

Traffic sign recognition

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

by

KARNAPU

SAINATHREDDY(Reg.

No. 37110322)

K.NIKHIL REDDY(Reg.

No. 37110358)



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

SATHYABAMA

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

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

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

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KARNAPU SAINATH REDDY (Reg. No. 37110322)** and **K.NIKHIL REDDY (Reg.**

No.37110358) who

carried out the project entitled "**TRAFFIC SIGN RECOGNITION**" undermy supervision from **August 2020** to **March 2021**.

Internal Guide

Mr.r.sethuraman M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **KARNAPU SAINATH REDDY** and **K.NIKHIL REDDY** hereby declare that the Project Report entitled “**TRAFFIC SIGN RECOGNITION**” is done by me under the guidance of **MR.R.SETHURAMAN . M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **MR.R.SETHURAMAN M.E, Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Traffic sign recognition (TSR) represents an important feature of advanced driver assistance systems, contributing to the safety of the drivers, pedestrians and vehicles as well. Developing TSR systems requires the use of computer vision techniques, which could be considered fundamental in the field of pattern recognition in general. Despite all the previous works and research that has been achieved, traffic sign detection and recognition still remain a very challenging problem, precisely if we want to provide a real time processing solution. In this paper, we present a comparative and analytical study of the two major approaches for traffic sign detection and recognition.

We propose an approach for traffic sign detection based on Convolutional Neural Networks (CNN). We first transform the original image into the gray scale image by using support vector machines, then use convolutional neural networks with fixed and learnable layers for detection and recognition. The fixed layer can reduce the amount of interest areas to detect, and crop the boundaries very close to the borders of traffic signs. The learnable layers can increase the accuracy of detection significantly.

TABLE OF CONTENTS

	ABSTRACT	v
CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 INTROCUCTION	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 DOMAIN OVERVIEW	1
	1.3.1 MACHINE LEARNING	1
2.	LITERATURE SURVEY	6
3.	SYSTEM ANALYSIS	8
	3.1 EXISTING SYSTEM	13
	3.2 DRAWBACKS FOR EXISTING SYSTEM	13
	3.3 PROPOSED SYSTEM	13
	3.4 ADVANTAGES OF PROPOSED SYSTEM	14
4.	REQUIRES SPECIFICATION	15
	4.1 INTRODUCTION	15
	4.2 HARDWARE & SOFTWARE SPECIFICATION	15
	4.3 SOFTWARE DESCRIPTION	16
	4.3.1 ABOUT PYTHON	16
	4.3.2 ABOUT IDE	25
5.	SYSTEM DESIGN	29
	5.1 INTRODUCTION	29
	5.2 ARCHITECTURE DIAGRAM	29

6.	SYSTEM IMPLEMENTATION	31
	6.1 LIST OF MODULES	31
	6.1.1 MODULE 1: EXPLORATORY DATA EVALUATION	31
	6.1.2 MODULE 2: PRE- PROCESSING	31
	6.1.3 MODULE 3: FEATURE ENGINEERING	32
	6.1.4 MODULE 4: PREDICTION	33
7.	SOFTWARE TESTING	34
	7.1 GENERAL	38
	7.2 TEST DRIVEN DEVELOPMENT	39
	7.3 UNIT TESTING	40
	7.4 BLACKBOX TESTING	41
	7.5 INTEGRATION TESTING	41
	7.6 SYSTEM TESTING	42
	7.7 SANITY TESTING	44
	7.8 REGRESSION TESTING	45
8.	RESULT & ANALYSIS	49
9.	CONCLUSION AND FUTURE ENHANCEMENT	52
	REFERENCES	53
	APPENDIX	
	A. PLAGIARISM REPORT	56
	B. JOURNAL PAPER	60
	C. SOURCE CODE	64

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
1.3.1.1	CLASSIFICATION	3
1.3.1.2	SUPERVISED LEARNING	4
3.1	WATERFALL MODEL	9
3.2	RAD MODEL	10
4.3.1	PYTHON	16
4.3.1-2	PYTHON IMPLEMENTATING MACHINE LEARNING	17
5.1.1	ARCHITECTURE DIAGRAM	30
7.1	IMPLEMENTATION	37
7.2	BENEFITS OF TESTING	39
7.3	INTEGRATION TESTING	42
7.4	SYSTEM TESTING	43
7.5	PERFORMANCE TESTING	47
8.1	TRAINING & CROSS VALIDATION GRAPH	49
8.2	ANAMOLY SCORE CURVE	49
8.3	HEAT MAP	50
8.4	VARIANCE & NUMBER OF COMPONENTS CURVE	51

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Recently, well-known automobile companies like BMW, Mercedes-Benz, and etc., are enthusiastically developing ADAS (Advanced Drive Assistance System). Mobileye, technology company that develops vision-based ADAS, commercialize ADAS system equipped not only LKAS (Lane Keeping Assist System), but also TSR (Traffic Sign Recognition) system for speed limit signs. And according to EURO NCAP rating policy, car should be able to warn the driver when the car's speed is above the set speedthreshold. Speed limit value could be also known by digital map data, but this method is restricted to roads that are parts of the map data. Therefore, vision-based traffic sign recognition system is needed for other cases.

Also, it is thought that various types of traffic sign should be recognized to get more detailed information of road. However, it is hard to make a specific country version traffic sign recognition system (except speed limit signs) because only US and German traffic sign datasets are available freely.

The art deep learning technology is considered as a main algorithm to recognize traffic signs. Especially in the deep learning architecture, convolutional neural network (hereinafter referred to as CNN) has been widely used for image classification problem. CNN is a kind of multilayer perceptron that uses small sub-region(called a receptive field). Receptive fields are tiled to cover the entire input image, then produce feature maps by share the same weight and bias. So, it could reduce the number of parameters needed for neural network learning, and increases learning efficiency.

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMAIN OVERVIEW

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is

able to identify patterns in order to make predictions about the future.

Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

1.3.1 MACHINE LEARNING

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model,

it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows-

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

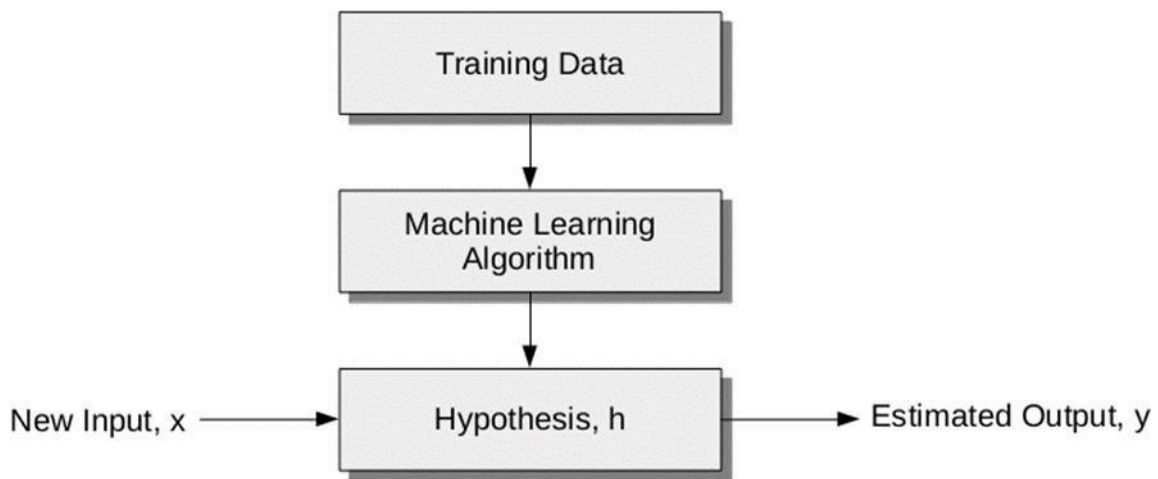


FIG 1.3.1 1 CLASSIFICATION

Supervised Learning

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price

predictor might take not only square-footage (x1) but also number of bedrooms (x2), number of bathrooms (x3), number of floors (x4), year built (x5), zip code (x6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.

