A

Minor Project

On

ROAD SIGN INTIMATION USING CNN

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Ch. Sharoon
(19E41A0563)
J. Akshara Bharathi
(19E41A0571)
A. Manasa
(19E41A0572)
B. Arun
(19E41A0591)

Under the Guidance of

Dr. S Venkata Achuta Rao

(Principal)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SREE DATTHA INSTITUTE OF ENGINEERING AND SCIENCE

(Accredited by NAAC, Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) of the UGCAct.1956,

Sheriguda (V), Ibrahimpatnam (M), Ranga Reddy – 501510.

2019-23

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**ROAD SIGN INTIMATION USING CNN**" is being submitted by **CH.SHAROON(19E41A0563), J.AKSHARA BHARATHI (19E41A0571), A.MANASA(19E41A0572), B.ARUN(19E41A0591)** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mrs. P. Swapna Reddy
Internal Guide

Dr. Md. Sameeruddin Khan
Principal

Dr. M. Vara Prasad Rao
HOD

External Examiner

Submitted for viva voice Examination held on _____________________

ACKNOWLEDGEMENT

Apart from our efforts, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard for my guide **Mrs. P. Swapna Reddy,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We are also thankful to **Dr. M. Varaprasad Rao,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. S.Venkata Achuta Rao,** Dean Acadamics for his valuable technical support **Dr. Md. Sameeruddin Khan,** Principal for being cooperative throughout this project. We would like to express our sincere gratitude to **Dr. GNV Vibhav Reddy**, ViceChairman and **Sri. G. Panduranga Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout this project.

The guidance and support were received from all the members of **Sree Dattha Institute of Engineering and Science** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

ABSTRACT

To ensure a smooth and secure flow of traffic, road signs are essential. A major cause of road accidents is negligence in viewing the Traffic signboards and interpreting them incorrectly. The proposed system helps in recognizing the Traffic sign and sending a voice alert through the speaker to the driver so that he/ she may take necessary decisions. There have been a lot of technological advancements and cars with auto-pilot mode have come up. Autonomous vehicles have come into existence. There has been a boom in the self-driving car industry. However, these features are available only in some high-end cars which are not affordable to the masses. We wanted to devise a system which helps in easing the job of driving to some extent. The proposed system is trained using Convolutional Neural Network (CNN) which helps in traffic sign image recognition and classification. A set of classes are defined and trained on a particular dataset to make it more accurate. The German Traffic Sign Benchmarks Dataset was used, which contains approximately 43 categories and 51,900 images of traffic signs. The accuracy of the execution is about 98.52 percent. Following the detection of the sign by the system, a voice alert is sent through the speaker which notifies the driver. The proposed system also contains a section where the vehicle driver is alerted about the traffic signs in the near proximity which helps them to be aware of what rules to follow on the route. The aim of this system is to ensure the safety of the vehicle's driver, passengers, and pedestrians.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

INTRODUCTION

1.1 PROBLEM DEFINITION

Traffic signs are road facilities that convey, guide, restrict, warn, or instruct information using words or symbols. With the development of automotive intelligent technology, famous car companies, such as Mercedes-Benz, BMW, etc., have actively invested in ADAS (Advanced Driver Assistance System) research. Commercialized ADAS systems not only include Lane Keep Assist Systems, but also include TSR (Traffic Sign Recognition) systems to remind drivers to pay attention to the speed. If drivers and pedestrians do not notice this information, it can lead to the occurrence of traffic accidents. With the increasing demand for the intelligence of vehicles, it is extremely necessary to detect and recognize traffic signs automatically through computer technology. Research in this area began in the 1980s, to solve this problem. To make them 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.

Traffic signs are road facilities that convey, guide, restrict, warn, or instruct information using words or symbols. With the development of automotive intelligent technology, famous car companies, such as Mercedes-Benz, BMW, etc., have actively invested in ADAS (Advanced Driver Assistance System) research.

Commercialized ADAS systems not only include Lane Keep Assist Systems, but also include TSR (Traffic Sign Recognition) systems to remind drivers to pay attention to the speed. If drivers and pedestrians do not notice this information, it can lead to the occurrence of traffic accidents.

However, traffic sign recognition is not an easy task, because there are many adverse factors, such as bad weather, viewpoint variation, physical damage, etc. There have been a lot of technological advancements and cars with auto-pilot mode have come up. Autonomous vehicles have come into existence. There has been a boom in the self driving car industry. However, these features are available only in some high end cars which are not affordable to the masses. We wanted to devise a system which helps in easing the job of driving to some extent. On conducting a survey we found that the magnitude of road accidents in India is alarming. Reports suggest that every hour there are about 53 mishaps taking place on the roads.

1.2 PROJECT OVERVIEW

Although the same kind of traffic signs have some consistency in color, in outdoor environments the color of the traffic signs is greatly influenced by illumination and light direction. Therefore, the color information is not fully reliable. As vehicle mounted cameras are not always perpendicular to the traffic signs, and the shape of traffic signs are often distorted in road scenes, the shape information of traffic signs is no longer fully reliable. Traffic signs in some road scenes are often obscured by buildings, trees, and other vehicles; therefore, we needed to recognize the traffic signs with incomplete information. Traffic sign discoloration, traffic sign damage, rain, snow, fog, and other problems, are also given as challenges in the process of traffic sign detection and classification.

Traffic signs will be stored in a database so that the driver will be notified with message and voice alert regarding the approaching Traffic Sign. Our Idea is to secure the drivers and pedestrians from accidents.

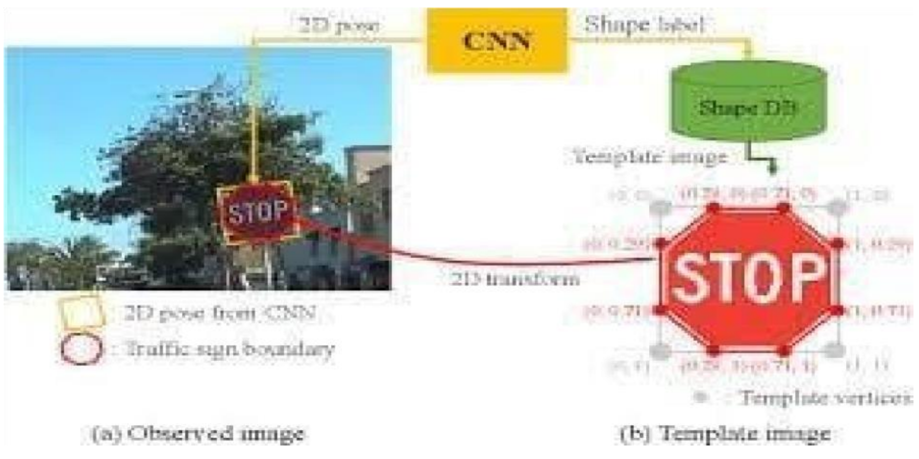


FIG 1: System Architecture

2. LITERATURE SURVEY

LITERATURE SURVEY

2.1 EXISTING SYSTEM

In this era of fast paced life, people generally tend to miss out on recognizing the traffic sign and hence break the rules.

A lot of research has been done in this domain to reduce the number of accidents using many efficient algorithms.

2.1 EXISTING SYSTEM

The algorithms and their accuracies are :

- AlexNet : 92.63%
- GoogleNet : 80.5%
- VssaNet : 94.2%

2.2 PROPOSED SYSTEM

We propose the system to train using Convolution Neural network(CNN) Algorithm. CNN can take picture as input, assign priority to different items in the picture and distuinguish them from one another by dividing the image into pixels.

Our method performs only one feature extraction through the detection and classification stage, which causes feature sharing throughout the two stages. Compared with algorithms used in the different feature extraction methods, in the detection and classification stage, this saves a lot of processing time and makes it feasible for use in real time applications.

