TRAFFIC SIGN DETECTOR

A project submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
INFORMATION TECHNOLOGY

Submitted by:
Nikhil Kumar Arora
Roll No.: 11912063
Group No- 213

Supervised by:
Dr. Mukesh Mann
Assistant Professor

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SONEPAT -131201,HARYANA, INDIA

ACKNOWLEDGEMENTS

First of all, I am immensely indebted to Almighty God for his blessings and grace without which I could not have undertaken this task and my efforts would never have been a success.

I humbly consider it a privilege and honor to express my heartiest and profound gratitude to Prof **(Dr) Mukesh Mann**, Assistant Professor -IT, and IIIT Sonepat, for his appropriate direction, valuable suggestion, under judging assistance so generously extended tome.

This guidance and support received from my entire classmates who contributed and who are contributing to this project, is vital for the success of this project. I am grateful for their constant support and help.

I also owe a sense of gratitude to my parents for their encouragement and support throughout the project.

Nikhil Kumar Arora

SELF DECLARATION

I hereby declare that the work contained in the project titled "**Traffic Sign Detector**" is original. I have followed the standards of project ethics to the best of my abilities. I have acknowledged all sources of information that I have used in the project.

Name: Nikhil Kumar Arora
Roll No.: 11912063
Department of Information Technology,
Indian Institute of Information Technology,
Sonepat-131201, Haryana, India.

CERTIFICATE

This is to certify **Nikhil Kumar Arora** has worked on the project entitled "**Traffic Sign Detector**" under my supervision and guidance.

The contents of the project, being submitted to the **Information Technology**, **IIIT Sonepat,** for the award of the degree of **B.Tech** in **Information Technology**, are original and have been carried out by the candidate himself. This project has not been submitted in full or part for the award of any other degree or diploma to this or any other university.

Dr. Mukesh Mann
Supervisor

Department of Information Technology,
Indian Institute of Information Technology,
Sonepat-131201, Haryana, India

ABSTRACT

Name of the student–**Nikhil Kumar Arora,** Roll No: **11912063,** Degree for which submitted **B.Tech (IT).,** Department of **Information Technology**, **IIIT, Sonepat**.

Project Title: **Traffic Sign Detector**

Name of the thesis supervisor: **Dr. Mukesh Mann**

Month and year of the project submission: **April 2021**

Traffic sign detection and classification are of paramount importance for the future of autonomous vehicle technology. In this paper, we propose a Convolutional Neural Network (CNN)-based approach for detecting and classifying traffic signs which is robust against such challenging environmental conditions. In the proposed model, our approach is to consecutively (i) selectively preprocess the image to enhance the sign features (ii) classify the sign proposals extracted from the image. These tasks are carried out by implementing CNNs architecture and depth. The proposed model has been evaluated on the Traffic-Sign Recognition Dataset available on the Kaggle website.

LIST OF ABBREVIATIONS

TSD	Traffic Sign Detector
S/W	Software
AI	ARTIFICIAL INTELLIGENCE
ML	MACHINE LEARNING

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 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 of a self-driving car is the ability to detect traffic signs 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 to provide a safe and secure environment to all the parties concerned. We have focused our project on traffic signs which we have in our dataset is as used in the project. We used the kaggle traffic sign dataset. The dataset consisted of different types of traffic signs. 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 being visible to the driver/system. The other side of the coin is that we have to contend with lighting and weather conditions. The main objective of our project is to design and construct a computer-based system that can automatically detect the road signs to assist the user or the machine so that they can take appropriate actions. 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.

This paper is organized as follows: Chapter 2 presents the related works in the field of Design of the TSD system. In Chapter3, the overall implementation is discussed. The software testing and validation results are discussed in Chapter4. In Chapter5, the conclusion and some suggestions are made for future improvement in the field of automatic traffic sign detection and recognition.

1.2 Problem Outline

To solve the concerns over road and transportation safety, automatic traffic sign detection and recognition (TSDR) system has been introduced. An automatic TSDR system can detect and recognize traffic signs from and within images captured by cameras. In adverse traffic conditions, the driver may not notice traffic signs, which may cause accidents. In such scenarios, the TSDR system comes into action. The main objective of the research on TSDR is to improve the robustness and efficiency of the TSDR system. Developing an automatic TSDR system is a tedious job given the continuous changes in the environment and lighting conditions. Among the other issues that also need to be addressed are partial obscuring, multiple traffic signs appearing at a single time, and blurring and fading of traffic signs, which can also create a problem for the detection purpose. For applying the TSDR system in a real-time environment, a fast algorithm is needed. As well as dealing with these issues, a recognition system should also avoid erroneous recognition of nonsigns. This project

aims to develop an efficient TSDR system that can detect and classify traffic signs into different classes in a real-time environment.

1.3 PURPOSE

Traffic-sign detection and classification is an interesting topic in computer vision and it is especially important in the context of autonomous vehicle technology. Robust and real-time traffic-sign detection algorithms have to be employed if self-driving cars are to become commonplace in the roads of the future.

