset does not contain equal number of images for each class and hence, the model may be biased in detecting some traffic signs more accurately than other.

We can make the dataset consistent by altering the images using rotation or distortion techniques but we'll do this some other time.

As the dataset is divided into multiple folders and the naming of images is not consistent, we'll load all the images by converting them in $(64*64*3)$ dimension into one list image and the traffic sign it resembles into another list output. We'll be reading the images using `misread`. The dataset is loaded and now we need to divide it into training and testing set. And also, in validation set. But if we divide directly then the model will not be get trained all the traffic signs as the dataset is not randomized. So, first we'll randomize the dataset. We can see that I've converted the output array to categorical output as the model will return in such a way. Now, splitting the dataset. We'll split the dataset in 60:20:20 ratio as training, validation, test dataset respectively.

4.1.4 Building the Model

Convolutional neural networks are also a sequence of layers. Each layer transforms an input image to an output image with some differentiable function that may or may not have parameters. The Conv Net architecture consists of three types of layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. INPUT layer would hold the input image as a 3-D array of pixel values.

CONV layer will compute the dot product between the kernel and sub-array of an input image same size as a kernel. Then it'll sum all the values resulted from the dot product and this will be the single pixel value of an output image. This process is repeated until the whole input image is covered and for all the kernels.

RELU layer will apply an activation function $\max(0, x)$ on all the pixel values of an output image.

POOL layer will perform down sampling along the width and height of an image resulting in reducing the dimension of an image.

FC (Fully-Connected) layer will compute the class score for each of the classification category.

In this way, Conv Nets transform the original image layer by layer from the original pixel values to the final class scores. RELU and POOL layer implement the constant function and no variables are to be trained at this layer. Parameters at FC and CONV layer will be trained using gradient descent optimizer.

4.1.4.1 Working

The input 3-D volume is passed to this layer. The dimension would be $H \times W \times C$. H, W, and C represent height, width, and the number of channels respectively.

There can be K filter used where K represents the depth of an output volume. The dimension of all the K filters is the same which is $f \times f \times C$. f is the filter size and C is the number of channels input image has.

If we have configured padding then will add padding to the input volume. If padding is equal to same then will add one row and one column at each side of the dimension and the value would be zero. Padding is applicable only along the height and width of an input dimension and is applied along each layer.

After padding, the computation begins. Now we'll slide our filter starting from the top-left corner. The corresponding values of filter and input volume are multiplied and then the summation of all the multiplied value takes place. Now the filter is slide horizontally taking stride number of steps in each slide. So, if stride is 2 we'll slide 2 columns horizontally. The same process is repeated vertically until

the whole image is covered.

After getting all the values from the filter computation they are passed through RELU activation which is $\max(0, x)$. In this, all the negative values obtained are replaced by zero as negative values have no significance in the pixel.

Step 4 & 5 generates just one layer of an output volume that is the 3-D input volume is transformed into a 2-D volume.

Now, step 4 & 5 gets repeated for K filters. And the output of each filter is stacked above one another and hence the depth of an output image is of dimension k.

Now, to calculate the dimension of an output volume we require all the hyperparameters of a convolutional layer. All the filters used at this layer needs to be trained and are initialized with random small numbers.

4.1.5 Pooling Layer

This layer is used for reducing the dimension of an input volume. This layer does not reduce the depth of an input. This layer can be used to reduce the spatial size so that the computational power required to process the image is reduced. Pooling layer does not lose the important property of an image. Pooling layer extracts the most dominant information and hence maintains the process of effective training of the model.

Two types of pooling are used: Max Pooling and Average Pooling.

In max pooling, the maximum value present in a selected kernel is retained and all the other values are discarded. While on average pooling the average of all the values present in a kernel selected is stored. Pooling also acts as a noise suppressant. But max pooling performs better than average pooling and hence, it is more frequently used.

4.1.6 Fully-Connected Layer (FC-Layer)

This layer is used for the classification of the complex features extracted from previous layers. This layer is the same as the neural networks in which each neuron is connected to all the neurons on a consecutive layer. The final output is calculated using SoftMax which gives the probability of each class for the given features.

The working of FC-layer is similar to the deep neural networks used for classification.

To pass an input image to the FC-layer we need to flatten out the image so that all the pixel values are arranged in one column. Now, this flattened feature is passed to the FC-layer. All the weights assigned in FC-layer are initialized by small random number and are trained using back-propagation algorithm.

Now, we've learned about all the types of layers used in Conv Nets and hence, we can now apply this knowledge for recognizing hand-written digits from images. We'll be using Keras for implementing Conv Net. Now we'll create a model with 6 Convolutional layers, 3 max-pooling layers, and FC-layer.

4.1.7 Training the Model

I have generated datasets for training my model using augmentation pipeline mentioned earlier. This dataset simply contains 20 times more data than the original one – e.g. for each training example we generate 19 additional examples by jittering original image, with augmentation intensity = 0.75. This dataset is balanced across classes and has 20000 examples for each class. These 20000 contain original training dataset, as well as jittered images from the original training set (with augmentation intensity = 0.75) to complete number of examples for each class to 20000 images. It is worth noting that training on extended dataset may not be the best idea, as some classes remain significantly less represented than the others there. Training a model with this dataset would make it biased towards predicting overrepresented classes. However, in our case we are trying to score highest accuracy on supplied test dataset, which (probably) follows the same classes distribution. I then use 25% of these augmented datasets for validation while training in 2 stages. These two training stages could easily get you past 99% accuracy on the test set. You can, however, improve model performance even further by re-generating balanced dataset with slightly decreased augmentation intensity. After training the model for 11 epochs we will save the model in a traffic_recognition.h5 file.