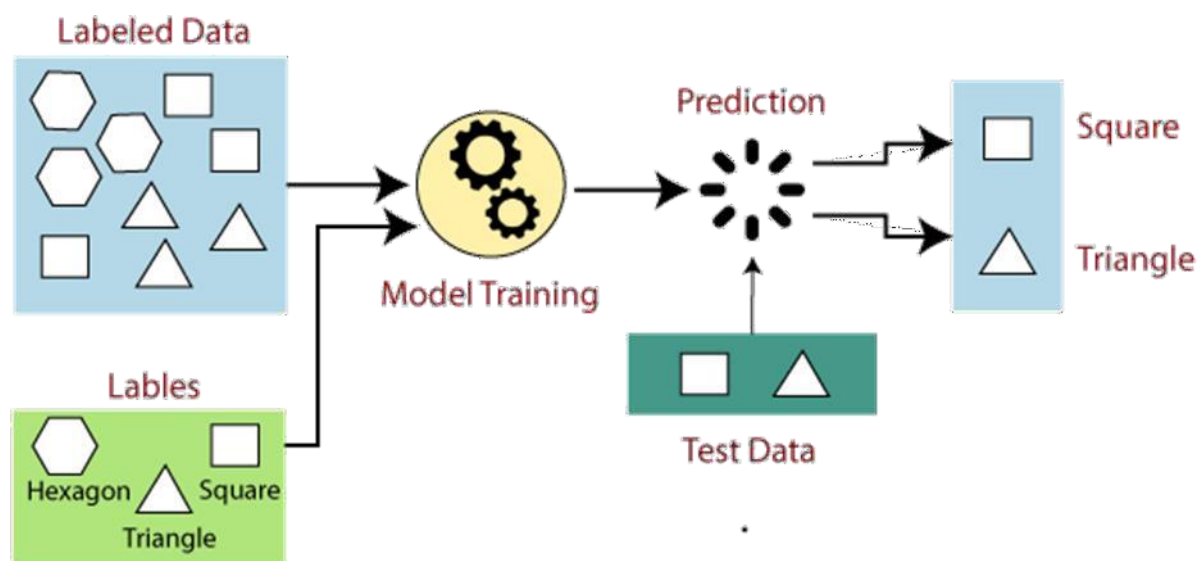


FIG 1.3.1 2 SUPERVISED LEARNING

Steps involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/gather the labelled training dataset.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather

forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering
- Random Forest
- Decision Trees
- Logistic Regression
- Support Vector Machines

CHAPTER 2

LITERATURE SURVEY

Behavioural-Based Classification and Identification of Ransomware Variants Using Machine Learning

Due to the changing behavior of ransomware, traditional classification and detection techniques do not accurately detect new variants of ransomware. Attackers use polymorphic and metamorphic techniques to avoid detection of signature-based systems. We use machine learning classification to identify modified variants of ransomware based on their behavior. To conduct our study, we used behavioral reports of 150 ransomware samples from 10 different ransomware families. Our data-set includes some of the newest ransomware samples available, providing an evaluation of the classification accuracy of machine learning algorithms on the current evolving status of ransomware. An iterative approach is used to identify optimum behavioral attributes used to achieve best classification accuracy. Two main parts of this study are identification of the behavioral attributes which can be used for optimal classification accuracy and classification of ransomware using machine learning algorithms. We have evaluated classification accuracy of three machine learning classification algorithms.

Detection of Malicious Software on Based on Multiple Equations of API-calls Sequences

Development and dissemination of malicious software requires the creation of new methods for their detection. Therefore, we began to use proactive technologies that use the test program to detect the presence of certain symptoms, often occurring in malware. Dynamic analysis of the studied program launched for execution. There is a study of how the program interacts with the software environment that is read/write at certain registry keys, files, network activity the use of certain API calls. Due to the fact that studied the program is potentially harmful, to make its execution must be in an isolated environment. This paper discusses proactive methods based on API call analysis and propose a new method using a multiple sequence alignment to identify common traits in malware. The paper considers the scheme to detect malicious software, based on API calls, each of which is implemented in software. Also presented a completely new malware detection scheme based on multiple sequence API calls alignment. This scheme is described in detail and implemented in software. A test on a set of software and the legitimacy of the viral nature. Testing has shown that the established scheme of competitive shows and identifies malicious software with high accuracy.

The impact of malware evolution on the analysis methods and infrastructure

The huge number of malwares introduced each day demands methods and tools for their automated analyses. Complex and distributed infrastructure of malicious software and new sophisticated techniques used to obstruct the analyses are discussed in the paper based on real-life malware evolution observed for a long time. Their impact on both toolsets and methods are presented based on practical development of systems for malware analyses and new features for existing tools.

Detection of Malicious Software on Based on Multiple Equations of API-calls Sequences

Development and dissemination of malicious software requires the creation of new methods for their detection. Therefore, we began to use proactive technologies that use the test program to detect the presence of certain symptoms, often occurring in malware. Dynamic analysis of the studied program launched for execution. There is a study of how the program interacts with the software environment that is read/write at certain registry keys, files, network activity the use of certain API calls. Due to the fact that studied the program is potentially harmful, to make its execution must be in an isolated environment. This paper discusses proactive methods based on API call analysis and propose a new method using a multiple sequence alignment to identify common traits in malware. The paper considers the scheme to detect malicious software, based on API calls, each of which is implemented in software. Also presented a completely new malware detection scheme based on multiple sequence API call alignment. This scheme is described in detail and implemented in software. A test on a set of software and the legitimacy of the viral nature. Testing has shown that the established scheme of competitive shows and identifies malicious software with high accuracy.

Literature Review

S . N o	PAPER DESCRIPTION	ALGORITHM USED	DATASET USED	PERFORMANCE MEASURE	PROBLEM IN EXISTING SYSTEM	PROPOSED METHOD
1	Y. Aoyagi, T. Asakura A study on traffic sign recognition in scene image using genetic algorithms and neural networks 22nd International Conference on Industrial Electronics, Control, and Instrumentation, IEEE, August (1996)	GA with the searching ability is applied for this problem and a circle pattern in the image is matched by giving the gene information.	Traffic sign from a video image	among the 24 pattern other than the speed sign, it was recognized that only one pattern was not a sign	image pattern recognition has been chiefly researched only for an individual object. However, it is an advanced direction to recognize the object which becomes a target from a scene image, with the development of the visual system of the robot [1].	The Technique based on GA [7] is proposed to recognize a circle traffic sign by using only brightness. This method is thought as a searching problem from the scene image. By treating both the position and the size of the traffic sign as gene information, the method detecting the traffic sign is developed.
2	Real time traffic sign detection using color and shape-based features T. Le, ST Tran, S. Mita, TD Nguyen - Asian Conference on Intelligent ..., 2010 - Springer	Iterative training algorithm and AdaBoost algorithm	12132 images from the eight cameras	this proved to yield a speed of 2 fps at a level of 96% detection rate and false positives per image	(i) The selective extraction of windows of interest, followed by their Classification (2) Exhaustive sliding window based classification	We proposed a multi-view scheme, which combines 2D and 3D analysis and efficient bounded evaluation of linear AdaBoost-like classifier