Traffic sign detection and classification are of paramount importance for the future of autonomous vehicle technology. The main purpose of this project is to reduce the number of accidents during driving. During Driving Our 100% focus must be on driving not on seeing which traffic signs are around them. That why we develop a project that is Traffic sign Detector which detects traffic signs around them and tells what is the traffic sign.

1.4 Project Description

Nowadays, Self Driving cars are a trending topic. One of the many important aspects of a self-driving car is the ability to detect traffic signs to provide safety and security for the people not only inside the car but also outside of it.

The project is built using OpenCV, Python library to implement machine learning and deep learning principles in JupyterNotebookEditor.Keras Algorithm is used to build this deep learning model to detect the traffic-sign detector. The camera can be installed in the car after this, This model can work on real-time camera footage and detect the traffic sign and tell the driver what the traffic sign means. The model can predict up to certain accuracy about the traffic sign.

Deep learning is used to develop our traffic sign model. The architecture used for the object detection purpose is Single Shot Detector (SSD) because of its good performance accuracy and high speed**.** Alongside this, we have used basic concepts of transfer learning in neural networks to finally output the meaning of traffic signs in an image or a video stream. We aim to show that our model performs well on the test data with max% precision and recall, respectively.

1.5 PROJECT OBJECTIVES

1. **Live Traffic Sign Detection:** To identify the traffic sign and tell the meaning of it with the help of computer vision and deep learning.

2. **Max Accuracy:** To achieve maximum accuracy on detection, to do this data set is needed as big as possible

3. **Reduce Accident:** As this software detects the traffic sign so the driver will be able to focus 100% on driving.

ALL THE ABOVE LISTED OBJECTIVES WILL BE ACHIEVED BY USING SASHIMI MODEL

1.6 PROJECT METHODOLOGY –

All the objectives mentioned above are going to be achieved with the help of the **SASHIMI** model.

The first objective i.e.Live Traffic Sign Detection: To identify the traffic sign and tell the meaning of it with the help of computer vision and deep learning.

Our second objective is to maximize Accuracy: To achieve maximum accuracy on detection to do this we are going to take the data set as big as possible

Our second objective is to Reduce Accident: As this software detect the traffic sign so the driver will be able to focus 100% on driving. Installing the camera in the car and detecting traffic signs live using RasberiPi is the future goal of our project.

In this project, the changing of requirements is required and we are going to overlap the different S/W phases over each other. All the resources for the project are available to us and we wanted the project to get started soon, moreover, we wanted to shorten our time scale that’s why we used this model.

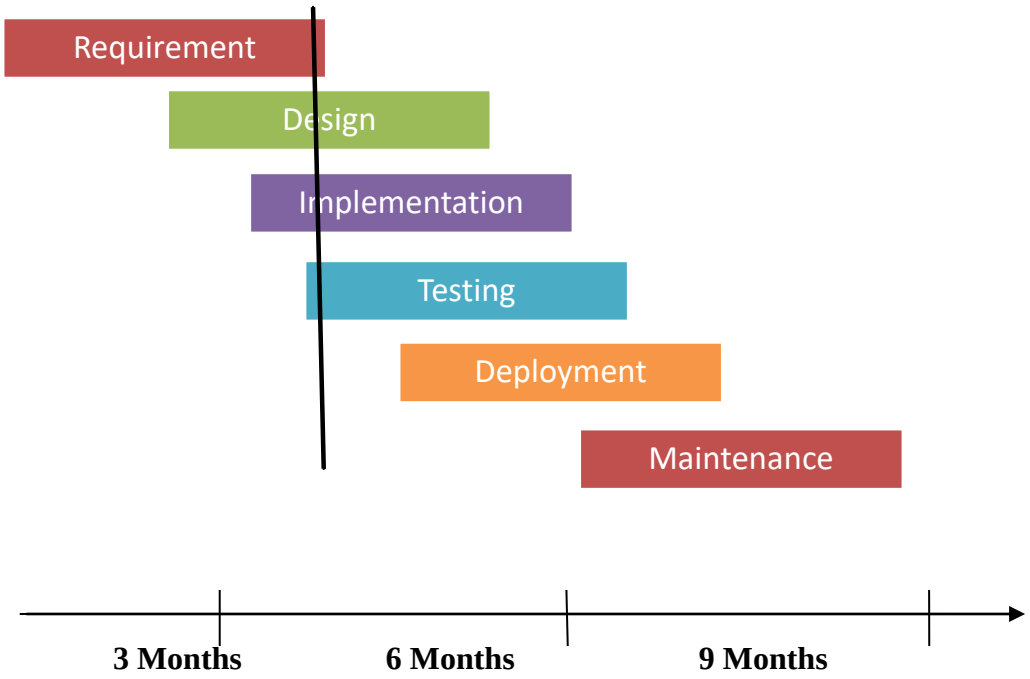


Fig 1.1 THE SASHIMI MODEL

Advantages of using the SASHIMI Model in this project-

- Shortens the development time, as different S/W layers are overlapped over each other.
- A 12-month project could be done in 9 months.
- People with different skills can start working on the project without waiting for the work done for the previous phase.
- The architect can start working on the project before even the analyst or who is doing the requirements is done.