2.2.1 PROJECT FLOW

- Detect the traffic sign board in live and capture the image
- In the captured image detect the traffic sign using Convolution Neural Network
- Compare the image with the datasets that is provided and classify the image
- If the image is present in the dataset, then message along with voice alert to the driver

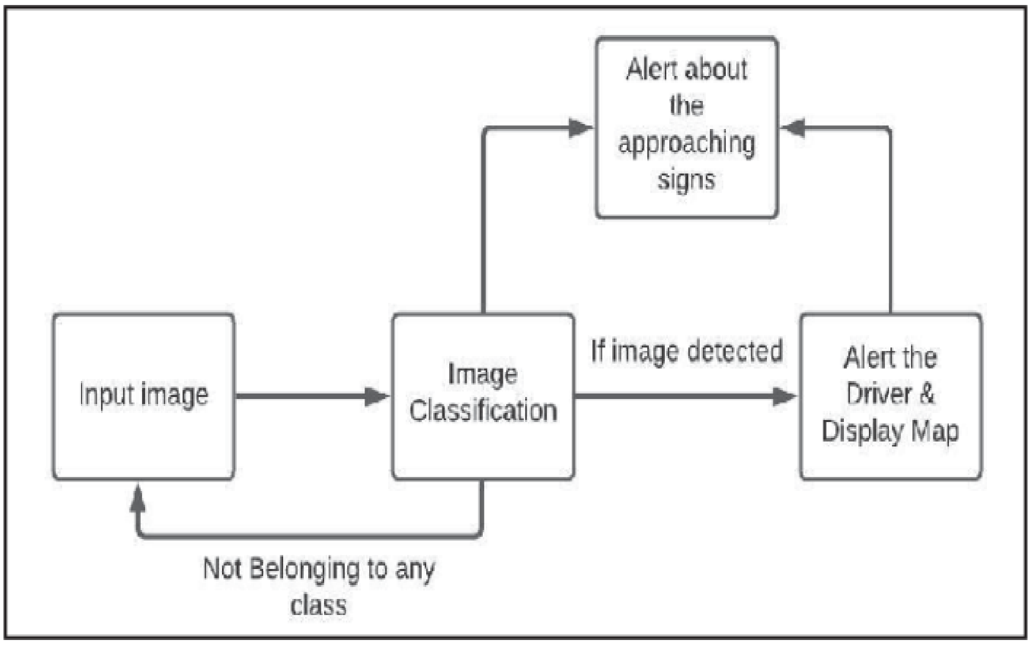

Alert about the approaching signs
Input image
Image Classification
If image detected
Alert the Driver & Display Map
Not Belonging to any class

Fig 2.2.1: Project Flow

ADVANTAGES

- Traffic signals help for movement of traffic securely without any collision.
- They can reduce the number of accidents on roads like pedestrian accident and right- angle collision of two cars.
- It will give voice alert to the driver.
- It reduces the accidents.
- Accuracy is huge.

2.2.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out.This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three considerations involved in the feasibility study are,

i. Economical Feasibility

ii.TechnicalFeasibility

iii. Social Feasibility

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had tobe purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demandon the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3. SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

3.1 HARDWARE SPECIFICATIONS

- **System** **-** Pentium IV 2.4 GHz**.**
- **Hard Disk** **-** 40 GB.
- **Floppy Drive** **-** 1.44 Mb.
- **Monitor** **-** 14’ Colour Monitor.
- **Mouse** **-** Optical Mouse.
- **RAM** - 8 Gb.

3.2 SOFTWARE SPECIFICATIONS

- **Operating system** **-** Windows 10 or 11.
- **Coding Language** **-** Python.
- **Front-End** - Python.
- **Designing** **-** Html,CSS, java script.
- **Data Base** **-** MySQL.

4. SYSTEM DESIGN

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

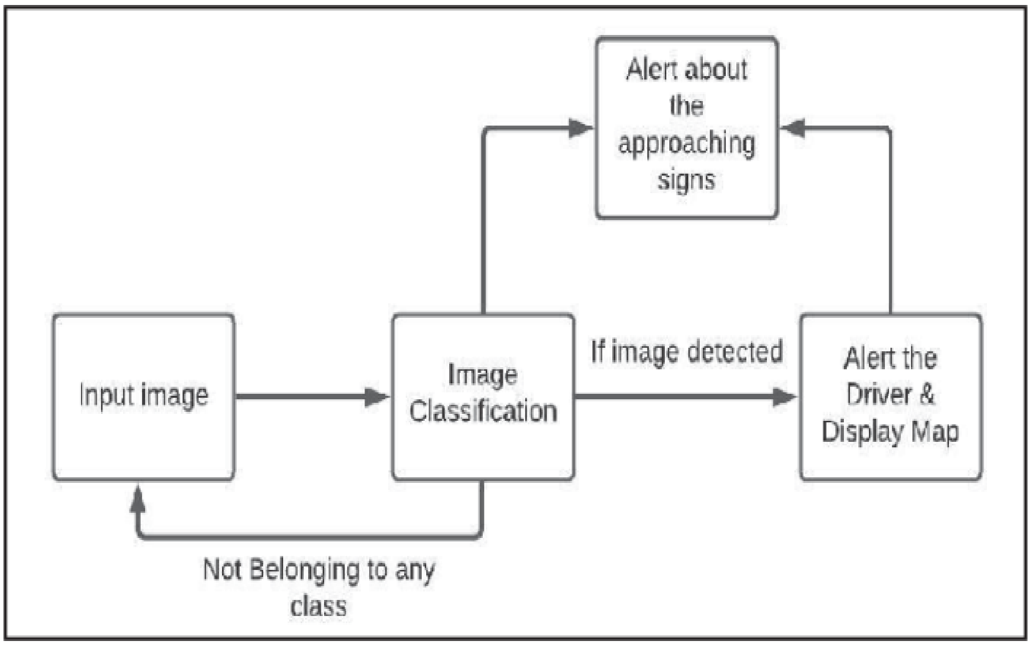

Alert about the approaching signs
Input image
Image Classification
If image detected
Alert the Driver & Display Map
Not Belonging to any class