3	Road sign detection and recognition using matching pursuit method SH Hsu, CL Huang - Image and Vision Computing, 2001 - Elsevier	algorithm is based on the results of color segmentation method followed by the template matching.	(a) 30 triangular road signs; and (b) 10 circular road signs	Triangular road signs :94% Circular road signs :91%	Edges are tested at different levels of resolution by using so-called a Hierarchical Structure Code. It is assumed that closed edge contours are available at one of these levels of resolution, and failures happen when the outline of the traffic sign merges with the background.	To decrease the likelihood that signs are misidentified, the MP-training algorithm searches for a basis that separates the L1 coefficient vectors between different classes of road signs
4	Traffic Sign Detection and Recognition using Fully Convolutional Network Guided Proposals Yingying Zhu, Chengquan Zhang, Duoyou Zhou, Xinggang Wang, Xiang Bai, Wenyu Liu, Elsevier.	training fully convolutional network (FCN) using bounding box level supervision	Swedish Traffic Signs Dataset (STSD)	average precision of 98.67%	color-based methods, shape-based methods and sliding window based methods.	EdgeBox [19], BING [42], Geodesic Object Proposals [43]
5	Generalized traffic sign detection model for developing a sign inventory Y Tsai, P Kim, Z Wang - Journal of Computing in Civil Engineering, 2009	polygon approximation algorithm detects an octagonal area that should be a stop sign	LADOT D roadways video log image sets. (37,640 video log images)	images with sign in them 83.67% images with no sign in them = 80.19%	proprietary algorithms use specific color filters and the features of specific shapes to distinguish a specific type of traffic sign. but they can detect only stop signs	a generalized traffic sign detection model that can cope with the essential features of traffic signs;

6	Image segmentation and shape analysis for road sign detection - JF Khan, SMA Bhuiyan... - IEEE Transactions on ..., 2011 - ieeexplore.ieee.org	distortion-invariant fringe-adjusted JTC (FJT) C) technique for recognition	121 different road-sign images	88.88%	object detection has been image feature clustering. Bahlmann et al. [26] detecte d signs using a set of color-sensitive Haar wavele t features obtained from AdaBo ost training	automatic road-sign recognitio n method based on image segmentation and joint transform correlation (JTC) with the integration of shape analysis.
---	--	---	--------------------------------	--------	---	---

					an dtemporal information propagation.	
7	Efficient algorithm for automatic road sign recognition and its hardware implementation C Souani, H Faiedh, K Besbes - Journal of real-time image processing, 2014 - Springer	the Multi-Scale Retinex (MSR) algorithm	The images were taken at different light conditions	The recognition rate achieved by the system was around 82 %	First, the image is color segmented. Second, the shape is extracted, and finally, the traffic sign is recognized by processing the local region.	algorithm is a Multi-scale version of the known algorithm as Single-Scale Retinex (SSR): just makes images obtained by averaging SSR at different scales. In the recognition stage, some extracted features, such as symbol, J Real-Time Image Processors and shapes are treated by a Support Vector Machine (SVM)
8	Traffic sign recognition algorithm based on shape signature and dual-tree complex wavelet transform Z Cai, M Gu - Journal of Central South University, 2013 - Springer	shape signature and dual-tree complex wavelet transform	60 000 traffic sign images from several traffic video sequences 21	95.39% at the peak	1) Edge detection in gray image. 2) Clustering and intelligent feature analysis. 3) Image segmentation by threshold in specific color space, then	Image segmentation based on transforming value of RGB color space and shape classifier based on signature feature can be effective

					analysis with geometrical edge	in detecting multipletraffic signs in complex urba n scenes.
--	--	--	--	--	--------------------------------------	--

9	Fast traffic sign recognition with a rotation invariant binary pattern based feature S Yin, P Ouyang, L Liu, Y Guo, S Wei - Sensors, 2015 - mdpi.com	the first optimization called RIBP (Rotation invariant binary pattern the second optimization called ANN (Artificial Neural Network) based feature dimension reduction and classification	GTSRB and STS	1)HOG +ANN=95.41 % 2)SIFT +ANN=97.10 %	traffic sign detection and recognition can be divided into three categories. First, pre-processing methods are researched to locate and recognize the traffic signs. Second, pre-processing methods combining with classification are adopted to achieve robust traffic signs recognition. Third, specific design features combining with the classifiers are used to achieve the robust and computing efficient recognition.	two optimizations for robust and fast traffic sign recognition algorithm exhibits the high discriminative performance among the state-of-the-art features, the techniques of artificial neural network based feature dimension reduction and classification are proposed to reduce the recognition time.
---	---	---	---------------	---	---	--

CHAPTER 3

REQUIREMENT SPECIFICATION

4.1 INTRODUCTION

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

4.2 HARDWARE & SOFTWARE SPECIFICATION

HARDWARE SPECIFICATION

Hardware Requirement

Hardware Minimum Requirement

Disk Space 32 GB or more, 10 GB or more for Foundation Edition

Processor 1.4 GHz 64 bit

Memory 512 MB

Display (800 × 600) Capable video adapter and monitor

SOFTWARE REQUIREMENTS

Backend Technologies

- Python
- Numpy
- Sci-learn
- Eclipse IDE

Frontend Technologies

- Web Technologies
- Bootstrap

4.3 SOFTWARE DESCRIPTION

4.3.1 ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

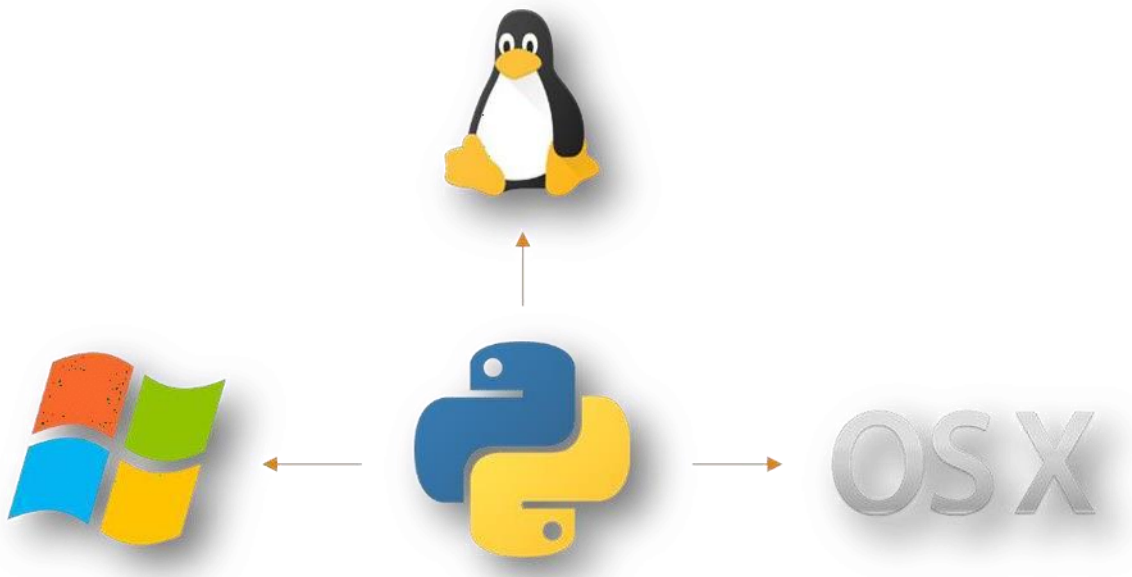


FIG 4.3.1 - 1 PYTHON

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.