Disadvantages of using the SASHIMI Model in this project-

- It may result in some rework because the design phase is started before all the requirements were done.
- It means that if something is found later, during the requirement phase, the design phase is then again needed to be adjusted.

If the coding phase is already started, then that requirement change may also result in some rework.

1.7 THE ORGANISATION OF PROJECT

- In Chapter 1, a brief Intro, problem outline, project objectives, methodology, the scope of TSD are discussed.
- In Chapter 2, the Study Design of TSD will be discussed.
- In Chapter 3, Implementation will be discussed– a tool used in the development of this project.
- In Chapter 4, Software testing and validation will be discussed.
- In Chapter 5, the Conclusion and Future of this Project will be discussed.

1.8SUMMARY

In this chapter, A brief introduction to Traffic Sign Detector is given. Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the many important aspects of a self-driving car is the ability to detect traffic signs to provide safety and security for the people not only inside the car but also outside of it.

After that, the problem outline is discussed with the purpose of the project, its goals, and its description which tells about why we selected this project.

Project Objectives are discussed afterward which are Live TrafficSignDetection, achieving Max Accuracy, and Reducing Accidents by taking large data set as much as possible.

Project Methodologies are also discussed to achieve the above-mentioned objectives in the future. THE SASHIMI MODEL will be used in the project methodology.

Further future scopes for Traffic Sign Detector are also discussed in this chapter. The aim is to implement this software in the car.

The organization of the project is discussed at the end in which a brief intro of all the chapters have been given

CHAPTER - 2

DESIGN

2.1Introduction –

TSD will be used to detect Traffic Signs. In this chapter, we will be discussing general components for TSD and designing parts like Development Platform tools used, Use Case Diagrams, Data Flow Diagrams, Activity Diagrams. At last, We Give a Summary of this Design Chapter.