Fig 4.1 Data Flow Diagram

4.2 ACTIVITY DIAGRAM

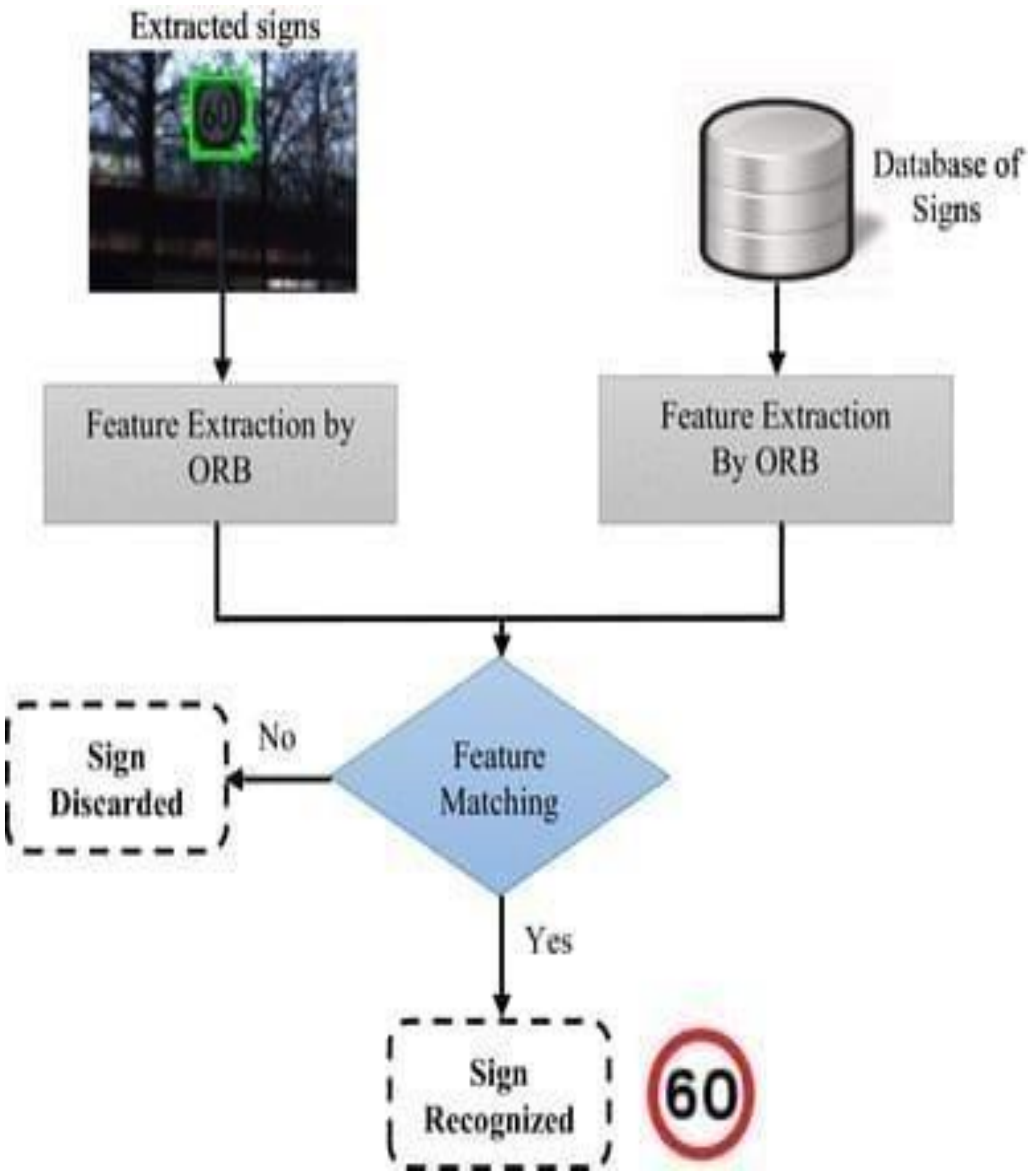


Fig 4.2 Activity Diagram

4.3 USECASE DIAGRAM

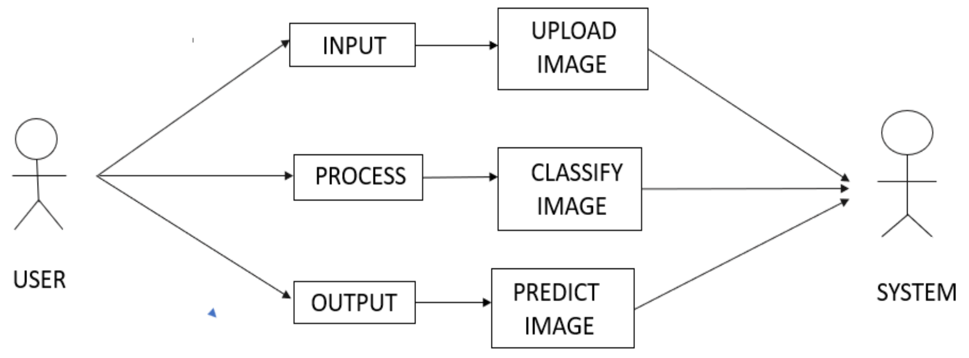


Fig 4.3 Usecase Diagram

5.SYSTEM ENVIRONMENT

SYSTEM ENVIRONMENT

5.1 PYTHON

It is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python interpreters are available for many operating systems. Python ismanaged by the non- profit Python Software Foundation. Python features adynamic types system and automatic memory management. It supports multiple programming paradigms, including object oriented, functional and procedural and has a large and comprehensive standard library. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

PYTHON IDLE

IDLE stands for Integrated Development and Learning Environment. The story behind the name IDLE is similar to Python. Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5.2b1 release. It is packaged as an optional part of the Python packaging with many Linux, Windows, and Mac distributions.

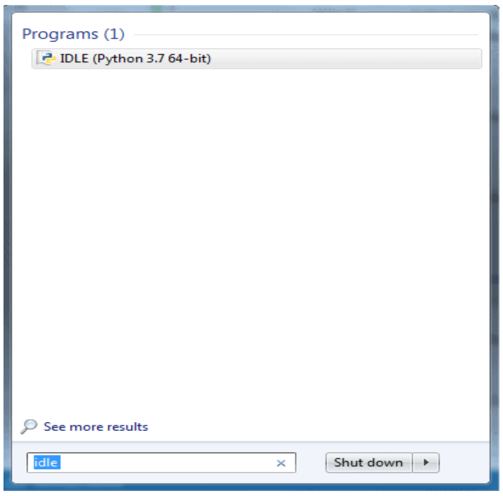
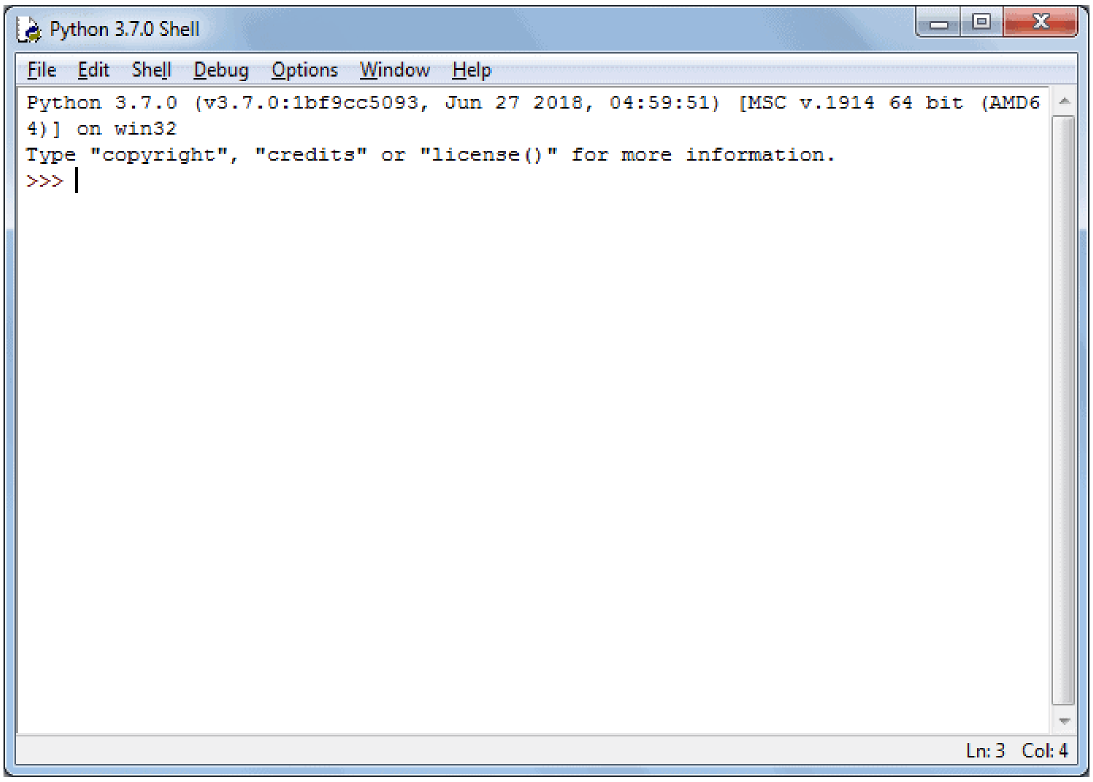


Fig 5.1.1 Python IDLE



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

.

Fig 5.1.2 Python IDLE

6. METHODOLOGY

METHODOLGY

6.1 CONVOLUTION NEURAL NETWORK

It is a type of feed-forward artificial network where the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Convolutional neural network is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm. Familiarity with the concepts and advantages, as well as limitations, of convolutional neural network is essential to leverage its potential to improve radiologist performance and, eventually, patient care. It is one of the techniques to do image classification and image recognition in neural networks. It is designed to process the data by multiple layers of arrays. This type of neural network is used in applications like image recognition or face recognition.And it operates directly on the images rather than focusing on feature extraction which other neural networks do. 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. CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as h * w * d, where h= height w= width and d= dimension. For example, An RGB image is 6 * 6 * 3 array of the matrix, and the grayscale image is 4 * 4 * 1 array of the matrix.

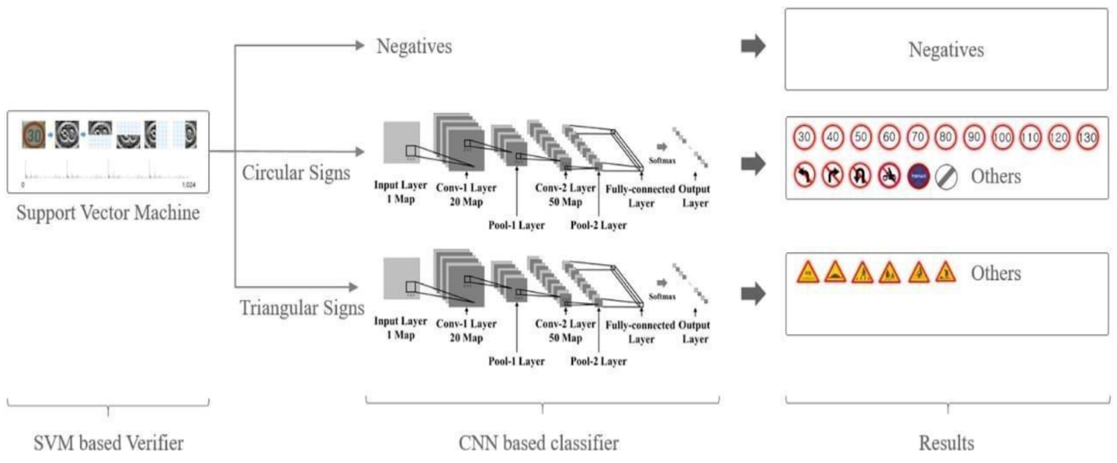


Fig 6.1.1: CNN Process

6.2 LIBRARIES

TENSORFLOW

TensorFlow is a popular framework of **machine learning** and **deep learning**.