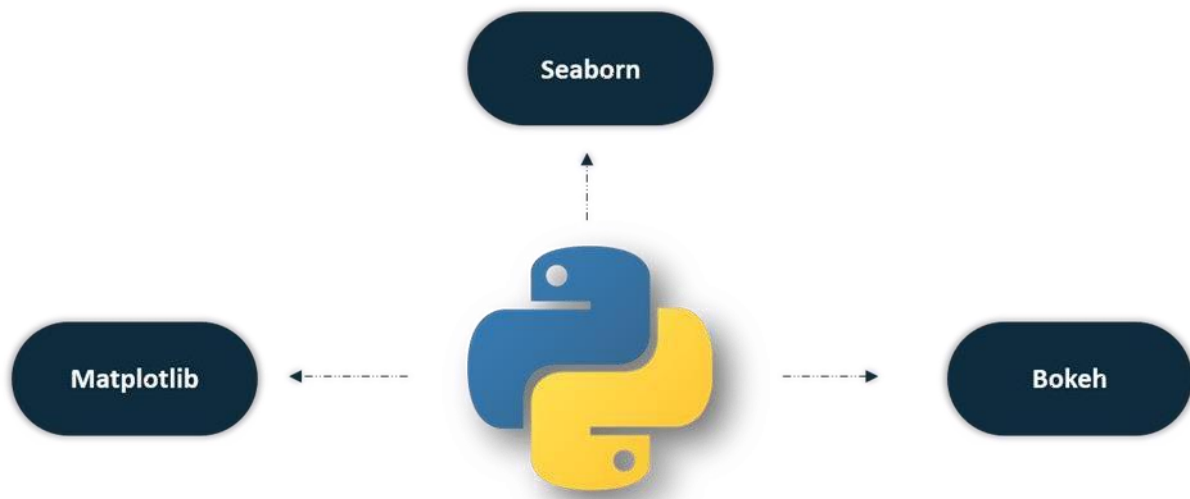


FIG 4.3.1 - 2 PYTHON IMPLEMENTING MACHINE LEARNING

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code.

Following are different kinds of comments that can be included in our Python program:

- Single Line Comments
- Docstring Comments
- Multiline Comments

Single line Python comments are marked with # character. These comments end at the end of the physical line, which means that all characters starting after the # character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low-Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container holds multiple data types and each type doesn't require its own function
- Learning Ease and open-source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more.

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The Data Frame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy,

plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' Data Frames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and Data Frames.

1. **Series:** a pandas Series is a one-dimensional data structure ("a one-dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. **DataFrame:** a panda DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, Data Frames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and

efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't

miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

- NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
- Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.
- NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats

and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

4.3.2 ABOUT IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- We can easily write our code in its text editor window.
- To store the resource files of a project, we have a project editor window.
- It makes debugging of the code easier.
- An IDE's autosuggest feature helps us in writing efficient code.
- We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish

and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse (Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse:

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web-based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be

completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

- Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.
- Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.
- GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.
- High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.
- Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.
- Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.
- Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.
- Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

CHAPTER 5

SYSTEM DESIGN

5.1 INTRODUCTION

we can adopt the latest architectures for object detection such as feature pyramid network and multi-scale training. Neural networks can be trained to recognise patterns containing certain colours. Colour segmentation neural networks can be used to reduce the colour resolution of image . Neural networks are well known for their powerful classification capability. When used in classification, neural networks can be trained to recognise road signs' features within a region of interest. The commonly used models in road sign recognition are Multilayer Perceptron (MLP) network, Radial Basis Network (RBF), etc...,

5.2 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

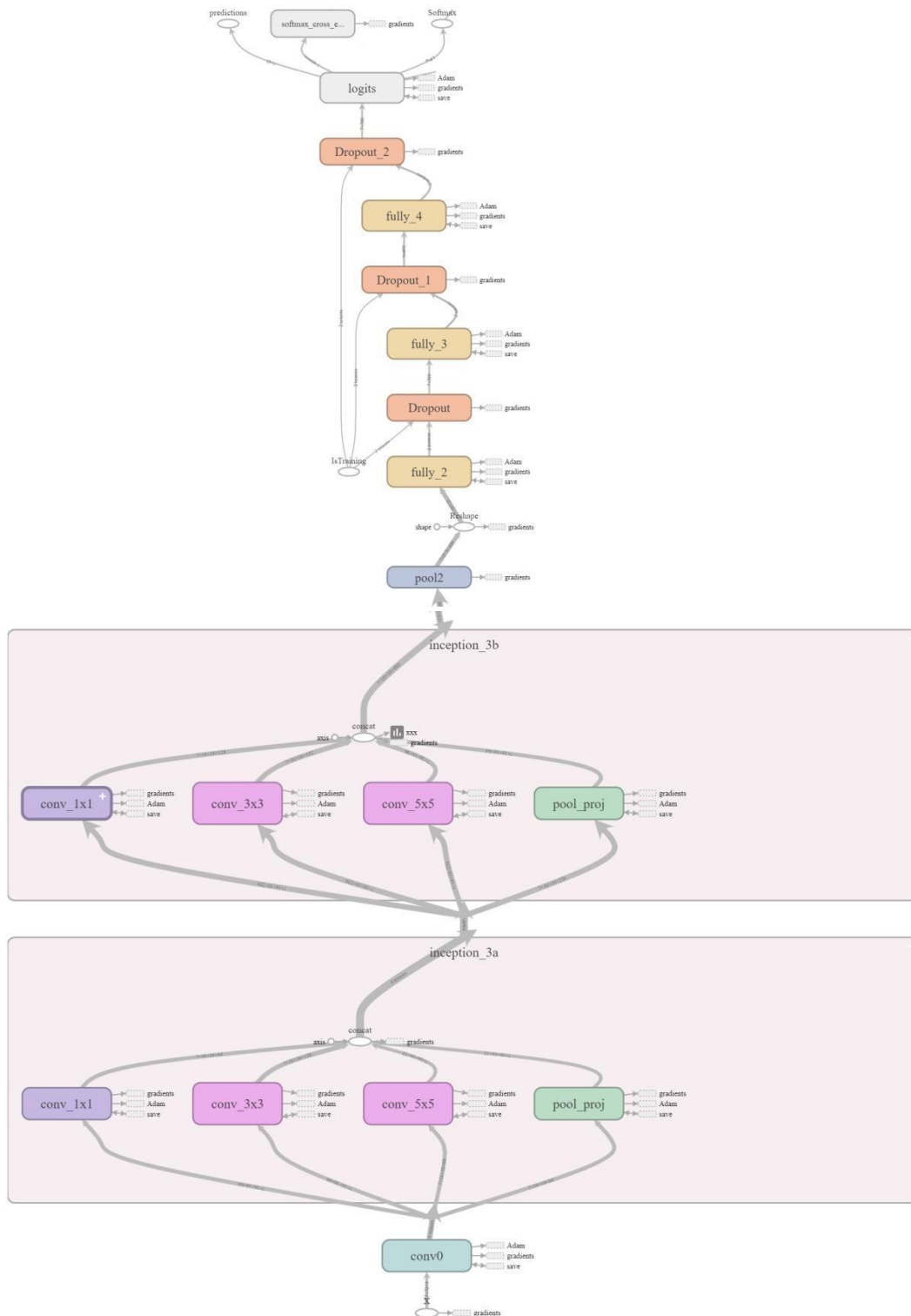
What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the

architecture, discuss changes, and communicate intentions clearly.