2.2Diagrams -

2.2.1 Activity Diagram:

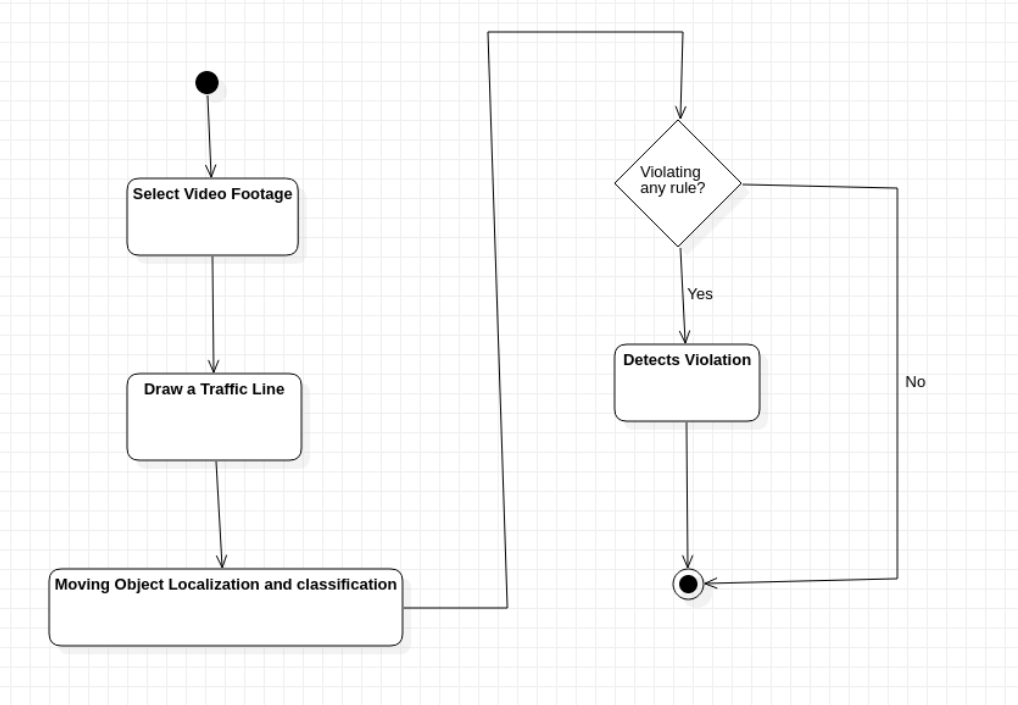


Fig 2.1 Activity Diagram

2.2.2 Data Flow Diagram :

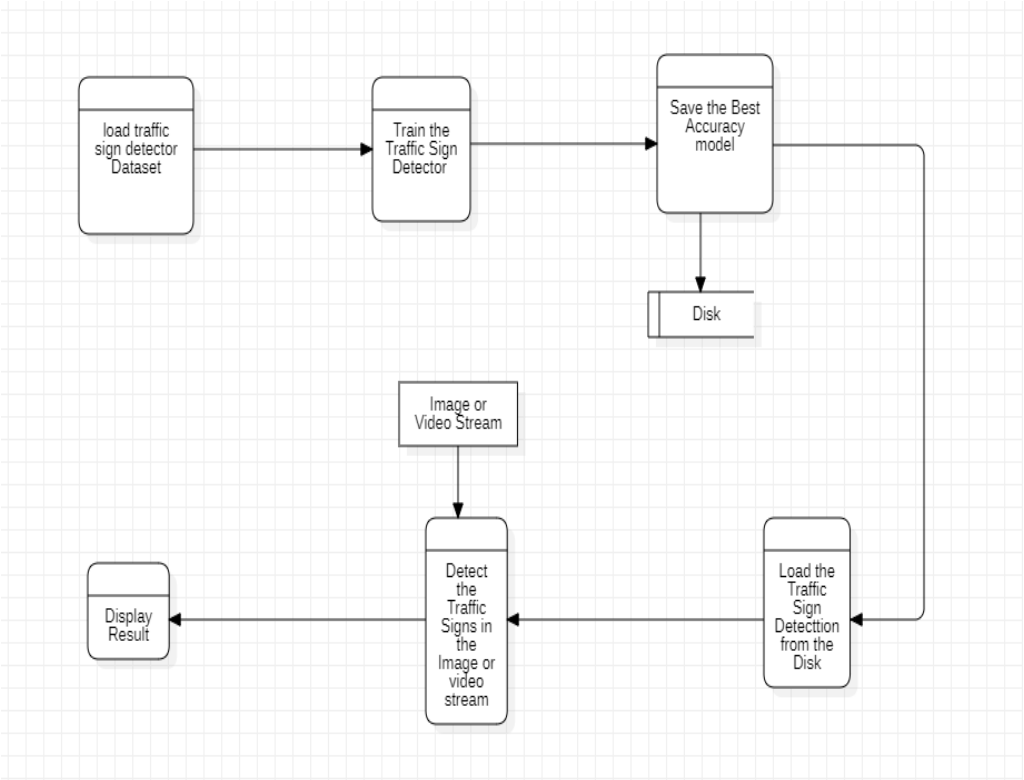


Fig 2.2 Data Flow Diagram

2.2.3 Use Case Diagram :

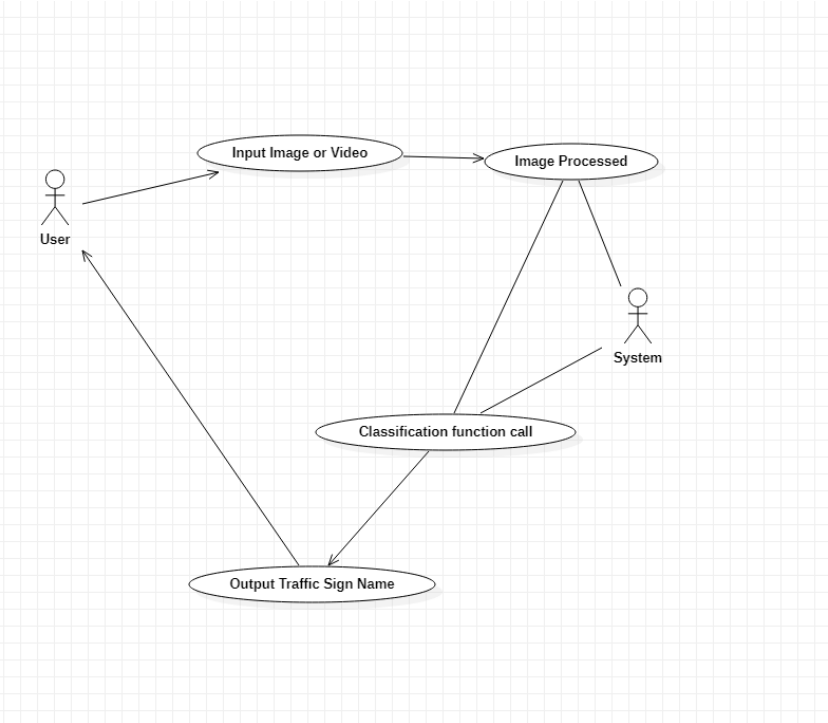


Fig 2.3 Use Case Diagram

2.3 DEVELOPMENT PLATFORM-TOOLS –

The following development tools are needed for the development of this project
1. Star UmlFor Developing Various Diagrams related to this project.
2. Python, as the programming language.
3. Jupyter Notebook for Coding purpose.
4. GitHub for collaboration purposes.

1) STAR UML -

FIG 2.7 STAR UML

StarUML is a sophisticated software modeler for agile and concise modeling. It is compatible with diagrams: Class, Object, Use Case, Component, Deployment, Composite Structure, Sequence, Communication, Statechart, Activity, Timing, Interaction Overflow, Information Flow, and Profile Diagram. It is a UML tool by MKLab. The software was licensed under a modified version of GNU GPL until 2014 when a rewritten version 2.0.0 was released for beta testing under a proprietary license.

2) PYTHON LANGUAGE -

FIG 2.8 PYTHON

Python 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.

3) JupyterNotebbok -

FIG 2.9 Jupyter Notebook

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

4) GITHUB -

FIG 2.10 GITHUB

GitHub, Inc. is a subsidiary of Microsoft which provides hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration, and wikis for every project.[3] Headquartered in California, it has been a subsidiary of Microsoft since 2018.