- It is **a free** and **open-source** library which is released on **9 November 2015** and developed by **Google Brain Team**.
- It is entirely based on Python programming language and use for numerical computation and data flow, which makes machine learning faster and easier.
- TensorFlow can train and run the deep neural networks for image recognition, handwritten digit classification, recurrent neural network, **word embedding**, **natural language processing**, video detection, and many more.
- TensorFlow is run on multiple **CPU**s or **GPU**s and also mobile operating systems.

The word TensorFlow is made by two words, i.e., Tensor and Flow.

1. **Tensor** is a multidimensional array
2. **Flow** is used to define the flow of data in operation.

KERAS

- Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK.
- It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks.
- It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

OPENCV

- **OpenCV** is a huge open-source library for computer vision, machine learning, and image processing.

- OpenCV supports a wide variety of programming languages like Python, C++, Java, etc.

- It can process images and videos to identify objects, faces, or even the handwriting of a human.

- When it is integrated with various libraries, such as **Numpy** which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal.

PILLOW

- The Python Imaging Library is best suited for image archival and batch processing applications.
- python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

MATPLOTLIB

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- It was introduced by John Hunter in the year 2002.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.
- Matplotlib consists of several plots like line, bar,scatter,histogram,etc

PYTORCH

- PyTorch is an open source machine learning library for Python and is completely based on Torch.
- It is primarily used for applications such as natural language processing.
- PyTorch is developed by Facebook's artificial-intelligence research group along with
- Uber's "Pyro" software for the concept of in-built probabilistic programming.

PANDAS

- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively.
- It provides various data structures and operations for manipulating numerical data and time series.
- This library is built on top of the NumPy library.
- Pandas is fast and it has high performance & productivity for users.

SCIPY

- The SciPy is an open-source scientific library of Python that is distributed under a BSD license. It is used to solve the complex scientific and mathematical problems. It is built on top of the Numpy extension, which means if we import the SciPy, there is no need to import Numpy.
- The Scipy is pronounced as Sigh pi, and it depends on the Numpy, including the appropriate and fast dimension array manipulation.
- It provides many user-friendly and effective numerical functions for numerical integration and optimization.

SCKIT-LEARN

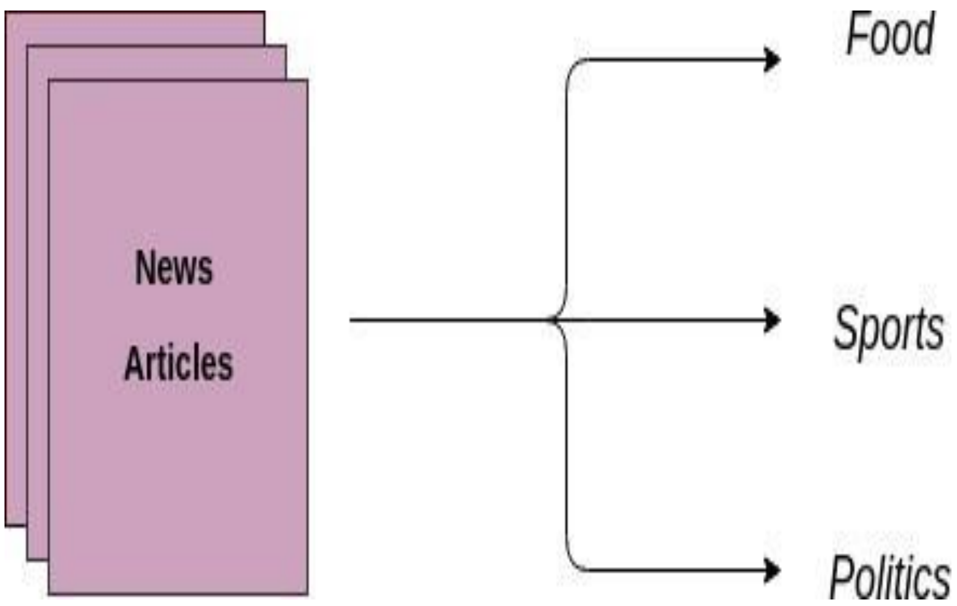
- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
- This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

THEANO

- Theano is a Python library that allows us to evaluate mathematical operations including multi-dimensional arrays so efficiently. It is mostly used in building Deep Learning Projects. It works a way more faster on Graphics Processing Unit (GPU) rather than on CPU. Theano attains high speeds that gives a tough competition to C implementations for problems involving large amounts of data. It can take advantage of GPUs which makes it perform better than C on a CPU by considerable orders of magnitude under some certain circumstances.
- It knows how to take structures and convert them into very efficient code that uses numpy and some native libraries. It is mainly designed to handle the types of computation required for large neural network algorithms used in Deep Learning. That is why, it is a very popular library in the python.

6.3 TABLE OF CONTENTS

In this article, I will explain about the text classification and the step by step process to implement it in python.



Text Classification is an example of supervised machine learning task since a labelled dataset containing text documents and their labels is used for train a classifier. An end-to-end text classification pipeline is composed of three main components:

6.4.1 Dataset Preparation: The first step is the Dataset Preparation step which includes the process of loading a dataset and performing basic pre-processing. The dataset is then splitted into train and validation sets.

6.4.2 Feature Engineering: The next step is the Feature Engineering in which the raw dataset is transformed into flat features which can be used in a machine learning model. This step also includes the process of creating new features from the existing data.

6.4.3 Model Training: The final step is the Model Building step in which a machine learning model is trained on a labelled dataset.

6.4.4 Improve Performance of Text Classifier : In this article, we will also look at the different ways to improve the performance of text classifiers.

6.5 Getting your Machine Ready : Lets implement basic components in a step by step manner in order to create a text classification framework in python .

To start with, import all the required libraries. You would need requisite libraries to run this code – you can install them at their individual official links

- Tensor flow
- Scikit-learn
- Numpy
- OpenCV
- Pillow
- Imutils
- Keras

# libraries for dataset preparation, feature engineering, model training.

6.4 IMPLEMENTATION OF CODE

Step 1: Project implementation (or project execution) is the phase where visions and plans become reality. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying for funds and finding the financial resources of a project. Technical implementation is one part of executing a project.

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Built by experienced developers, it takes care of much of the hassle of Web development.