FIG 5.1 - 1 ARCHITECTURE DIAGRAM



CHAPTER 6

SYSTEM IMPLEMENTATION

PROBLEM STATEMENT:

The method is to develop a new framework for traffic sign detection and recognition based on the proposals by the guidance of fully conventional neural networks, which largely reduces the search area of traffic sign under the premise of ensuring the detection rate

Existing efficient traffic sign detection method where locations of traffic signs are estimated together with their precise boundaries. To this end, generalized the object bounding box detection problem and formulated an object pose estimation problem, and the problem is effectively modeled using CNN based on the recent advances in object detection networks. The estimated pose of traffic signs is used to transform the boundary of traffic sign templates into the input image plane, providing precise boundaries with high accuracy. Since the base network is a dominant factor in speed-accuracy trade off, developing the base network specialized for traffic sign detection.

we can adopt the latest architectures for object detection such as feature pyramid network and multi-scale training. Neural networks can be trained to recognise patterns containing certain colours. Colour segmentation neural networks can be used to reduce the colour resolution of image. Neural networks are well known for their powerful classification capability. When used in classification, neural networks can be trained to recognise road signs' features within a region of interest. The commonly used models in road sign recognition are Multilayer Perceptron (MLP) network, Radial Basis Network (RBF), etc...,

SOLUTION:

We'll use Convolutional Neural Networks to classify the images in this dataset. The reason behind choosing ConvNets is that they are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They automatically learn hierarchies of invariant features at every level from data. We will implement two of the most famous ConvNets. Our goal is to reach an accuracy of +95% on the validation set. We specify the learning rate of 0.001, which tells the network how quickly to update the weights.

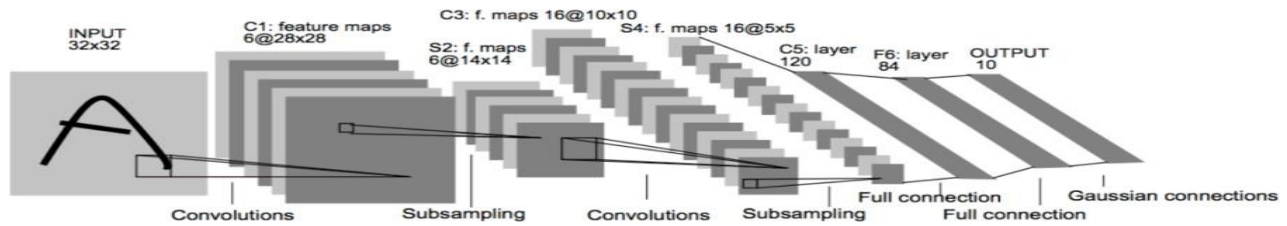
We minimize the loss function using the Adaptive Moment Estimation (Adam) Algorithm. Adam is an optimization algorithm introduced by D. Kingma and J. Lei Ba in a 2015 paper named Adam: A Method for Stochastic Optimization. Adam algorithm computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop algorithms, Adam also keeps an exponentially decaying average of past gradients m_t , similar to momentum algorithm, which in turn produce better results.

we will run `minimize()` function on the optimizer which use backpropagation to update the network and minimize our training loss.

1. LeNet-5

LeNet-5 is a convolutional network designed for handwritten and machine-printed character recognition. It was introduced by the famous Yann LeCun in his paper Gradient-Based Learning

Applied to Document Recognition in 1998. Although this ConvNet is intended to classify hand-written digits, we're confident it have a very high accuracy when dealing with traffic signs, given that both hand-written digits and traffic signs are given to the computer in the form of pixel images.



This ConvNet follows these steps:

Input => Convolution => ReLU => Pooling => Convolution => ReLU => Pooling => FullyConnected
=>ReLU => FullyConnected

Layer 1 (Convolutional): The output shape should be 28x28x6.

Activation. Your choice of activation function.

Pooling. The output shape should be 14x14x6.

Layer 2 (Convolutional): The output shape should be 10x10x16.

Activation. Your choice of activation function.

Pooling. The output shape should be 5x5x16.

Flattening: Flatten the output shape of the final pooling layer such that it's 1D instead of 3D.

Layer 3 (Fully Connected): This should have 120 outputs.

Activation. Your choice of activation function.

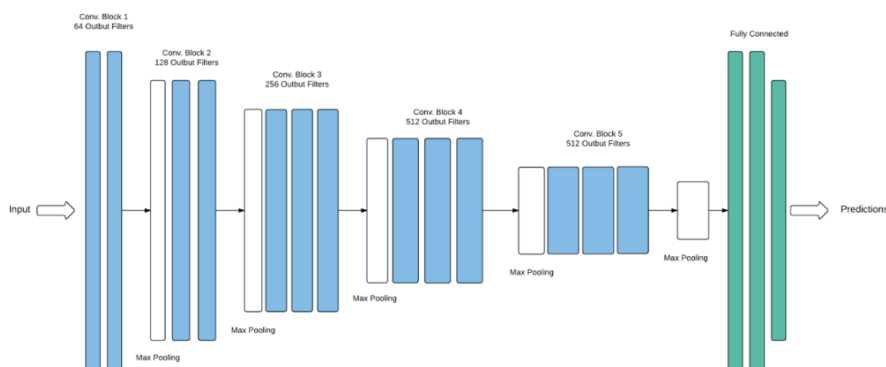
Layer 4 (Fully Connected): This should have 84 outputs.

Activation. Your choice of activation function.

Layer 5 (Fully Connected): This should have 10 outputs.

2. VGGNet

VGGNet was first introduced in 2014 by K. Simonyan and A. Zisserman from the University of Oxford in a paper called Very Deep Convolutional Networks for Large-Scale Image Recognition. They were investigating the convolutional network depth on its accuracy in the large-scale image recognition setting. Their main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3x3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers.



The original VGGNet architecture has 16-19 layers, but I've excluded some of them and

implemented a modified version of only 12 layers to save computational resources.

This ConvNet follows these steps:

Input => Convolution => ReLU => Convolution => ReLU => Pooling => Convolution => ReLU => Convolution => ReLU => Pooling => Convolution => ReLU => Pooling => FullyConnected => ReLU => FullyConnected => ReLU => FullyConnected

Layer 1 (Convolutional): The output shape should be 32x32x32.

Activation. Your choice of activation function.

Layer 2 (Convolutional): The output shape should be 32x32x32.

Activation. Your choice of activation function.

Layer 3 (Pooling) The output shape should be 16x16x32.

Layer 4 (Convolutional): The output shape should be 16x16x64.

Activation. Your choice of activation function.

Layer 5 (Convolutional): The output shape should be 16x16x64.

Activation. Your choice of activation function.

Layer 6 (Pooling) The output shape should be 8x8x64.

Layer 7 (Convolutional): The output shape should be 8x8x128.

Activation. Your choice of activation function.

Layer 8 (Convolutional): The output shape should be 8x8x128.

Activation. Your choice of activation function.

Layer 9 (Pooling) The output shape should be 4x4x128.

Flattening: Flatten the output shape of the final pooling layer such that it's 1D instead of 3D.