2.4 **SUMMARY** –

This chapter talks about the design part of the Traffic Sign Detector. The working of TSD is also discussed in this chapter. General components of TSDi.e End Users, Front-end-interface.

This chapter also talks about the system Design of TSD in which a brief discussion is there for the total design of the TSD.

The total design consists of two phases: training and testing

Then a brief discussion of the Data Flow Diagram, Activity Diagram, and Use Case Diagram is also there which tells the depth working of TSD.

In the end, development tools used in TSD are discussed.

CHAPTER 3

IMPLEMENTATION

3.1 INTRODUCTION

Implementation i.e how TSD is implemented is discussed in this chapter.

3.2 List of Libraries Used:

- **Tensorflow:** TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Fig 3.1

- **Keras:** Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

Fig 3.2

- **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

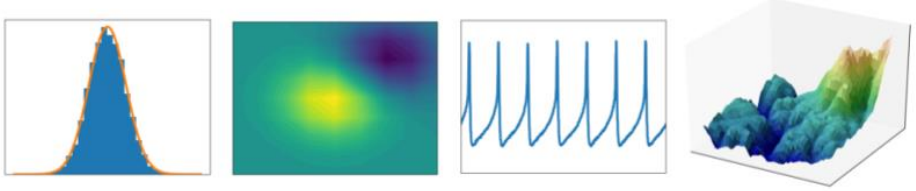


Fig 3.3

Matplotlib 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 shell, web application servers, and various graphical user interface toolkits.

- **Numpy:** NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python.

Fig 3.4

- **Open-cv:** OpenCV was started at Intel in 1999 by Gary Brodsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Brodsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Brodsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

  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 is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

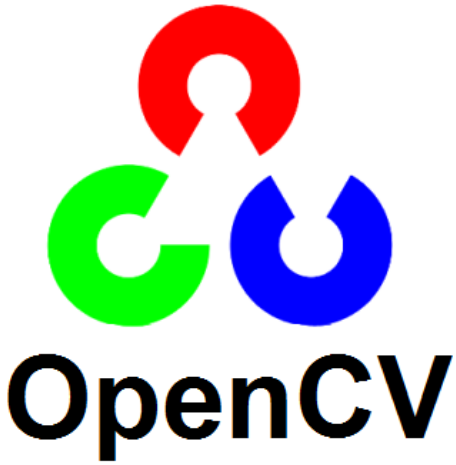


Fig3.5

- **Imutils:** A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much easier with OpenCV and both Python 2.7 and Python 3.

3.3 The Architecture of Lenet-5 -

Suppose you want to learn a new subject. There are two ways to do that. One is to buy some books and start reading everything from scratch. The other way round is to go to a teacher and he will share all the knowledge and experience he has gained. Which process is faster and easier as well. The second one i.e taking the help of a teacher.

Transfer learning works similarly. Transfer learning is the method that uses a neural network trained on a large and generalized enough dataset and is being used for another problem. These neural networks are called Pre-trained networks.

The basic requirement for transfer learning is the availability of a pre-trained network. Luckily, we have several state-of-the-art deep learning networks shared by the respective teams. Out of these pre-trained networks, in this article, we are going to discuss Lenet-5 in detail.

3.4 What is Lenet5?

Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.

The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification.