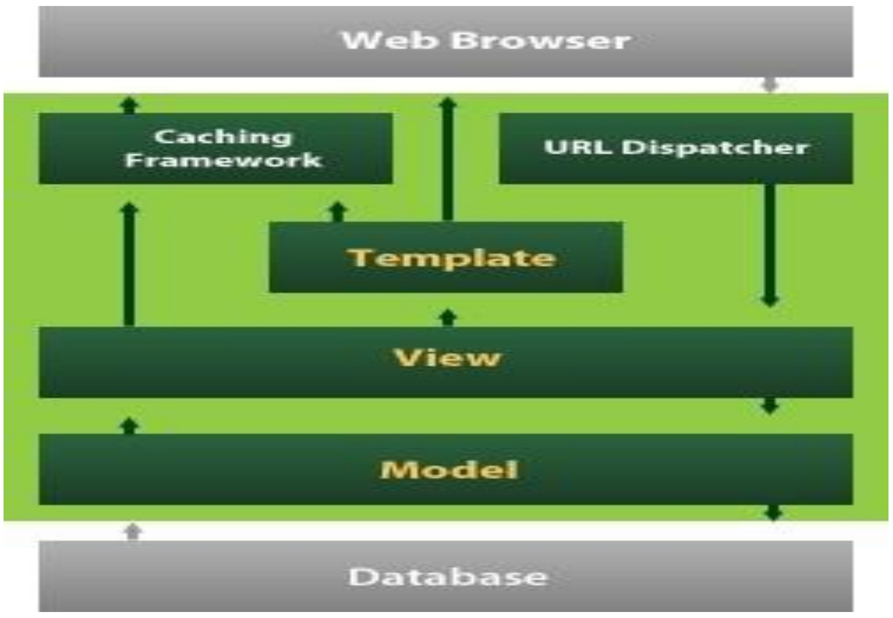


Figure 6.1.1Django

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

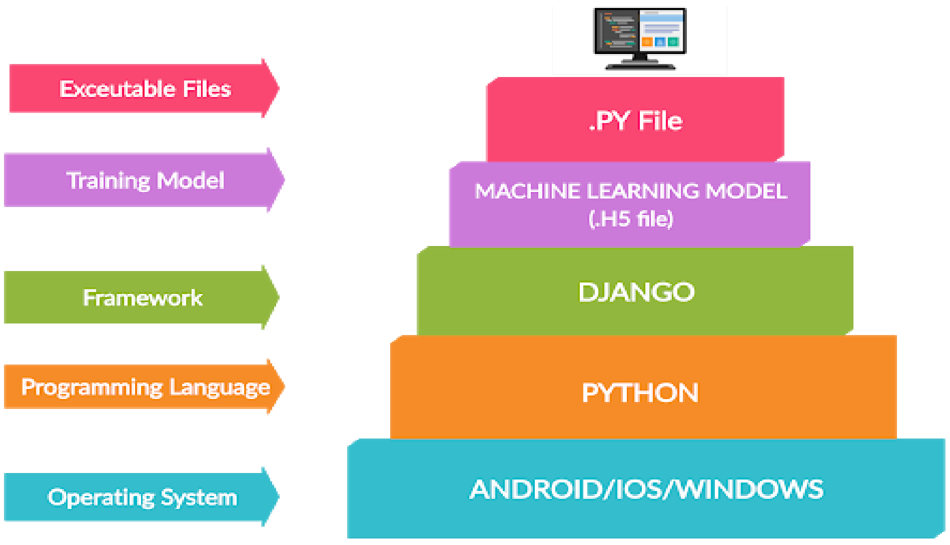


Figure 6.4.2 Structure of Django application

STEP 2: Dataset preparation

For the purpose of this article, I am the using dataset of amazon reviews which can be downloaded at this link. The dataset consists of 3.6M text reviews and their labels, we will use only a small fraction of data. To prepare the dataset, load the downloaded data into a pandas dataframe containing two columns – text and label. (Source)

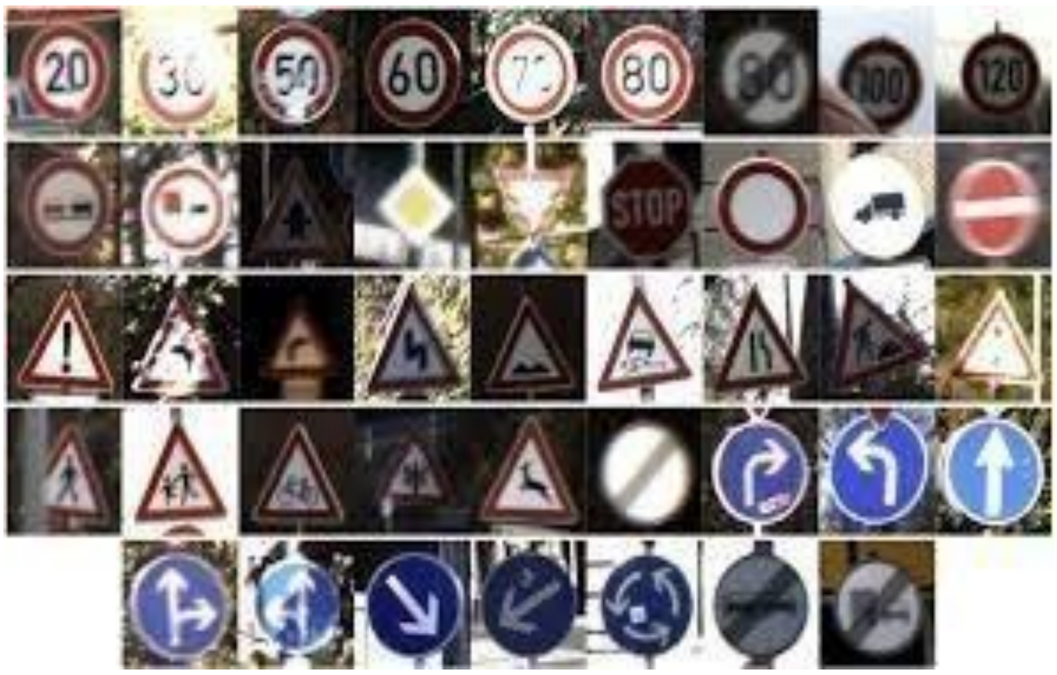


Fig 6.4.3 : Dataset of traffic signs

Model Building

The final step in the text classification framework is to train a classifier using the features created in the previous step. There are many different choices of machine learning models which can be used to train a final model. We will implement following different classifiers for this purpose:

1. Naive Bayes Classifier
2. Linear Classifier
3. Support Vector Machine
4. Boosting Models
5. Shallow Neural Networks
6. Deep Neural Networks

- Convolutional Neural Network (CNN)
- Long Short Term Modelr (LSTM)
- Gated Recurrent Unit (GRU)
- Bidirectional RNN
- Recurrent Convolutional Neural Network (RCNN)

Lets implement these models and understand their details. The following function is a utility function which can be used to train a model. It accepts the classifier, feature_vector of training data, labels of training data and feature vectors of valid data as inputs.

```
def train_model(classifier, feature_vector_train, label, feature_vector_valid,
is_neural_net
=False):

    # fit the training dataset on the classifier

    classifier.fit(feature_vector_train, label)

    # predict the labels on validation dataset

    predictions = classifier.predict(feature_vector_valid)

    if is_neural_net:

predictions = predictions.argmax(axis=-1)

    return metrics.accuracy_score(predictions, valid_y)
```

NAIVE BAYES

Implementing a naive bayes model using sklearn implementation with different featuresNaive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.

```
# Naive Bayes on Count Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid
_count)

print "NB, Count Vectors: ", accuracy
```

```
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf, train_y, xvalid_ tfidf)

print "NB, WordLevel TF-IDF: ", accuracy

# Naive Bayes on Ngram Level TF IDF Vectors

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram, train_y,
xvalid_tfidf_ngram)

print "NB, N-Gram Vectors: ", accuracy

accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram_chars, tr ain_y,
xvalid_tfidf_ngram_chars)

print "NB, CharLevel Vectors: ", accuracy
```

LINEAR CLASSIFIER

Implementing a Linear Classifier (Logistic Regression) Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function. One can read more about logistic regression.

```
  # Linear Classifier on Count Vectors

  accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, train_y, xvalid_
  count)
 print "LR, Count Vectors: ", accuracy

  # Linear Classifier on Word Level TF IDF Vectors
```

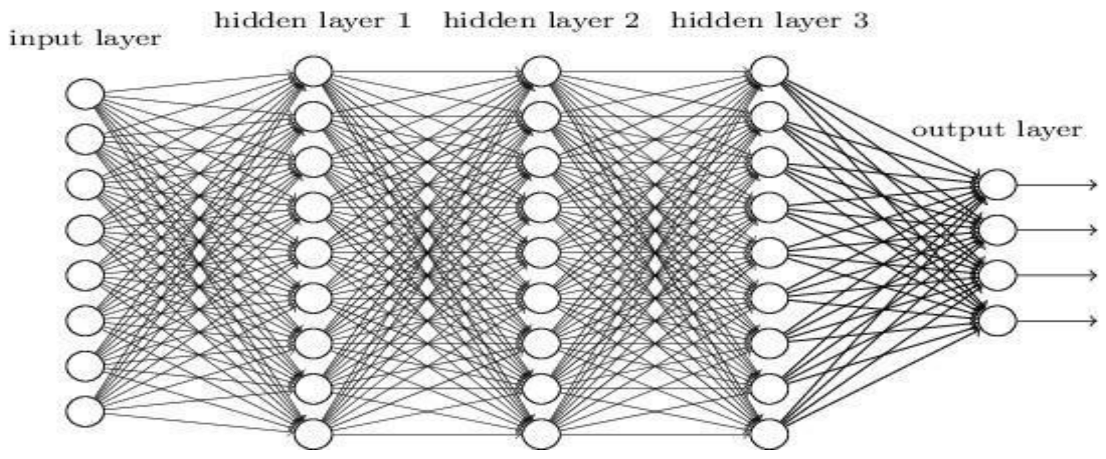
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf, train_y, xvalid_tf idf)

print "LR, WordLevel TF-IDF: ", accuracy

# Linear Classifier on Ngram Level TF IDF Vectors accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngram, train_y, x valid_tfidf_ngram)

DEEP NEURAL NETWORKS

Deep Neural Networks are more complex neural networks in which the hidden layers perform much more complex operations than simple sigmoid or relu activations.