4.1.8 Evaluating the Model

To test our model, we have a test folder that contains around 12,000 images. The test.csv file contains the path of the image along with the label of the class. Pandas is a great library to extract path and label from the CSV file and then with the help of sklearn accuracy_score () function, we can compare the real values with the predicted values of our model.

4.1.9 Predicting the Result

Finally, this model provides mildly interesting predictions for types of signs it wasn't trained for the input image of any format is taken into the model for evaluation purpose. You can predict the class for each image and can verify how the model works. The method flow is shown in the figure below.

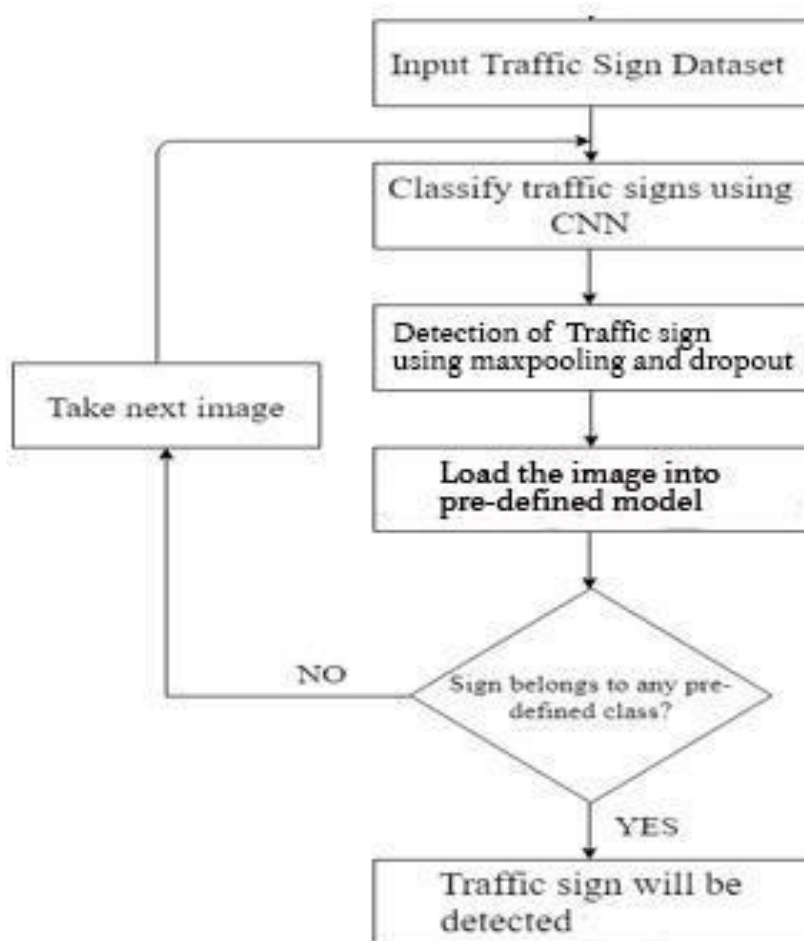


Fig 4.2 Flow Chart

Data is taken from Kaggle open source repository. It was a collection of signs which are random images irrespective of size and orientation. The input path is set to the location where the images resides. Now images which are presented as lists are converted into arrays for categorization. Then splitting of images to train dataset and test dataset. Converting the labels into one hot encoding and building a CNN model takes place. They are taken into consideration there by maxpooling and dropping out the non-required elements. Then compiling the model with train and test data takes place for maximizing the results. After compiling testing accuracy on test dataset is taken into consideration then the input images are given to the model to identify then if its present in the predefined class the results are more accurate if it's not present it tries to test from the model and provides the output sign.

CHAPTER 5

RESULTS AND DISCUSSION

First, I will import the dependencies, that will make this program a little easier to write. I'm importing the machine learning library sklearn, numpy, and pandas. I will load the Sign Board Data Set from sklearn. Datasets and print it after storing it into the variable dataset by using a function/method called load data (). but first I will import the library sklearn. Datasets. I will split the data into independent variables (X's) and dependent variable (Y) data sets while training and testing. Split the data into 67% training and 33% testing data, and then train the model with the training data set that contains the independent variables.

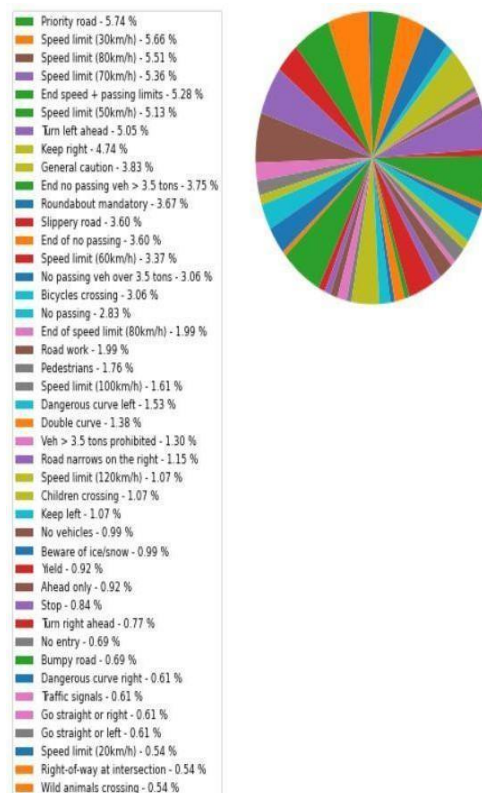


Fig 5.1 Splitting Classes

From fig 5.1 We will know the upper left, upper right, right, left, lower right, lower left, center, center left, center right. From the above diagram these functions are used to specify the coordinates of the legends. And attributes represent the columns that the legend has.

