CONTENTS

List of Figures

List of Table

1. Introduction

Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the many important aspects for a self-driving car is the ability for it to detect traffic signs in order to provide safety and security for the people not only inside the car but also outside of it. The traffic environment consists of different aspects whose main purpose is to regulate the flow of traffic, make sure each driver is adhering to the rules so as to provide a safe and secure environment to all the parties concerned. We used the German traffic sign dataset. The dataset consisted of 43 different types of German traffic sign. The problem we are trying to solve has some advantages such as traffic signs being unique thereby resulting in object variations being small and traffic signs are clearly visible to the driver/system. The other side of the coin is that we have to contend with lighting and weather conditions. The proposed approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using color information. We have used convolutional neural networks (CNN) to classify the traffic signs and we used color-based segmentation to extract/crop signs from images.

FIG 1.1 : Input sign board

Technologies Used

The following technologies will be used in this project:

Python:Python is an interpreted, high-level, general-purpose programming language.Python has a design philosophy that emphasizes code readability, notably using significant whitespace.It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

NumPy: It is a package in Python used for Scientific Computing. NumPy package isused to perform different operations. The nparray (NumPy Array) is a multidimensional array used to store values of same datatype. These arrays are indexed just like Sequences, starts with zero.

SciPy: SciPy is a library that uses NumPy for more mathematical functions. SciPy uses NumPy arrays as the basic data structure, and comes with modules for various commonly used tasks in scientific programming, including linear algebra, integration (calculus), ordinary differential equation solving, and signal processing.

sckit-learn:Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, ..

Keras:Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. It allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Matplotlib:Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Pandas: pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

1.2 Field of Project

Keras Deep Learning: Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

2. Rationale

“Traffic Sign Recognition” helps to make the task easy for drivers to read and recognize.Traffic signs are often designed to be of a particular shape and color with symbols inside, so that there is a significant difference between the traffic signs and the background. For example, the speed limit 60 traffic sign is a circular shape with a strong number “60”. These features are also important information for traffic sign recognition systems. However, traffic sign recognition is not an easy task, because there are many adverse factors, such as bad weather, viewpoint variation, physical damage, etc.

3. Feasibility Study

3.1 Technical Feasibility

For inspecting, cleansing and transforming the dataset, Python's Keras deep learning library will be used. And for modelling, convoluntional Neural networks will be used.Both are present in Python 3.4 & hence project is technical feasible.

3.2 Operational Feasibility

Initially the individuals has to remember the signs or to understand it with their knowledge but with the implementation of this project it will detect the sign and show the information accordingly which will help the users .

3.3 Economical Feasibility

The System will follow freeware Software.Keras is an open-source library and the open source application used is a Jupyter which enables users to create interactive environment for working .

4. Objectives

The main objective of our project is to design and construct a computer-based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions.

1. To extract data from images using convolutional neural network & Machine learning

2. To validate the proposed model on created dataset

3. It is essential for self-driving cars to operate on roads.

5. Methodology

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

- **GTSRB Data Set**
  The German Traffic Sign Recognition Benchmark (GTSRB) is a publicly available data set containing 51839 images of German road signs, divided into 43 classes. Images in the data set exhibit wide variations in terms of shape and color among some classes, as well as strong similarities among others. The data pose several challenges to classification, including varying lighting and weather conditions, motion-blur, viewpoint variations, partial occlusions, physical damage and other real-world variabilities. Furthermore, resolution is not homogeneous and as low as 15×15 pixels for some images. For these reasons, human performance on this data set is not perfect, and estimated at around 98.84% on average. An additional challenge in training classification algorithms on the GTSRB data is that the 43 classes are not equally represented. The relative frequency of classes 0, 19, and 37, for example, is only 0.56%, significantly lower than the mean 1/43 = 2.33%. Moreover, since the data set was created by extracting images from video footage, it contains a track of 30 images for each unique physical real-world traffic sign instance. As a result, images from the same track are heavily correlated.
- **Convolutional Neural Network**
  The convolutional neural networks (CNN or ConvNet) are a class of deep and feed forward artificial neural network that has successfully been applied to analyzing visual imagery. CNN's use a variation of multilayer perceptron's designed to require minimal preprocessing. A CNN consists of an input and an output layer, as well as multiple hidden layers which consist of convolutional layers, pooling layers, fully connected layers and normalization layers. Here in our project, we define a convolutional neural network which is then later trained with the GTSRB training dataset. During the training, the CNN use to learn and classify data based on given classes. A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume through a differential function. A few distinct types of layers commonly used are:

- Convolution Layer: The convolution layer is the core building block of CNN. The layers' parameters consist of a set of learnable filters, which have a small receptive field. But extend through the full depth of input volume. During the forward pass, each filter is convolved across width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimensions form the full stack output volume of the convolution layer.

- Pooling Layer: Pooling in CNN is a form of down-sampling. There are several non- linear functions to implement pooling among which max pooling is most common. It partitions the input image into a set of non-overlapping rectangles and, for each sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its

rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.

- ReLU layer: ReLU is the abbrevation of Rectified Linear Units. This layer applies the non-saturating function f(x)=max(0,x). It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolutional layer.

- Fully connected layer: The high-level reasoning in a neural network is done via fully connected layers, after several convolutional and max pooling layers. Neurons in a fully connected layer have connections to all activations in the previous layer. Their activation can hence be computed with a matrix multiplication followed by a bias offset.

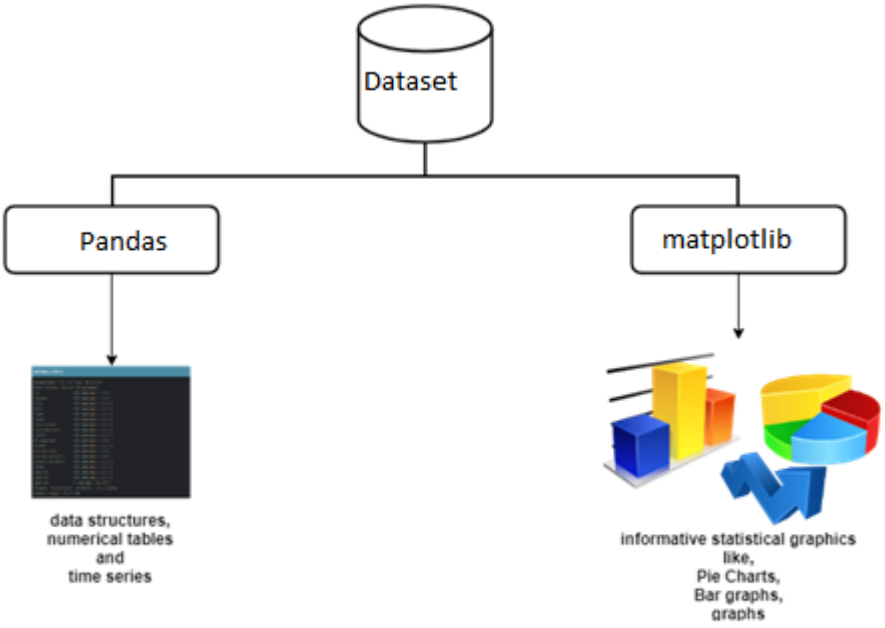

FIG 5.1 :Analysis of Dataset

6. Specific Requirements

Table 6.1: Requirements

Software Requirements	**Hardware Requirements**
• Jupyter • Importing different packages i.e. cv2 ,opencv-contrib-python,, numpy lib etc. • Latest version of python • Web Camera • Operating System-Windows	• 64-bit computer • ForAnaconda—disk space to download and install from Ananco

7.Expected outcomes

The expected outcome of this model is that it can be implemented in public transports, personal cars, and other vehicles in order to keep drivers alert and reduce human errors that lead to accidents. The project has a wide implementation of self-driving vehicles.

8. References

1. Python Tutorials :https://www.w3schools.com/python/
2. Keras Documentation: https://keras.io/
3. Convolutional Neural Network documentation: http://deeplearning.net/tutorial/lenet.html
4. www.stackoverflow.com