CONVOLUTIONAL NEURAL NETWORK

In Convolutional neural networks, convolutions over the input layer are used to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters and combines their results.

```
# Add an Input Layer

    input_layer = layers.Input((70, ))



    # Add the word embedding Layer

    embedding_layer = layers.Embedding(len(word_index) + 1, 300, weights=[embedding_
  matrix], trainable=False)(input_layer)

    embedding_layer = layers.SpatialDropout1D(0.3)(embedding_layer)



    # Add the convolutional Layer

   conv_layer = layers.Convolution1D(100, 3, activation="relu")(embedding_layer)

    # Add the pooling Layer

    pooling_layer = layers.GlobalMaxPool1D()(conv_layer)

output_layer1 = layers.Dense(50, activation="relu")(pooling_layer) output_layer2 = la yers.Dense(1,
activation="sigmoid")(output_layer1)
```

6.5 SOURCE CODE

TRAFFIC SIGN CODE

```
from google.colab import drive
drive.mount('/content/gdrive')
cd '/content/gdrive/My Drive/akshara/gtsrb-german-traffic-sign/Train'
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
data = []
labels = []
classes = len(os.listdir('/content/gdrive/MyDrive/akshara/gtsrb-german-
traffic-sign/Train'))
cur_path = os.getcwd()
!unzip '/content/gdrive/MyDrive/akshara/archive.zip' -
d '/content/gdrive/MyDrive/akshara'
#Retrieving the images and their labels
for i in range(classes):
    path = os.path.join(cur_path,'gtsrb-german-traffic-sign/Train',str(i))
    images = os.listdir(path)
for a in images:
    try:
        path2 = os.path.join(path,a)
        image = Image.open(path2)
        image = image.resize((30,30))
        image = np.array(image)
#sim = Image.fromarray(image)
        data.append(image)
        labels.append(i)
    except:
        print("Error loading image")
 #Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
# Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.
2, random_state=42)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
#Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)

#Building the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
#Compilation of the model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accur
acy'])
epochs = 50
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation
_data=(X_test,
y_test))
model.save("my_model1.h5")
#plotting graphs for accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
#Accuracy with the test data
from sklearn.metrics import accuracy_score
#load the trained model to classify sign
```

```
from keras.models import load_model
model = load_model('/content/gdrive/My Drive/akshara/my_model.h5')
import os
import cv2
path2  = "/content/gdrive/MyDrive/akshara/gtsrb-german-traffic-
sign/Test/00002.png"


img = cv2.imread(path2)
img=cv2.resize(img,(30,30))
img = cv2.cvtColor(img,cv2.COLOR_RGB2BGR)
plt.imshow(img)
plt.show()
pred = model.predict(img[np.newaxis, :, :, :])
pred_class = np.argmax(pred)
print(pred_class,path2)
if pred_class == 0:
    print("0")
elif pred_class == 1:
    print("1")
elif pred_class == 2:
    print("2")
elif pred_class == 3:
    print("3")
elif pred_class == 4:
    print("4")
elif pred_class == 5:
    print("5")
elif pred_class == 6:
    print("6")
elif pred_class == 7:
    print("7")
elif pred_class == 8:
    print("8")
elif pred_class == 9:
    print("9")
elif pred_class == 10:
    print("10")
elif pred_class == 11:
    print("11")
elif pred_class == 12:
    print("12")
elif pred_class == 13:
    print("13")
elif pred_class == 14:
    print("14")
elif pred_class == 15:
    print("15")
```

```
elif pred_class == 16:
    print("16")
elif pred_class == 17:
    print("17")
elif pred_class == 18:
    print("18")
elif pred_class == 19:
    print("19")
elif pred_class == 20:
    print("20")
elif pred_class == 21:
    print("21")
elif pred_class == 22:
    print("22")
elif pred_class == 23:
    print("23")
elif pred_class == 24:
    print("24")
elif pred_class == 25:
    print("25")
elif pred_class == 26:
    print("26")
elif pred_class == 27:
    print("27")
elif pred_class == 28:
    print("28")
elif pred_class == 29:
    print("29")
elif pred_class == 30:
    print("30")
elif pred_class == 31:
    print("31")
elif pred_class == 32:
    print("32")
elif pred_class == 33:
    print("33")
elif pred_class == 34:
    print("34")
elif pred_class == 35:
    print("35")
elif pred_class == 36:
    print("36")
elif pred_class == 37:
    print("37")
elif pred_class == 38:
    print("38")
elif pred_class == 39:
    print("39")
```

```
elif pred_class == 40:
    print("40")
elif pred_class == 41:
    print("41")
elif pred_class == 42:
    print("42")

else:
      print("Please Enter a Valid Input")
```

7.SYSTEM TESTING

SYSTEM TESTING

INTRODUCTION

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is the best performed when user development is asked to assist in identifying all error and bugs. The sample data are used for testing. Testing is aimed at ensuring that the system was accurately an efficient before live operation commands.

TESTING OBJECTIVES

The main objective of the testing to uncover host of error, systematically and with minimum effort and time. Stating formally, we can say testing is a process of executing a program with intent of finding an error.

A successful test is one that uncover an as yet undiscovered error.
A good testcase is one that has probability of finding an error, if it exists. A test is inadequate
to detect possibly present errors.

BASIC LEVEL OF TESTING

CODE TESTING

This examines the logic of the program. For example the logic for uploading various sample
data with the various sample files and directories were tested and verified.

SPECIFICATION TESTING

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situations and combination of the conditions in all the modules are tested.

Test-Driven Development TDD: Unit Testing should be done along with the Python, and for that developers use Test-Driven Development method. In TDD method, you first design Python Unit tests and only then you carry on writing the code that will implement this feature.

Stubs and Mocks: They are two main techniques that simulate fake methods that are being tested. A Stub is used to fill in some dependency required for unit test to run correctly. A Mock on the other hand is a fake object which runs the tests where we put assert.

BLACK BOX TESTING

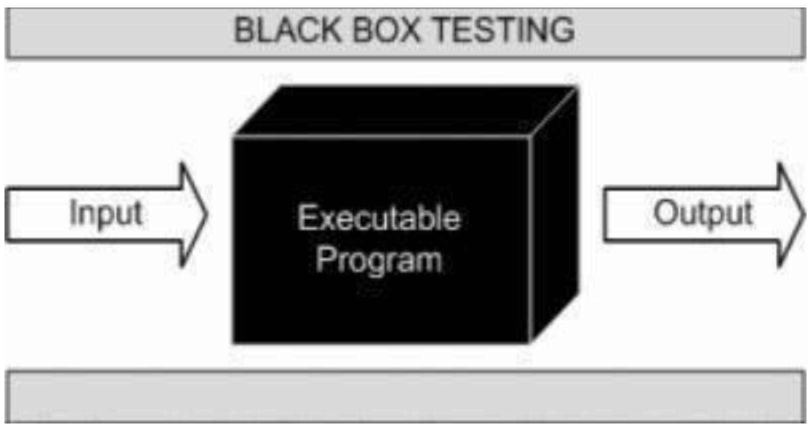
What is Black Box testing?

Black box testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In BlackBox Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

How to do Black Box Testing?

Here are the generic steps followed to carry out any type of Black Box Testing.
Initially, the requirements and specifications of the system are examined.
Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
Tester determines expected outputs for all those inputs.
Software tester constructs test cases with the selected inputs.
The test cases are executed.



WHITE BOX TESTING

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:
Internal security holes
Broken or poorly structured paths in the coding processes
The flow of specific inputs through the code
Expected output
The functionality of conditional loops
Testing of each statement, object, and function on an individual basis
The testing can be done at system, integration and unit levels of software development.