3.5 Some features of Traffic Sign Detector:-

Different types of sign detecions:-

Figure 3.6 Speed Limit Sign Detection

Figure 3.7 No passing Sign Detection

Figure 3.8 Turn Right Ahead Sign Detection

Figure 3.9 Wild Animals Crossing Sign Detection

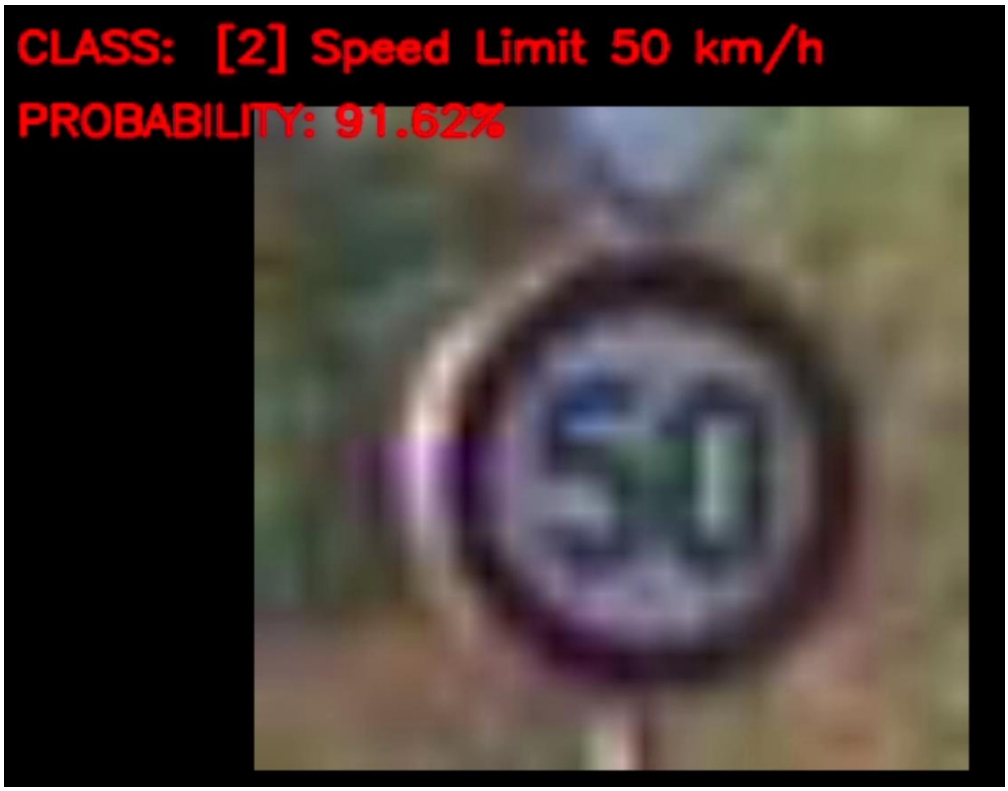
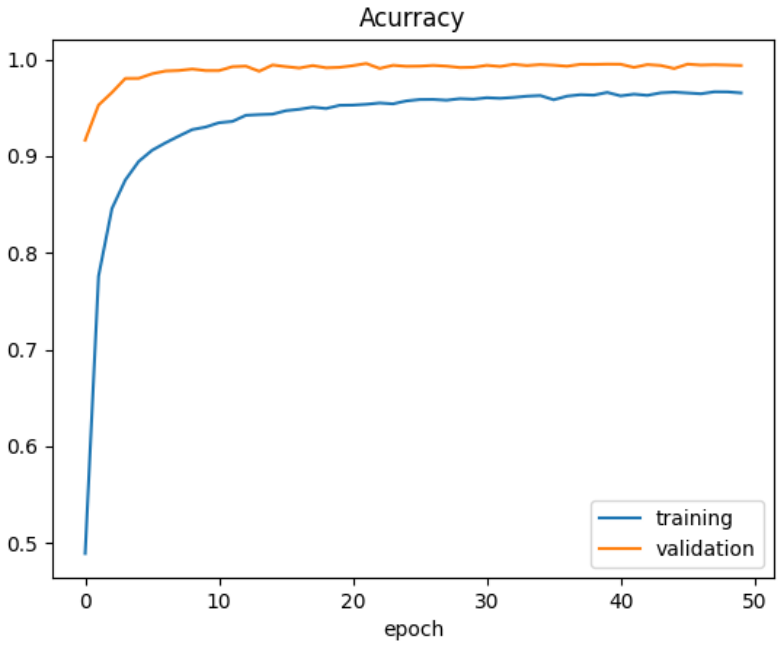


Figure 3.10 Speed limit Sign Detection (More blur image)

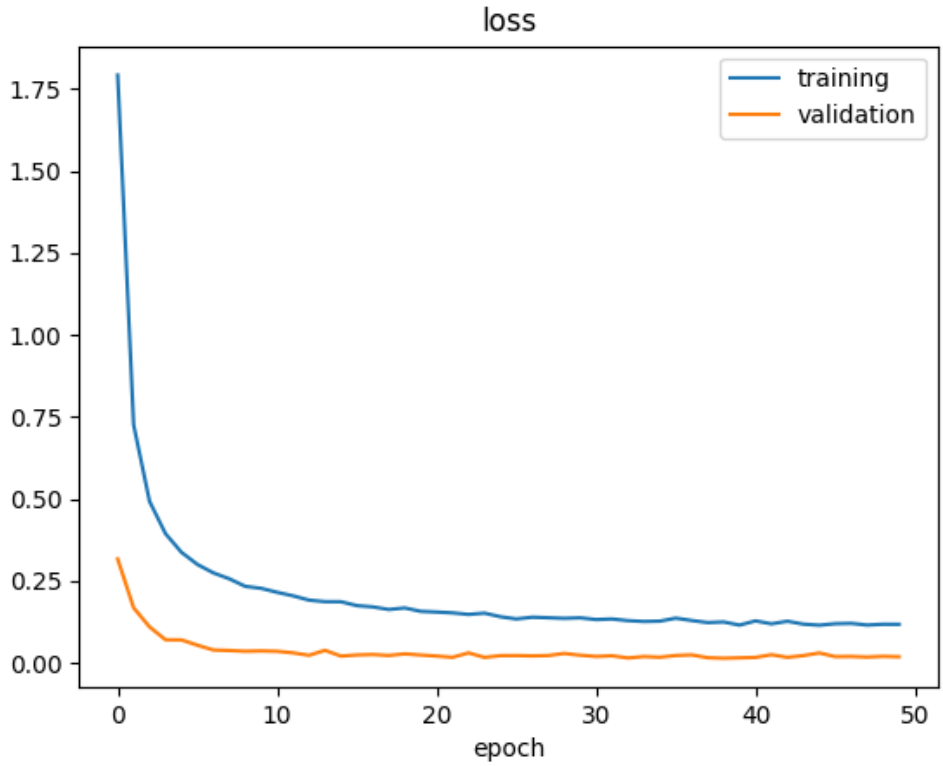
Figure 3.11 No passing Sign Detection (More Blur image)