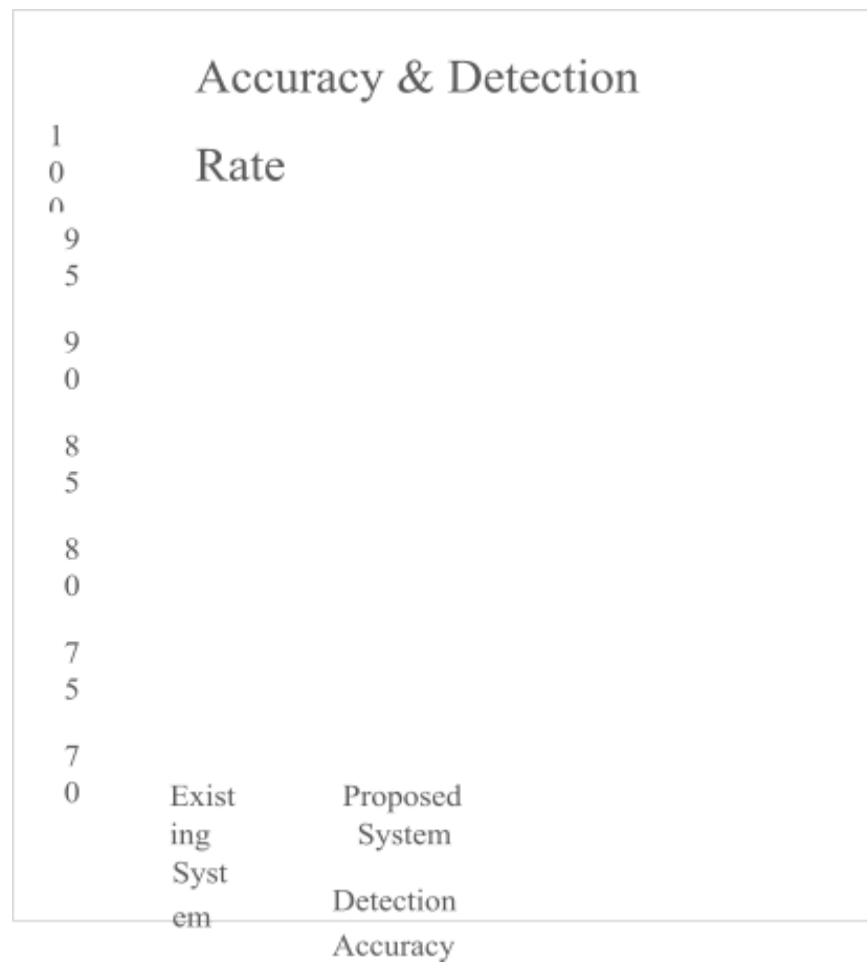


Fig 5.2 Accuracy and Detection rate

From fig 5.2 Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. And the existing model is compared with proposed model and gives the best detection.



Fig 5.3 Analysis

From fig 5.3 want to know what was the analysis for that test data set, so I can see it like user friendly to the screen, just to make it a little easier to compare data (Training and Testing).

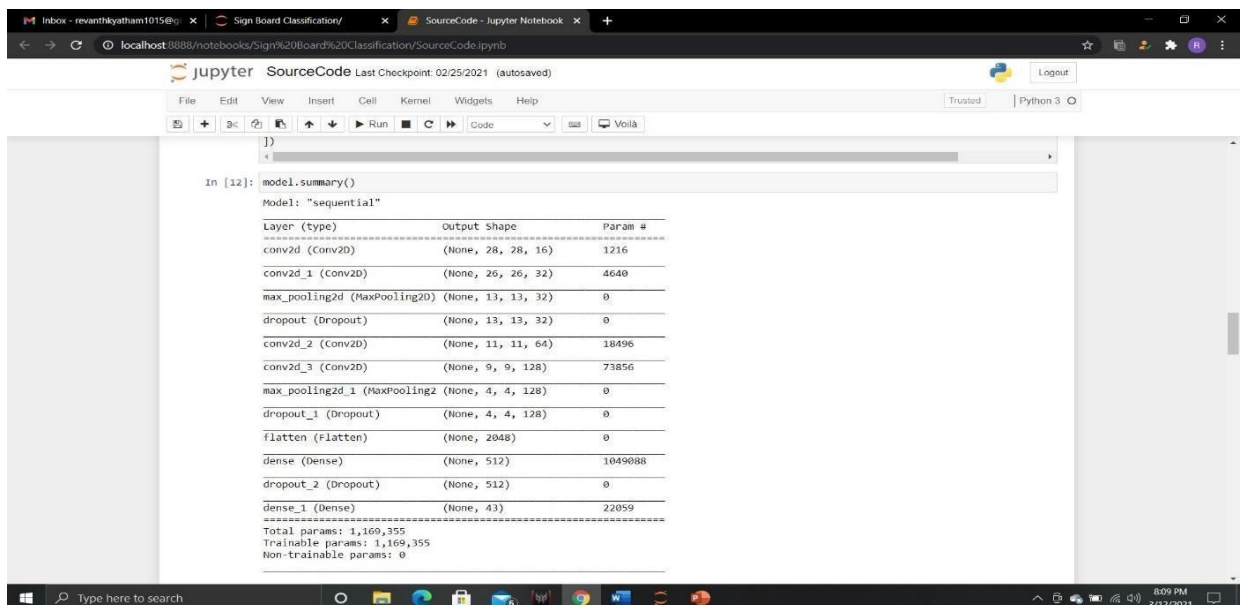


Fig 5.4 Model Summary

From fig 5.4 Model summary here displays all the methods like maxpooling, dropout, convolution 2D network, flattern, dense and so., these methods are used in the model.