SYSTEM TESTING

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer- based system. There are three main kinds of system testing

Alpha Testing
Beta Testing Acceptance
Testing

ALPHA TESTING

Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or the public. The focus of this testing is to simulate real users by using a black box and white box techniques. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually, the testers are internal employees of the organization. To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

BETA TESTING

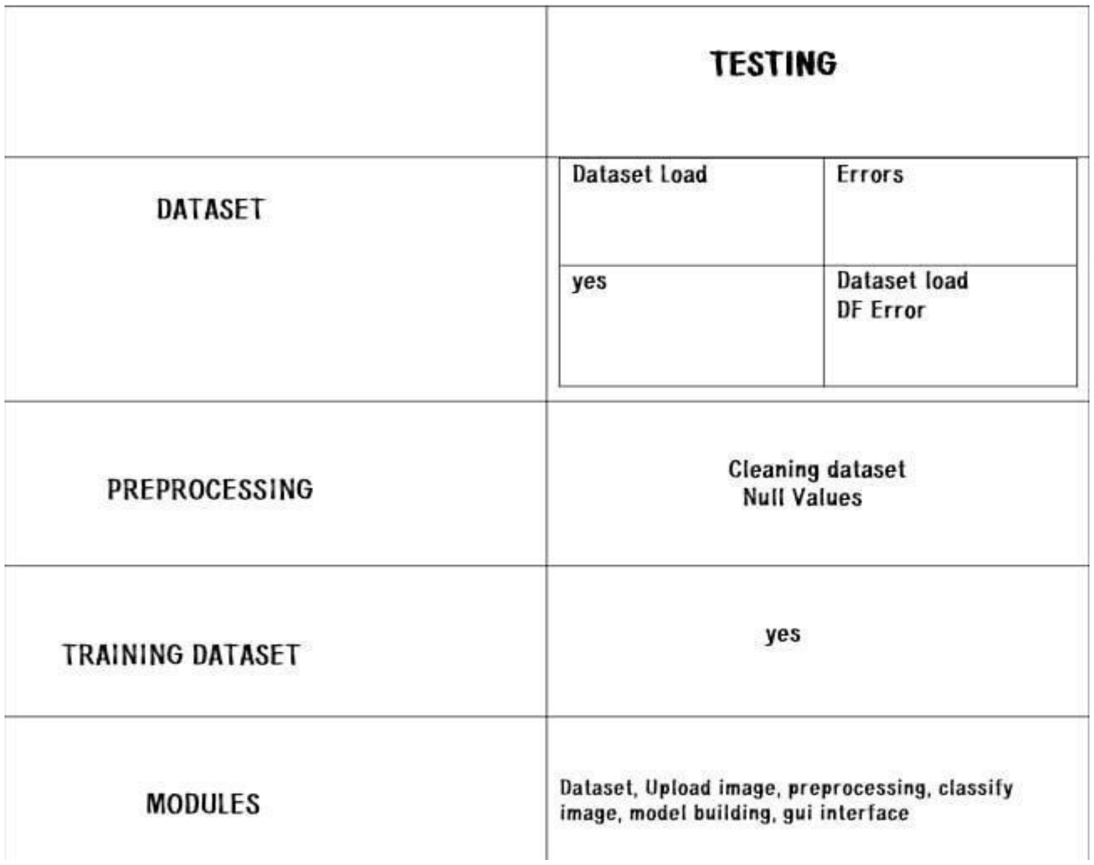
Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing.Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications,

e.g. components in a software system or one step up software applications at the company level interact without error.



	TESTING	
DATASET	Dataset Load	Errors
	yes	Dataset load DF Error
PREPROCESSING	Cleaning dataset Null Values	
TRAINING DATASET	yes	
MODULES	Dataset, Upload image, preprocessing, classify image, model building, gui interface	