3.6 Some Graphs Generated

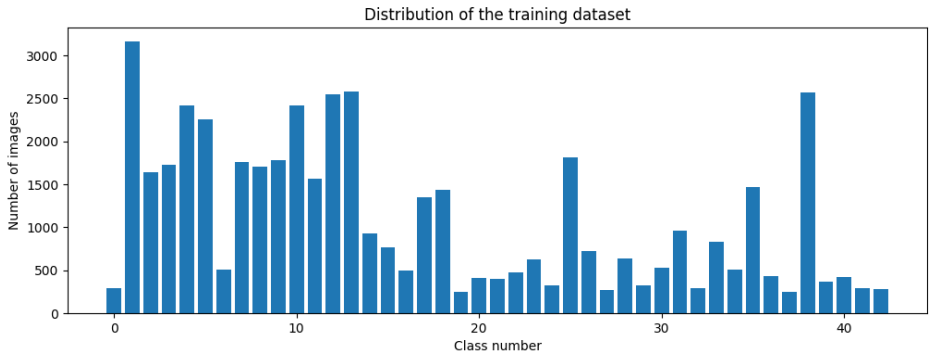
3.6.1 Accuracy Graph



3.6.2 Loss graph



3.6.3 Distribution of the training dataset



3.7 SUMMARY – In this chapter, the implementation of TSD is discussed. Some main features are also discussed.

CHAPTER – 4

TESTING AND VALIDATION

4.1 Software Testing - Validation Testing -

Validation is the process of examining whether or not the software satisfies user requirements. It is carried out at the end of the SDLC. If the software matches the requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes user requirements.

4.2 Software Verification -

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications ?"
- Verifications concentrate on the design and system specifications.

The target of the test is -

- **Errors** - These are actual coding mistakes made by developers. In addition, there is a difference in the output of software and the desired output is considered as an error.
- **Fault** - When an error exists fault occurs. A fault, also known as a bug, is a result of an error that can cause the system to fail.
- **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when a fault exists in the system.

5 Manual Vs Automated Testing

4.3 Testing can either be done manually or using an automated testing tool:

- **Manual** - This testing is performed without taking the help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests, and reports the result to the manager.

  Manual testing is time and resource-consuming. The tester needs to confirm whether or not the right test cases are used. A major portion of testing involves manual testing.

- **Automated** This testing is a testing procedure done with aid of automated testing tools. The limitations of manual testing can be overcome using automated test tools.

A test needs to check if a webpage can be opened in Internet Explorer. This can be easily done with manual testing. But to check if the webserver can take the load of 1 million users, it is quite impossible to test manually.

There are software and hardware tools that help testers in conducting load testing, stress testing, regression testing.

4.4 Testing Approaches -

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation into concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

Exhaustive tests are the best-desired method for perfect testing. Every single possible value in the range of the input and output values is tested. It is not possible to test each and every value in a real-world scenario if the range of values is large.

4.4.1 Black-box testing -

It is carried out to test the functionality of the program. It is also called 'Behavioral' testing. The tester, in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

Fig 4.1 Black-Box Testing

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end-users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower-end values. If these values pass the test, it is assumed that all values in between may pass too.

- **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested systematically.
- **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** - The system changes state on the provision of input. These systems are tested based on their states and input.

4.4.2 White-box testing -

It is conducted to test the program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing.

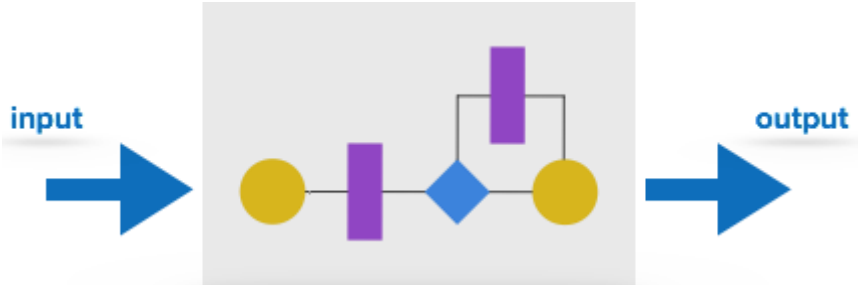


Fig 4.2 White-Box testing

In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

Below are some White-box testing techniques:

- **Control-flow testing** - The purpose of control-flow testing is to set up test cases that cover all statements and branch conditions. The branch conditions are tested for both being true and false so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasis covering all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

4.4.3 Testing Levels -

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping to the next stage, a stage is tested, validated, and verified.

Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels -

4.4.4 Unit Testing -

While coding, the programmer performs some tests on that unit of the program to know if it is error-free. Testing is performed under the white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error-free.

4.4.5 Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated would also work without errors. For example, argument passing and data updation, etc.

4.4.6 System Testing

The software is compiled as a product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do the desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environmental conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and is accessed by a number of persons.