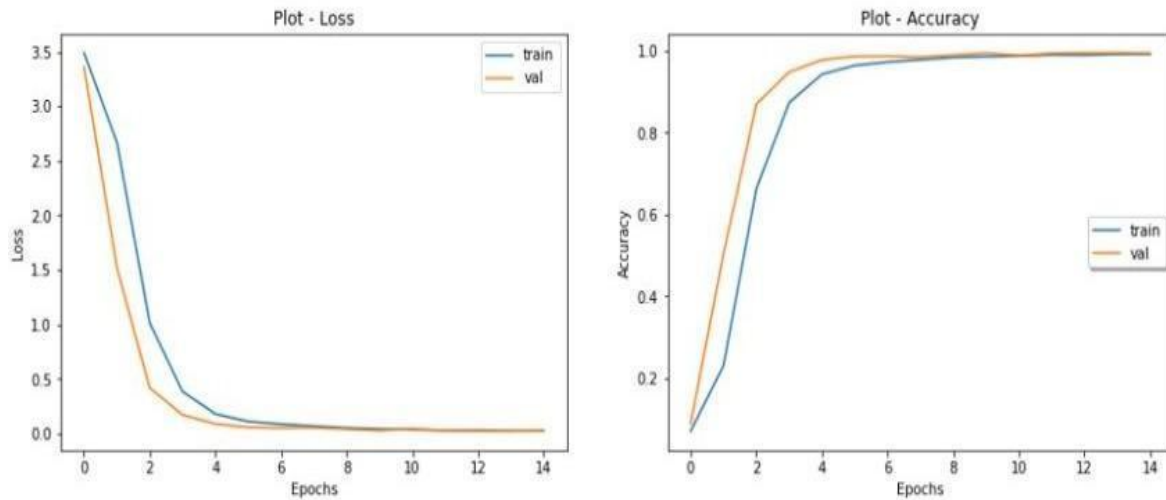


Fig 5.5 Plots of loss and accuracy of both training and validation Sets

From fig 5.5 the plots of loss and accuracy for both the sets are shown in the figure above. It can be noticed that though the network was trained for 15 epochs, the loss curve flattens after around 6 epochs. At the end of 15 epochs, both training and validation accuracy achieved above 99%. The relationship between the loss and trained epochs of the three networks is shown in Fig 5.5, and the accuracy–epoch curves are shown in Fig 5.5. The recognition accuracy maintained a substantial increase, and the loss showed an obvious decline as the number of epochs increased due to the training before the first 90 epochs, so accuracy was below 90%. This was because the CNNs had a severe lack of training, so almost all the training samples provided a positive range for the cost function. Therefore, we concluded that under our algorithm and the current training database, the maximum recognition rate was reached after training 470 epochs with a batch size of 100.

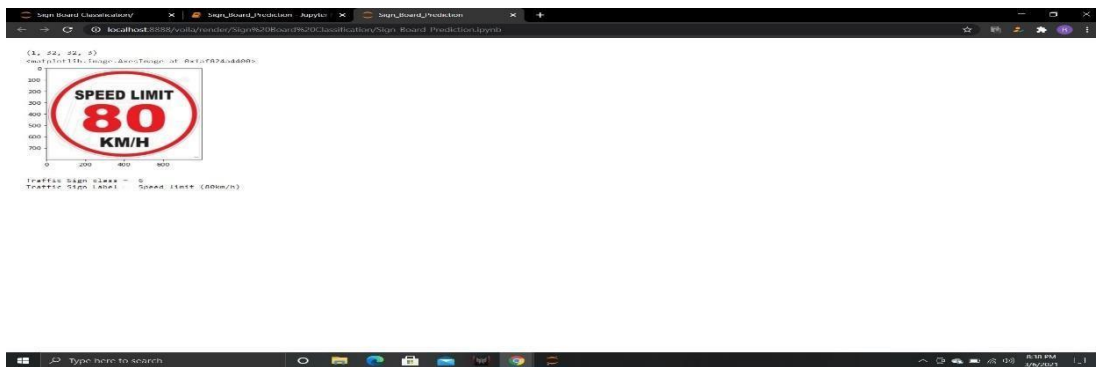


Fig 5.6 Sign Board prediction

We extensively evaluate our proposed deep networks on GTSRB (German Traffic Sign Recognition Benchmark) using our modified networks and also with original Google Net. GTSRB is the standard state-of-the-art revelation benchmark for traffic sign recognition/classification. There are significant similarities of German traffic sign with other European countries and with Indian conventions, which make it suitable to explore.

This prediction is concerned with the evaluation of the performance of the Road Sign Recognition System. It is visualized that the detection stage successfully detects and labels all images which contained road signs. A series of experiments are carried out for each stage of the detection, segmentation algorithms, the fuzzy shape recognizer, and the CNN classifier. The algorithm is evaluated with respect to the environmental and lighting conditions and quality of the image to provide better results. In this project, traffic sign recognition system was implemented which can help in creating an idea of sign which is about to approach, there by implemented and evaluated. In the sign recognition part, CNN was used and classifiers were invoked to put the unknown sign in one of the traffic sign categories depending on image. Altogether 39209 images, 31367 images for each type road sign are used for training purposes. 7842 images have been tested and the performance of the system is evaluated. The experimental results show the detection rate is above 90% and the accuracy of recognition is more than 95%.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

An In this project, traffic sign recognition system was implemented which can help in creating an idea of sign which is about to approach, there by implemented and evaluated. proposes a deep convolutional network with a fewer number of parameters and memory requirements in comparisons to existing methods. The presented network doesn't need data jittering and handcrafted data augmentations. Our main contribution includes the development of modified inception module and a deep network using spatial transformer layer for traffic sign classification In the sign recognition part, CNN was used and classifiers were invoked to put the unknown sign in one of the traffic sign categories depending on image. Altogether 39209 images, 31367 images for each type road sign are used for training purposes. 7842 images have been tested and the performance of the system is evaluated. The experimental results show the detection rate is above 90% and the accuracy of recognition is more than 95%.