Layer 10 (Fully Connected): This should have 128 outputs.

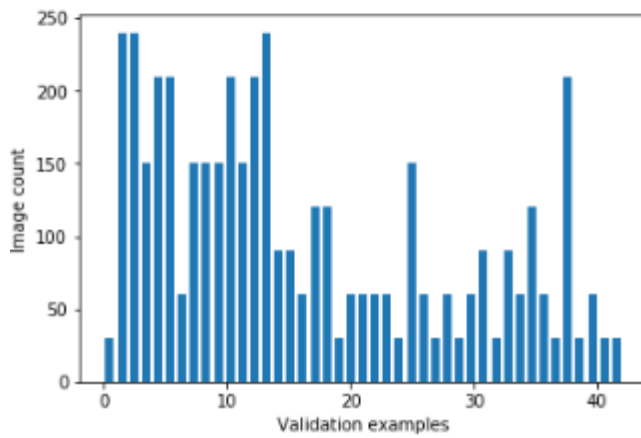
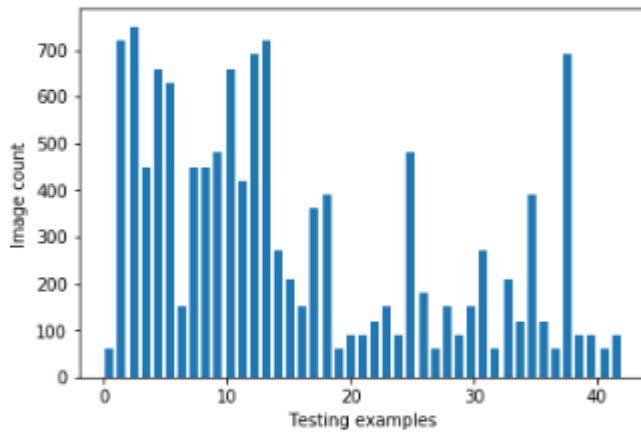
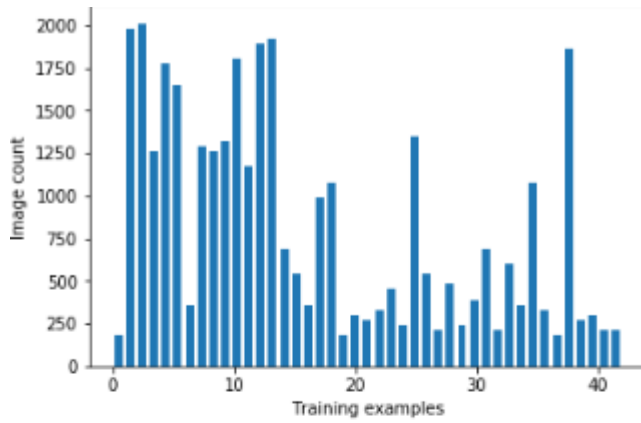
Activation. Your choice of activation function.

Layer 11 (Fully Connected): This should have 128 outputs.

Activation. Your choice of activation function.

Layer 12 (Fully Connected): This should have 43 outputs

RESULTS AND DISCUSSION:

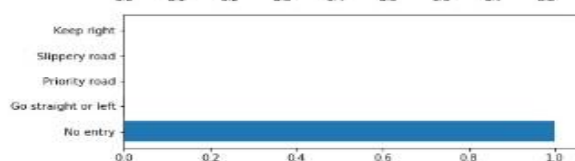
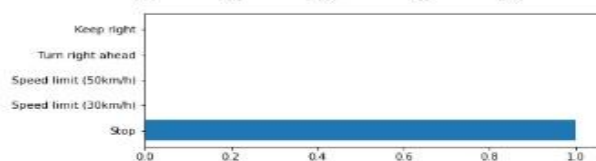
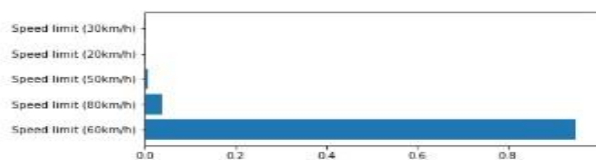
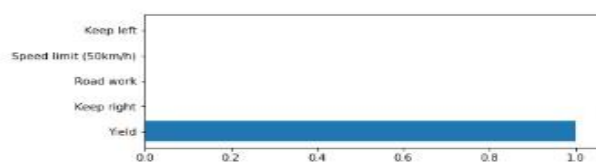
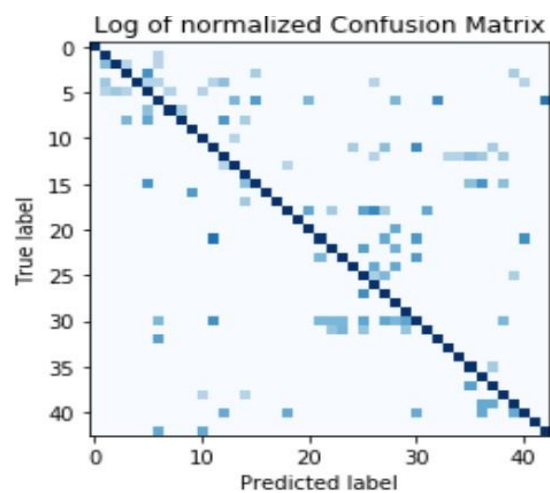


Result with LeNet-5:

```
EPOCH 1 : Validation Accuracy = 81.451%
EPOCH 2 : Validation Accuracy = 87.755%
EPOCH 3 : Validation Accuracy = 90.113%
EPOCH 4 : Validation Accuracy = 91.519%
EPOCH 5 : Validation Accuracy = 90.658%
EPOCH 6 : Validation Accuracy = 92.608%
EPOCH 7 : Validation Accuracy = 92.902%
EPOCH 8 : Validation Accuracy = 92.585%
EPOCH 9 : Validation Accuracy = 92.993%
EPOCH 10 : Validation Accuracy = 92.766%
EPOCH 11 : Validation Accuracy = 93.356%
EPOCH 12 : Validation Accuracy = 93.469%
EPOCH 13 : Validation Accuracy = 93.832%
EPOCH 14 : Validation Accuracy = 94.603%
EPOCH 15 : Validation Accuracy = 93.333%
EPOCH 16 : Validation Accuracy = 93.787%
EPOCH 17 : Validation Accuracy = 94.263%
EPOCH 18 : Validation Accuracy = 92.857%
EPOCH 19 : Validation Accuracy = 93.832%
EPOCH 20 : Validation Accuracy = 93.605%
EPOCH 21 : Validation Accuracy = 93.447%
EPOCH 22 : Validation Accuracy = 94.286%
EPOCH 23 : Validation Accuracy = 94.671%
EPOCH 24 : Validation Accuracy = 94.172%
EPOCH 25 : Validation Accuracy = 94.399%
EPOCH 26 : Validation Accuracy = 95.057%
EPOCH 27 : Validation Accuracy = 95.329%
EPOCH 28 : Validation Accuracy = 94.218%
EPOCH 29 : Validation Accuracy = 94.286%
EPOCH 30 : Validation Accuracy = 94.853%
```