Fig 7.1 Test Scenarios

8.OUTPUT SCREENS

OUTPUT SCREENS

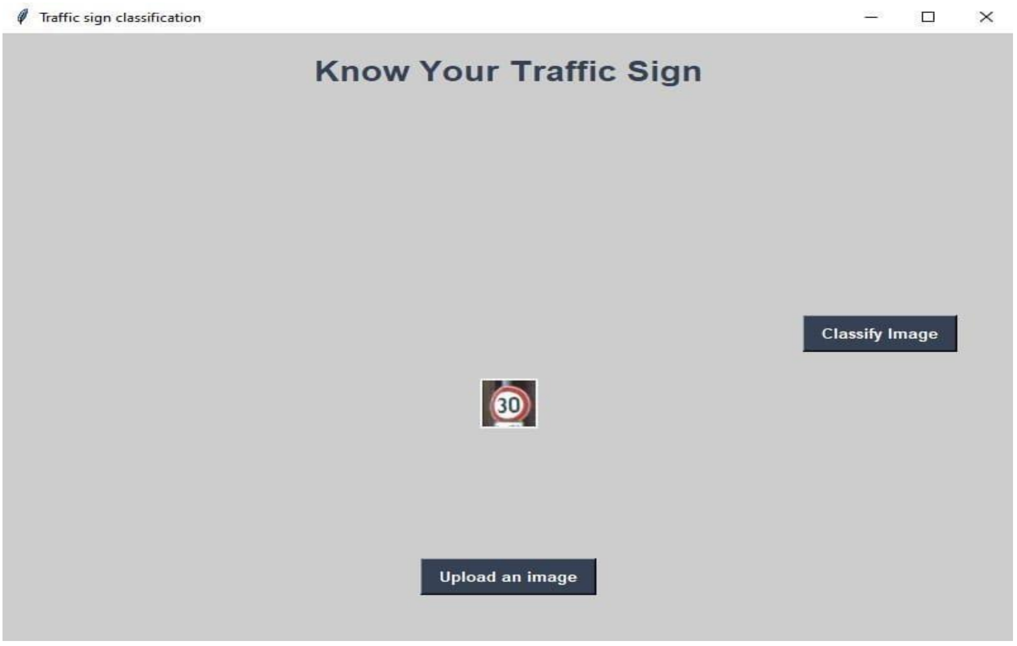


Fig 8.1 Opening image

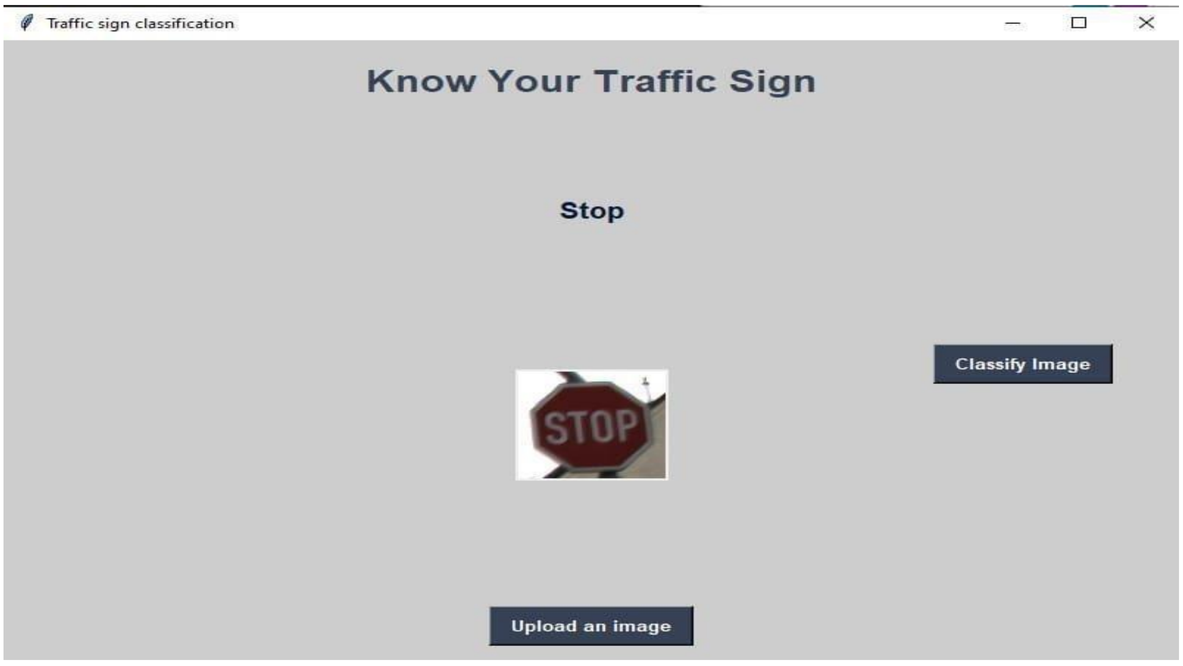


Fig 8.2 Traffic sign board detection

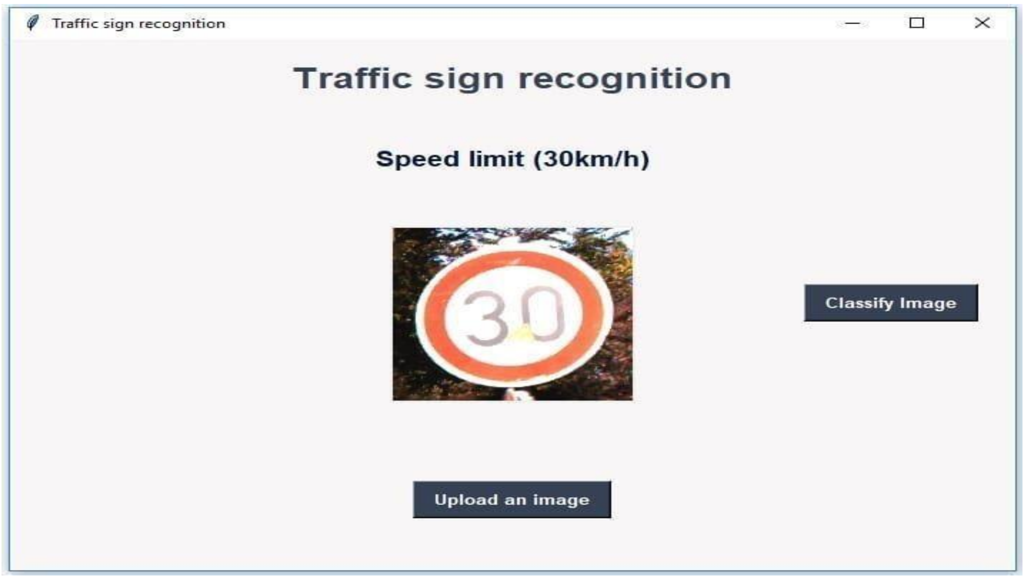
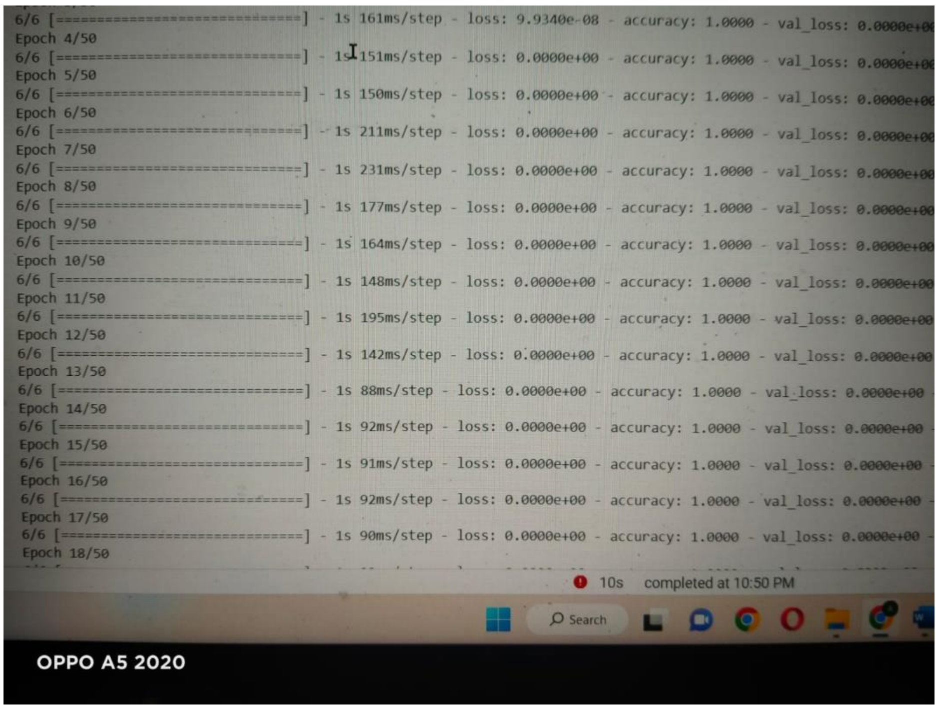


Fig 8.3 Sign board message alert



```
6/6 [==============================] - 1s 161ms/step - loss: 9.9340e-08 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/50
6/6 [==============================] - 1s 151ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/50
6/6 [==============================] - 1s 150ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/50
6/6 [==============================] - 1s 211ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/50
6/6 [==============================] - 1s 231ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/50
6/6 [==============================] - 1s 177ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/50
6/6 [==============================] - 1s 164ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/50
6/6 [==============================] - 1s 148ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 11/50
6/6 [==============================] - 1s 195ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 12/50
6/6 [==============================] - 1s 142ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 13/50
6/6 [==============================] - 1s 88ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
Epoch 14/50
6/6 [==============================] - 1s 92ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
Epoch 15/50
6/6 [==============================] - 1s 91ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
Epoch 16/50
6/6 [==============================] - 1s 92ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
Epoch 17/50
6/6 [==============================] - 1s 90ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 -
Epoch 18/50
```

Fig 8.4 Accuracy

9.CONCLUSION

CONCLUSION

This project proposed a system that is able to detect and classify a set of 28 traffic signs in different environments. The results are moderate and it can be improved by testing different neural network structures. As a neural network is often called a black box, there is no guarantee that it will perform best with the defined set of parameters stated above. New methods of data augmentation can also be applied to make the classifier more robust. Real- time detection and recognition can also be implemented in the future. The Traffic Sign Board Detection is implemented using Convolutional Neural Network. Various models under the CNN heading were studied and the one with highest accuracy on the GTSRB dataset was implemented. The creation of different classes for each Traffic sign has helped in increasing the accuracy of the model. A map is displayed on which the signs in the vicinity of the driver are displayed thus helping him/her take appropriate decisions. This paper is a significant advancement in the field of driving as it would ease the job of the driver without compromising on the safety aspect. Also this system can easily be implemented without the need of much hardware thus increasing its reach. The goal of this research is to develop an efficient TSDR system based on Malaysian traffic sign dataset. In the image acquisition stage, the images were captured by an on board camera under different weather conditions and the image preprocessing was done by using RGB colour segmentation. The recognition process is done by SVM with bagged kernel which is used for the first time for traffic sign classification. The developed system has shown promising results with respect to the accuracy of 95.71%, false positive rate (0.009), and processing time (0.43 s). The recognition performance is evaluated by using ROC curve analysis. The simulation results are compared with the existing methods showing the correctness of the implementation.

10. FUTURE SCOPE

FUTURE SCOPE

The prototype can be expanded to include an inbuilt alert system with a camera in the vehicle's centre. Also, the feature of getting the estimated time for reaching that particular traffic sign can be added. This system can also be expanded for identification of traffic signals and hence prompt the user about the time to reach that particular signal and its status as well. The user can accordingly plan their trip start time and hence cross all signals without having to wait. Also the driver verification will be done with the help of an API providing the information about the license holder and the license number. In the future, the recognition phase can be sped up using dimension reduction of feature vectors. The number and parameters of Gabor filters will greatly influence the results of the classification results. It is necessary to research the optimization of Gabor parameters using optimization algorithms, to improve the efficiency and accuracy of traffic sign detection and classification. Finally, we accelerated our method by a GPU and further improved the efficiency.

11.REFERENCES

REFERENCES

1. Zhong LIU, Weihai CHEN, Yuhua ZOU and Cun HU "Regions of Interest Extraction Based on HSV Color Space," IEEE 10th International Conference on Industrial Informatics, July 2012.

2. Dept. Transp., London, U.K., Traffic Signs Image Database, 2011

3. Jack Greenhalgh and Majid Mirmehdi "Real-Time Detection and Recognition of Road Traffic Signs," IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 13, NO. 4, DECEMBER 2012.

4.Jung-Guk Park, Kyung-Joong Kim "A METHOD FOR FEATURE EXTRACTION OF TRAFFIC SIGN DETECTION AND THE SYSTEM FOR REAL WORLD SCENE," IEEE International Conference on Emerging Signal Processing Applications, 12-14 Jan. 2012.

5. Sungho Kimh and Soon Kwon "Improvement of traffic sign recognition by accurate ROI refinement," 15th International Conference on Control, Automation and Systems (ICCAS 2015) Oct. 13-16, 2015 in BEXCO.

6. Hurriyatul Fitriyah, Edita Rosana Widasari, Gembong Edhi Setyawan "Traffic Sign Recognition using Edge Detection and Eigen-face," International Conference on Sustainable Information Engineering and Technology (SIET), 2017.

7. H. Fleyeh, E. Davami "Eigen-based traffic sign recognition," IET Intelligent Transport Systems (Volume: 5 Issue: 3 September 2011).

8. Ioan Cristian Schuszter "A Comparative Study of Machine Learning Methods for Traffic Sign Recognition," 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2017.