6.2 FUTURE WORK

In future, the model can be implemented in vehicles for road safety and prevention of accidents even in blind spots. This approach is based on the combination of color transformation and Convolutional Neural Networks (CNN) is proposed. Working on the image preprocessed by color transformation, the CNN with fixed and learnable layers has achieved good results. Our experiment results strongly supported our conclusion. Future improvement can be made for extracting signs from test images by using advanced segmentation methods Our future work is to improve the efficiency of this algorithm. Parallel algorithm can be introduced to speed up the process time of fixed and learnable layers. The process time of multiple learnable layers could be decreased by introducing sparsity in extracting features.

REFERENCES

- [1] A. de la Escalera and M. A. Salichs, "Road Traffic Sign Detection and Classification," *IEEE Transactions on Industrial Electronics*, vol. 44, issue 6, pp. 848 – 859, Dec, 2018.
- [2] A. Møgelmoose, M. Manubhai Trivedi, and Thomas B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484-1497, December 2017.
- [3] Cai Z and Gu M, "Traffic Sign Recognition Algorithm based on Shape Signature and Dual-tree complex wavelet transform," *Journal of Central South University*, vol. 20, pp. 433-439, 2016.
- [4] Chen L, Li Q, Li M, et al, "Traffic sign detection and recognition for intelligent vehicle," *Intelligent Vehicles Symposium (IV)*, pp.908-913, 2019.
- [5] D. Deguchi, M. Shirasuna, K. Doman, I. Ide, and H. Murase, "Intelligent Traffic Sign Detector: Adaptive learning based on online gathering of training samples," in *Intelligent Vehicles Symposium (IV)*, IEEE, pp. 72– 77, 2016.
- [6] Edison A. Roxas, Joshua N. Acilo, Ryan Rhay P. Vicerra, Elmer P. Dadios, Argel A. Bandala, "Vision-based traffic sign compliance evaluation using convolutional neural network", *Applied System Invention (ICASI) 2018 IEEE International Conference on*, pp. 120-123, 2018. R.
- [7] H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo, "Goal evaluation of segmentation algorithms for traffic sign recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 917–930, Dec. 2018.
- [8] M. Kobayashi, M. Baba, K. Ohtani, and Li Li, "A method for traffic sign detection and recognition based on genetic algorithm" *2015 IEEE/SICE International Symposium on System Integration (SII)*, Nagoya, pp. 455 – 460, 11-13 Dec. 2016.

- [9] M. Meuter, C. Nunn, S. M. Gormer, S. Muller-Schneiders, and A. Kummert, "A decision fusion and reasoning module for a traffic sign recognition system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1126–1134, Dec. 2016.
- [10] Md. A. A. Sheikh and S. Mukhopadhyay, "Noise tolerant classification of aerial images into manmade structures and natural-scene images based on statistical dispersion measures," *2012 Annual IEEE India Conference (INDICON)*, Kochi, India, pp. 653–658, 7-9 Dec. 2017.
- [11] R. C. Gonzalez, R. E Woods, *Digital Image Processing*, Second Edition, 2017, ISBN 0-201-18075-8.
- [12] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3D localisation," in *Machine Vision and Applications*. New York: Springer-Verlag, pp. 1–15. Dec. 2018.
- [13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. "Detection of Traffic Signs in Real World Images, The German Traffic Sign Detection Benchmark", *The 2016 International Joint Conference on Neural Networks (IJCNN)*, DOI: 10.1109/IJCNN.2013.6706807.
- [14] T. Chen and S. Lu, "Accurate and Efficient Traffic Sign Detection Using Discriminative AdaBoost and Support Vector Regression" *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4006-4015, June 2016.
- [15] V. Barrile, M. Cacciola, G. M. Meduri, and F. C. Morabito, "Automatic Recognition of Road Signs by Hough Transform", *Journal of Earth Science and Engineering* 2, pp. 42-50, 2017.
- [16] Wang, C. (2018)," Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning". *2018 International Conference on Robots & Intelligent System (ICRIS)*. doi:10.1109/icris.2018.00047
- [17] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, "Traffic Sign Recognition using Evolutionary Ad boost Detection and Forest-ECOC Classification," *IEEE Transactions on Intelligent Transportation Systems*
- [18] Yan Lai, Nanxin Wang, Yusi Yang and Lan Lin," Traffic Signs Recognition and Classification based on Deep Feature Learning", In *Proceedings of the 7th International Conference on Pattern Recognition Applications and*

Methods.DOI:10.5220/0006718806220629, pg;622-629,2018.