Result with VGGNet:

```
EPOCH 1 : Validation Accuracy = 31.655%
EPOCH 2 : Validation Accuracy = 59.592%
EPOCH 3 : Validation Accuracy = 78.639%
EPOCH 4 : Validation Accuracy = 88.617%
EPOCH 5 : Validation Accuracy = 92.812%
EPOCH 6 : Validation Accuracy = 95.601%
EPOCH 7 : Validation Accuracy = 96.667%
EPOCH 8 : Validation Accuracy = 97.528%
EPOCH 9 : Validation Accuracy = 98.390%
EPOCH 10 : Validation Accuracy = 98.322%
EPOCH 11 : Validation Accuracy = 98.776%
EPOCH 12 : Validation Accuracy = 98.730%
EPOCH 13 : Validation Accuracy = 98.617%
EPOCH 14 : Validation Accuracy = 98.571%
EPOCH 15 : Validation Accuracy = 99.025%
EPOCH 16 : Validation Accuracy = 99.116%
EPOCH 17 : Validation Accuracy = 98.776%
EPOCH 18 : Validation Accuracy = 98.707%
EPOCH 19 : Validation Accuracy = 98.526%
EPOCH 20 : Validation Accuracy = 98.685%
EPOCH 21 : Validation Accuracy = 99.297%
EPOCH 22 : Validation Accuracy = 99.320%
EPOCH 23 : Validation Accuracy = 99.297%
EPOCH 24 : Validation Accuracy = 99.161%
EPOCH 25 : Validation Accuracy = 98.798%
EPOCH 26 : Validation Accuracy = 98.707%
EPOCH 27 : Validation Accuracy = 99.048%
EPOCH 28 : Validation Accuracy = 99.116%
EPOCH 29 : Validation Accuracy = 98.458%
EPOCH 30 : Validation Accuracy = 99.161%
```



Step 0: Load The Data

Step 1: Dataset Summary & Exploration

The pickled data is a dictionary with 4 key/value pairs:

- 'features' is a 4D array containing raw pixel data of the traffic sign images, (num examples, width, height, channels).
- 'labels' is a 1D array containing the label/class id of the traffic sign. The file signnames.csv contains id -> name mappings for each id.
- 'sizes' is a list containing tuples, (width, height) representing the original width and height of the image.
- 'coords' is a list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image.

THESE COORDINATES ASSUME THE ORIGINAL IMAGE.
THE PICKLED DATA CONTAINS RESIZED VERSIONS
(32 by 32) OF THESE IMAGES

Include an exploratory visualization of the dataset

Visualize the German Traffic Signs Dataset using the pickled file(s). This is open ended, suggestions include: plotting traffic sign images, plotting the count of each sign, etc.

The [Matplotlib examples](#) and [gallery](#) pages are a great resource for doing visualizations in Python.

Distribution

Now we are going to explore the distribution and take a look at the distribution of classes in the training, validation and test set.

From the histograms below, we can clearly see that the distribution between train, validation and test is nearly the same, but the problem is that there is a huge variability of the distribution between class instances within the dataset, and we can further investigate whether it can cause some problems during our training, and maybe we can develop augmentation techniques to equalize them

Step 2: Design and Test a Model Architecture

Design and implement a deep learning model that learns to recognize trafficsigns. Train and test your model on the [German Traffic Sign Dataset](#).

The LeNet-5 implementation shown in the [classroom](#) at the end of the CNN lesson is a solid starting point. You'll have to change the number of classes and possibly the preprocessing, but aside from that it's plug and play!

With the LeNet-5 solution from the lecture, you should expect a validation set accuracy of about 0.89. To meet specifications, the validation set accuracy will need to be at least 0.93. It is possible to get an even higher accuracy, but 0.93 is the minimum for a successful project submission.

There are various aspects to consider when thinking about this problem:

- Neural network architecture (is the network over or underfitting?)
- Play around preprocessing techniques (normalization, rgb to grayscale, etc)
- Number of examples per label (some have more than others).
- Generate fake data.

Step 2.1: Pre-process the Data Set (normalization, grayscale, etc.)

- Minimally, the image data should be normalized so that the data has mean zero and equal variance. For image data, $(\text{pixel} - 128) / 128$ is a quick way to approximately normalize the data and can be used in this project.
- Other pre-processing steps are optional. You can try different techniques to see if it improves performance.
- Use the code cell (or multiple code cells, if necessary) to implement the first step of your project.

Step 2.2: Data augmentation

The first thing I tried is to augment the data replicating the class labels which are rare in the dataset, so it can reduce **high variance of our model** (Overfitting)

Step 2.3 : Train, Validate and Test the Model

A validation set can be used to assess how well the model is performing. A low accuracy on the training and validation sets imply underfitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting.

Step 3: Test a Model on New Images

To give yourself more insight into how your model is working, download at least five pictures of German traffic signs from the web and use your model to predict the traffic sign type.

You may find `signnames.csv` useful as it contains mappings from the class id

(integer) to the actual sign name.

Output Top 5 Softmax Probabilities For Each Image Found on the Web

For each of the new images, print out the model's softmax probabilities to show the **certainty** of the model's predictions (limit the output to the top 5 probabilities for each image). `tf.nn.top_k` could prove helpful here.

The example below demonstrates how `tf.nn.top_k` can be used to find the top k predictions for each image.

`tf.nn.top_k` will return the values and indices (class ids) of the top k predictions. So if `k=3`, for each sign, it'll return the 3 largest probabilities (out of a possible 43) and the corresponding class ids.

Take this numpy array as an example. The values in the array represent predictions. The array contains softmax probabilities for five candidate images with six possible classes. `tf.nn.top_k` is used to choose the three classes with the highest probability:

Looking just at the first row we get `[ 0.34763842, 0.24879643, 0.12789202]`, you can confirm these are the 3 largest probabilities in `a`. You'll also notice `[3, 0, 5]` are the corresponding indices.

Step 4 : Visualize the Neural Network's State with Test Images

While neural networks can be a great learning device they are often referred to as a black box. We can understand what the weights of a neural network look like better by plotting their feature maps. After successfully training your neural network you can see what its feature maps look like by plotting the output of the network's weight layers in response to a test stimuli image. From these plotted feature maps, it's possible to see what characteristics of an image the network finds interesting. For a sign, maybe the inner network feature maps react with high activation to the sign's boundary outline or to the contrast in the sign's painted symbol.

Provided for you below is the function code that allows you to get the visualization output of any tensorflow weight layer you want. The inputs to the function should be a stimuli image, one used during training or a new one you provided, and then the tensorflow variable name that represents the layer's state during the training process, for instance if you wanted to see what the [LeNet lab's](#) feature maps looked like for its second convolutional layer you could enter `conv2` as the `tf_activation` variable.

(5, 6) array

```
a = np.array([[ 0.24879643, 0.07032244, 0.12641572, 0.34763842, 0.07893497, 0.12789202],
              [ 0.28086119,  0.27569815,  0.08594638,  0.0178669 , 0.1806340
              1,
              0.15899337],
```

```
[ 0.26076848, 0.23664738, 0.08020603, 0.07001922, 0.113437
1 ,
0.23892179],

[ 0.11943333, 0.29198961, 0.02605103, 0.26234032, 0.135134
8 ,
0.16505091],

[ 0.09561176, 0.34396535, 0.0643941 , 0.16240774, 0.2420613
7,
0.09155967]]])
```

Running it through `sess.run(tf.nn.top_k(tf.constant(a), k=3))` produces: `TopKV2(values=array([[`
`0.34763842, 0.24879643, 0.12789202],`

```
[ 0.28086119, 0.27569815, 0.18063401],
[ 0.26076848, 0.23892179, 0.23664738]
```

```
[ 0.29198961, 0.26234032, 0.16505091],
[ 0.34396535, 0.24206137, 0.16240774]]), indices=array([[3,
0,
5],
[0, 1, 4],
[0, 5, 1],
[1, 3, 5],
[1, 4, 3]],
dtype=int32))
```

CHAPTER 7

SOFTWARE TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Benefits of Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as

documentation testing.