4.4.7 Acceptance Testing

When the software is ready to hand over to the customer it has to go through the last phase of testing where it is tested for user interaction and response. This is important because even if the software matches all user requirements and if the user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developers themselves performs alpha testing by using the system as if it is being used in the work environment. They try to find out how the user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use under their production environment only for testing purposes. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

4.4.8 Regression Testing

Whenever a software product is updated with new code, features, or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

4.5 Testing Documentation

Testing documents are prepared at different stages -

Before Testing

Testing starts with test case generation. Following documents are needed for reference –

- **SRS document** - Functional Requirements document

- **Test Policy document** - This describes how far testing should take place before releasing the product.
- **Test Strategy document** - This mentions detailed aspects of the test team, responsibility matrix, and rights/responsibility of the test manager and test engineer.
- **Traceability Matrix document** - This is an SDLC document, which is related to the requirement gathering process. As new requirements come, they are added to this matrix. These matrices help testers know the source of requirements. They can be traced forward and backward.

While Being Tested

The following documents may be required while testing is started and is being done:

- **Test Case document** - This document contains the list of tests required to be conducted. It includes the Unit test plan, Integration test plan, System test plan, and Acceptance test plan.
- **Test Description** - This document is a detailed description of all test cases and procedures to execute them.
- **Test case report** - This document contains a test case report as a result of the test.
- **Test logs** - This document contains test logs for every test case report.

After Testing

The following documents may be generated after testing :

- **Test summary** - This test summary is the collective analysis of all test reports and logs. It summarizes and concludes if the software is ready to be launched. The software is released under a version control system if it is ready to launch.

4.6 Testing vs. Quality Control, Quality Assurance, and Audit

We need to understand that software testing is different from software quality assurance, software quality control, and software auditing.

- **Software quality assurance** - These are software development process monitoring means, by which it is assured that all the measures are taken as per the standards of the organization. This monitoring is done to make sure that proper software development methods were followed.
- **Software quality control** - This is a system to maintain the quality of software products. It may include functional and non-functional aspects of the software product, which enhance the goodwill of the organization. This system makes sure that the customer is receiving a quality product for their requirement and the product certified as 'fit for use'.
- **Software audit** - This is a review of the procedure used by the organization to develop the software. A team of auditors, independent of the development team examines the software process, procedure, requirements, and other aspects of SDLC. The purpose of a software audit is to check that software

and its development process, both conform to standards, rules, and regulations.

4.7 TESTING TECHNIQUE USED FOR TRAFFIC SIGN DETECTOR

Manual testing is done for this project as it was not possible to automate the testing for this project. Testing is done for the detection of various types of masks and different combinations

4.8 FAIL CASE:-

The one failure of the Traffic Sign Detector is it can not work on the very low-quality camera.

4.9 SUMMARY –

Various software testing techniques are described in this chapter. Manual Testing is done for this project as automation for Traffic Sign Detector was not possible.

CHAPTER - 5

CONCLUSION AND FUTURE SCOPE

5.1 LIMITATION

The identification of traffic-sign detector is the data set, computation power, lighting conditions. However, the development of a Traffic Sign Detector condition identification network is challenging for several reasons.

The limitation in datasets is one main challenge. The Traffic sign detector conditions datasets are generally small, and their image quality is not high enough. Furthermore, the various performances of Traffic Sign Detection with different light conditions largely increase the difficulty of identification. There are a few limitations to our study. Firstly, the traffic Sign Dataset we used for classification identification is relatively small, where it cannot cover all types of signs used.

Also, the dataset does not contain video, where the identification result on a video stream cannot be tested. Another limitation is the excessive data loading time in Jupiter Notebook while loading the dataset into it.

5.2 SCOPE OF PROJECT

This project can be implemented in a car. Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the many important aspects of a self-driving car is the ability to detect traffic signs 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 to provide a safe and secure environment to all the parties concerned.

5.3 FUTURE GOALS

- Integrate with the voice system
- Collecting more dataset
- Integrate with RasberiPi using GPIOS coding
- After that coding integrates with the car system.

REFERENCES

1.Github ,accessed 1 October 2021, https://en.wikipedia.org/wiki/GitHub

2.Software Process Models, accessed 1 October 2021, https://www.thomasalspaugh.org/pub/fnd/softwareProcess.html

3.Keras, accessed 1 October 2021, https://en.wikipedia.org/wiki/Keras

4.Unified Modeling Language (UML) | Activity Diagrams, accessed 1 October 2021, https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

5. Software Development models, accessed 1 October 2021, https://mohamadkoteich.com/2019/01/30/software-development-models/

6. StarUml, accessed 2 October 2021, https://en.wikipedia.org/wiki/StarUML

7.Python, accessed 3 October 2021, https://en.wikipedia.org/wiki/Python_(programming_language)

8. OpenCv, accessed 3 October 2021, https://en.wikipedia.org/wiki/OpenCV

9. Jupyter Notebook, accessed 4 October 2021, https://jupyter.org/

10.smart Vehicles & IoT, accessed 4 October 2021,

https://www.intelligenthq.com/smart-vehicles-the-internet-of-things/

APPENDIX

GITHUB LINK- https://github.com/Nikhil9958/Traffic-Sign-Detector