APPENDICES

A. SOURCE CODE

```
import time

import numpy as np

import pandas as pd

import os

import cv2

from PIL import Image

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

np.random.seed(42)

tf.random.set_seed(42)

input_path = '/kaggle/input/gtsrb-german-traffic-sign/'

data_train = pd.read_csv(input_path + 'Train.csv',header=None)

data_test = pd.read_csv(input_path + 'Test.csv',header=None)

data_meta = pd.read_csv(input_path + 'Meta.csv',header=0)

width = data_train[0].to_numpy()[1:].astype(np.int)

height = data_train[1].to_numpy()[1:].astype(np.int)
```

```
print(width)
```

```
print(height)
```

```

import statistics

print("Median of data-set is : % s "

      % (statistics.median(width)))

print("Median of data-set is : % s "

      % (statistics.median(height)))

# display number of records in each category

data_train[data_train.columns[6]].value_counts()

sorted_series = data_train[6].value_counts().sort_index(ignore_index=True)

dt_label_count_array = sorted_series.to_numpy()

y = dt_label_count_array[0:43]

print(y)

#dictionary to label all traffic signs class.

x = ['Speed limit (20km/h)', 'Speed limit (30km/h)', 'Speed limit (50km/h)', 'Speed limit
(60km/h)',

      'Speed limit (70km/h)',

      'Speed limit (80km/h)',

      'End of speed limit (80km/h)',

      'Speed limit (100km/h)',

      'Speed limit (120km/h)',

      'No passing',

      'No passing veh over 3.5 tons',

      'Right-of-way at intersection',

```

'Priority road',

'Yield',

'Stop',

'No vehicles',

'Veh > 3.5 tons prohibited',

'No entry',

'General caution',

'Dangerous curve left',

'Dangerous curve right',

'Double curve',

'Bumpy road',

'Slippery road',

'Road narrows on the right',

'Road work',

'Traffic signals',

'Pedestrians',

'Children crossing',

'Bicycles crossing',

'Beware of ice/snow',

'Wild animals crossing',

'End speed + passing limits',

'Turn right ahead',

'Turn left ahead',

'Ahead only',

```
'Go straight or right',  
  
'Go straight or left',  
  
'Keep right',  
  
'Keep left',  
  
'Roundabout mandatory',  
  
'End of no passing',  
  
'End no passing veh > 3.5 tons']
```

```
arkgreen','yellow','grey','violet','magenta','cyan']
```

```
percent = 100.*y/y.sum()
```

```
patches, texts = plt.pie(y, startangle=90, radius=1.2)
```

```
labels = ['{0} - {1:1.2f} %'.format(i,j) for i,j in zip(x,  
percent)]
```

```
sort_legend = True
```

```
if sort_legend:
```

```
    patches, labels, dummy = zip(*sorted(zip(patches, labels, y),
```

```
        key=lambda x: x[2],
```

```
        reverse=True))
```

```
plt.legend(patches, labels, loc='left center', bbox_to_anchor=(-0.1, 1.),
```

```
    fontsize=8)
```

```
plt.savefig('piechart.png', bbox_inches='tight')
```

```
start = time.time()

image_data = []

image_labels = []

total_classes = 43

height = 43

width = 43

channels = 3

input_path = '/kaggle/input/gtsrb-german-traffic-sign/'

for i in range(total_classes):

    path = input_path + 'Train/' + str(i)

    print(path)

    images = os.listdir(path)

    for img in images:

        try:

            image = cv2.imread(path + '/' + img)

            image_fromarray = Image.fromarray(image, 'RGB')

            resize_image = image_fromarray.resize((height, width))

            image_data.append(np.array(resize_image))

            image_labels.append(i)

        except:
```

```
print("Error - Image loading")
```

```
#Converting lists into numpy arrays
```

```
image_data = np.array(image_data)
```

```
image_labels = np.array(image_labels)
```

```
print(image_data.shape, image_labels.shape)
```

```
end = time.time()
```

```
print("Execution Time: ", round(end-start,5), "Seconds")
```

```
#shuffling data
```

```
shuffle_indexes = np.arange(image_data.shape[0])
```

```
np.random.shuffle(shuffle_indexes)
```

```
image_data = image_data[shuffle_indexes]
```

```
image_labels = image_labels[shuffle_indexes]
```

```
#Splitting training and testing dataset
```

```
X_train, X_valid, y_train, y_valid = train_test_split(image_data, image_labels,
```

```
test_size=0.2, random_state=42, shuffle=True)
```

```
X_train = X_train/255
```

```
X_valid = X_valid/255
```

```
print("X_train.shape", X_train.shape)
```

```
print("X_valid.shape", X_valid.shape)
```

```
print("y_train.shape", y_train.shape)
```

```

print("y_valid.shape", y_valid.shape)

#Converting the labels into one hot encoding

y_train = keras.utils.to_categorical(y_train, total_classes)

y_valid = keras.utils.to_categorical(y_valid, total_classes)


print(y_train.shape)

print(y_valid.shape)

keras.backend.clear_session() # clearing session

np.random.seed(42) # generating random seed

tf.random.set_seed(42) # set.seed function helps reuse same set of random
variables

model = keras.models.Sequential([

    keras.layers.Conv2D(filters=16, kernel_size=(5,5), activation='relu',
input_shape=(height,width,channels)),

    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),

    keras.layers.MaxPool2D(pool_size=(2, 2)),

    keras.layers.Dropout(rate=0.25),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),

    keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),

    keras.layers.MaxPool2D(pool_size=(2, 2)),

    keras.layers.Dropout(rate=0.25),

    keras.layers.Flatten(),

```

```
keras.layers.Dense(512, activation='relu'),
```

```
keras.layers.Dropout(rate=0.5),

keras.layers.Dense(43,

activation='softmax')

])

model.summary()

#Compilation of the model

model.compile(loss='categorical_crossentropy', optimizer='adam',

metrics=['accuracy'])

epochs = 11

history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,

validation_data=(X_valid, y_valid))

pd.DataFrame(history.history).plot(figsize=(8, 5))

plt.grid(True)

plt.gca().set_ylim(0, 1) # setting limits for y-axis

plt.show()

#testing accuracy on test dataset

start = time.time()

test = pd.read_csv(input_path + '/Test.csv')

labels = test["ClassId"].values

test_imgs = test["Path"].values

img_index = 93
```

```
image = Image.open('/kaggle/input/gtsrb-german-traffic-sign/' +  
test_imgs[img_index])
```

```
img = image.resize((height,width))

img = np.array(img)/255.

img = img.reshape(1,height,width,channels)

print(img.shape)

print(labels[img_index])

plt.imshow(image)

pred = model.predict_classes(img)[0]

print(pred)

sign = classes[pred]

print(sign)

start = time.time()

test = pd.read_csv(input_path + 'Test.csv')

labels = test["ClassId"].values

imgs = test["Path"].values

data =[]

for img in imgs:

    #test_imgs.append(img_path)

    try:

        image = cv2.imread(input_path + img)
```

```
image_fromarray = Image.fromarray(image, 'RGB')
```

```
resize_image = image_fromarray.resize((height, width))

data.append(np.array(resize_image))

except:

    print("Error")

X_test = np.array(data)

X_test = X_test/255


pred = model.predict_classes(X_test)


from sklearn.metrics import accuracy_score

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

accuracy = accuracy_score(labels, pred)

print('Accuracy: %0.2f' % accuracy)

precision = precision_score(labels, pred, average="weighted")

print('Precision: %0.2f' % precision)

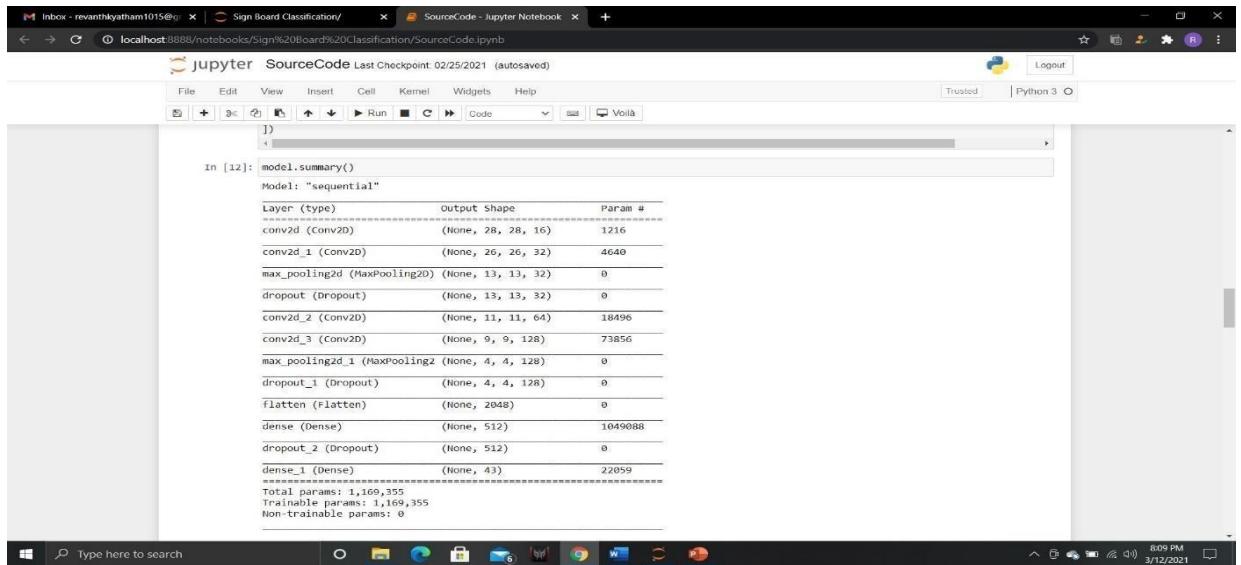
recall = recall_score(labels, pred, average="weighted")

print('Recall: %0.2f' % recall)

f1 = f1_score(labels, pred, average="weighted")

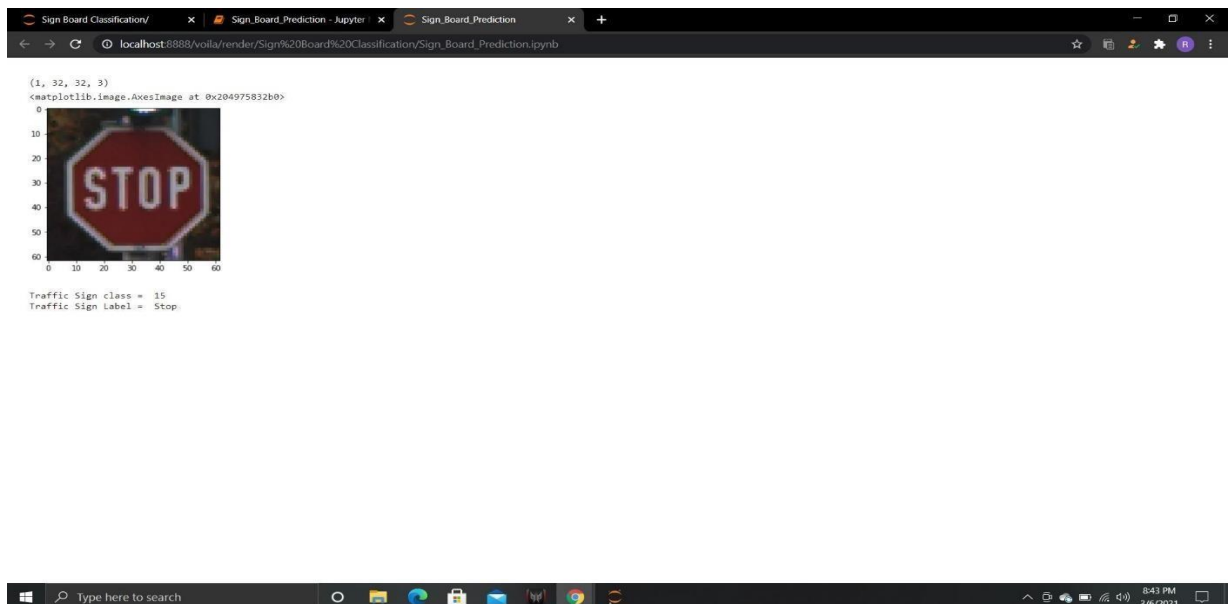
print('F1 score: %0.2f' % f1)
```

B. SCREEN SHOTS



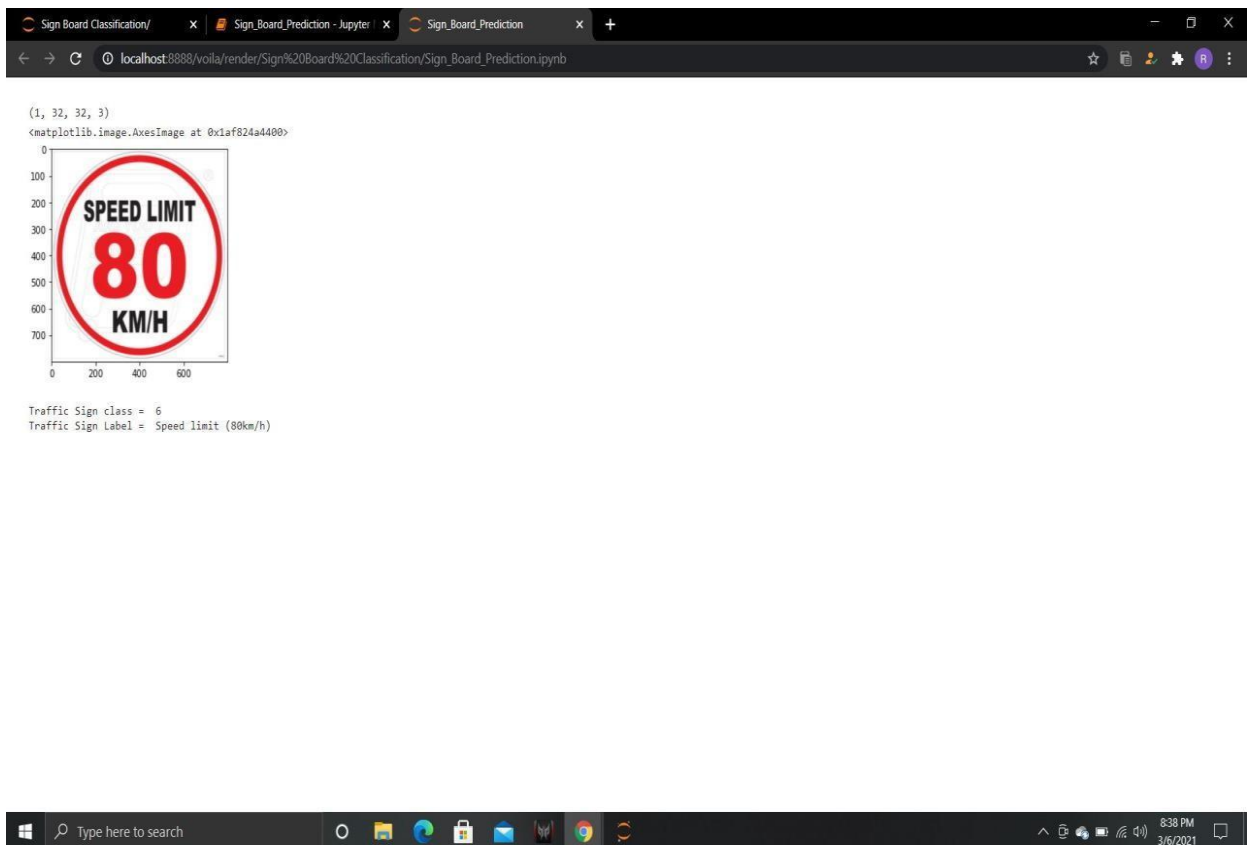
```
In [12]: model.summary()
Model: "sequential"
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 28, 28, 16)         1216
conv2d_1 (Conv2D)             (None, 26, 26, 32)         4640
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)         0
dropout (Dropout)            (None, 13, 13, 32)         0
conv2d_2 (Conv2D)             (None, 11, 11, 64)         18496
conv2d_3 (Conv2D)             (None, 9, 9, 128)          73856
max_pooling2d_1 (MaxPooling2D) (None, 4, 4, 128)         0
dropout_1 (Dropout)          (None, 4, 4, 128)         0
flatten (Flatten)            (None, 2048)               0
dense (Dense)                 (None, 512)                1049088
dropout_2 (Dropout)          (None, 512)                0
dense_1 (Dense)               (None, 43)                 22059
Total params: 1,169,355
Trainable params: 1,169,355
Non-trainable params: 0
```

Model. Summary () is used to see all parameters and shapes in each layer in our models. You can observe that total parameters are 1,169,355 and total trainable parameters are 1,169,355. Non-trainable parameter is 0.



```
(1, 32, 32, 3)
<matplotlib.image.AxesImage at 0x204975832b0>
0
10
20
30
40
50
60
0 10 20 30 40 50 60
Traffic Sign class = 15
Traffic Sign Label = Stop
```

We decided to test our model on new images as well, to make sure that it's indeed generalized to more than the traffic signs in our original dataset. We can clearly see that our model is quite confident in its predictions. Stop sign is predicted.



We decided to test our model on new images as well, to make sure that it's indeed generalized to more than the traffic signs in our original dataset. We can clearly see that our model is quite confident in its predictions. 80 Speed limits is predicted.