This is one of the most cost-effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

the most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

Test strategy. An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data. The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.

Test plans. A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.

Test scenarios. In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases. If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix. This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own code? What method do they use to test before releasing code for testing? How is dev-testing important in an agile process? The answer to all this is Unit Testing. I want to educate you on the importance of Unit Testing so that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.”
If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



FIG 7 - 1 IMPLEMENTATION

Unit Testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and fixing defects in the early stage of the software development cycle. Unit Testing serves the same purpose.

Unit Testing is an integral part of the agile software development process. When a nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ;-)). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

- Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration
- Fixing an issue in Unit Testing can fix many other issues occurring in later development and testing stages
- Cost of fixing a defect found in Unit Testing is very less than the one found in the system

or acceptance testing

- Code coverage can be measured
- Fewer bugs in the System and Acceptance testing
- Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.
- Expect robust design and development as developers write test cases by understanding the specifications first.
- Easily identify who broke the build
- Saves development time: Code completion may take more time but due to decreased defect count overall development time can be saved.

7.1 GENERAL

Unit Testing frameworks are mostly used to help write unit tests quickly and easily. Most of the programming languages do not support unit testing with the inbuilt compiler. Third-party open source and commercial tools can be used to make unit testing even more fun.

List of popular Unit Testing tools for different programming languages:

- Java framework – JUnit
- PHP framework – PHPUnit
- C++ frameworks – UnitTest++ and Google C++
- .NET framework – NUnit
- Python framework – py.test

- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing
- Non-functional Testing types include:

Performance Testing

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

7.3 UNIT TESTING

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of

an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing –

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.”
If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

7.4 BLACK BOX TESTING

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks app's functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

7.5 INTEGRATION TESTING

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction

between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant influence of software development thinking. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to program. It was not unusual for one programmer to be responsible for a single module, but this would be big enough that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it over to QA for testing.

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

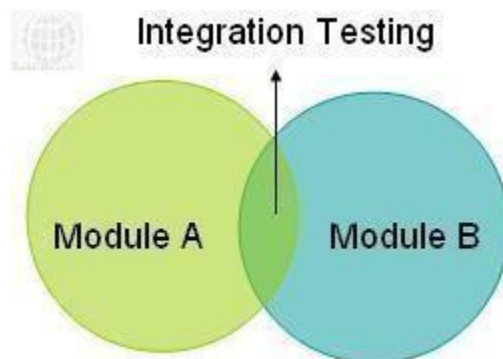


FIG 7 - 3 INTEGRATION TESTING

7.6 SYSTEM TESTING

SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not.

System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.

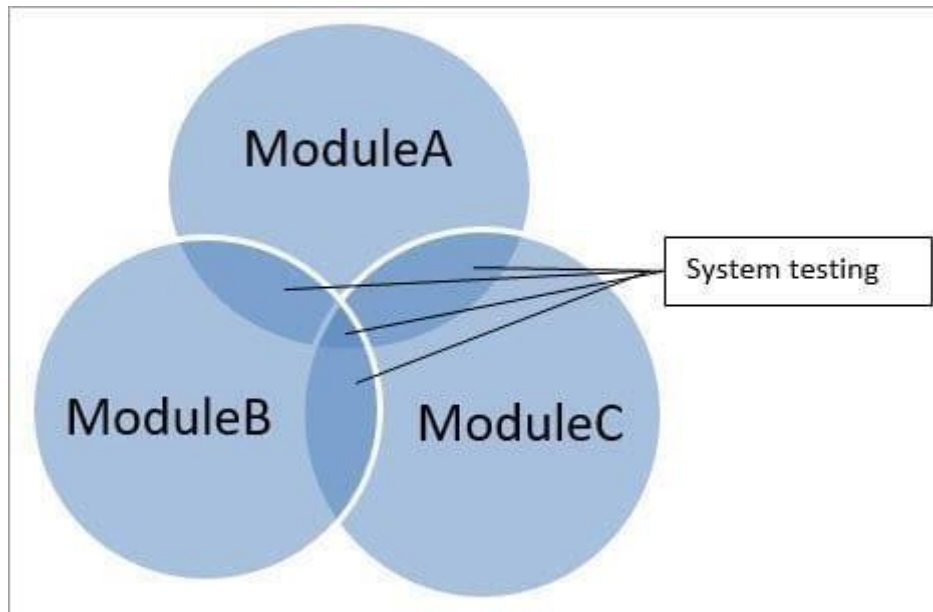


FIG 7 - 4 SYSTEM TESTING

System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next

to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

7.7 SANITY TESTING

Sanity Testing is done when as a QA we do not have sufficient time to run all the test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a checkpoint to determine if testing for the build can proceed or not.

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules- like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also checks if it does not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately - like a critical bug fix.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as “Day 0” check. It is done at the “build level”.

It helps not to waste the testing time to simply testing the whole application when the key features don't work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow.

To conduct smoke testing, we don't write test cases. We just pick the necessary test cases from already written test cases.

Do we really write test cases for all testing types? Here in this article, we have given clear idea on choosing testing types to write test cases.

As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test cases that won't take more than half an hour to execute.

After receiving a Software build with the minor issues fixes in code or functionality, Sanity testing is carry out to check whether the bugs reported in previous build are fixed & there is regression introduced due to these fixes i.e. not breaking any previously working functionality. The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is perform.

Sanity tests helps to avoid wasting time and cost involved in testing if the build is failed. Tester should reject the build upon build failure.

After completion of regression testing the Sanity testing is started to check the defect fixes & changes done in the software application is not breaking the core functionality of the software. Typically, this is done nearing end of SDLC i.e. while releasing the software. You can say that sanity testing is a subset of acceptance testing. We can also say Tester Acceptance Testing for Sanity testing.

7.8 REGRESSION TESTING

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to

check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.

For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an essential aspect of a dynamic, iterative development and deployment schedule.

But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size. The process incorporates “performance” indicators such as:

- Browser, page, and network response times
- Server request processing times
- Acceptable concurrent user volumes
- Processor memory consumption; number and type of errors that might be encountered with app

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



FIG 7 - 5 PERFORMANCE TESTING

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying power

of application when system is performing well under heavy load. Load testing comes under the Non-Functional Testing & it is designed to test the non-functional requirements of a software application.

Load testing is performed to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero

CHAPTER 9

CONCLUSION & FUTURE ENHANCEMENT

9.1 CONCLUSION & FUTURE ENHANCEMENT

The proposed system provides an assessment method in accordance with certain logic, reliable theoretical basis and existing research results in order to help users estimate the applications before installation. Via evaluation and statistical analysis, the assessment method can identify a part of malicious applications, mainly for those that apply for far too much permission or have relatively large differences with other applications which belong to the same category. novice users of Android are not aware of the permissions they have granted at the time of installation. Our proposed system provided the second level of security to the users, without modifying the underlying structure of Android operating system.

REFERENCES

- [1] K. Stenzel, K. Katkalov, M. Borek, and W. Reif, "A model-driven approach to noninterference," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 5, no. 3, pp. 30–43, September 2014.
- [2] S. Rasthofer, S. Arzt, and E. Bodden, "A machine-learning approach for classifying and categorizing android sources and sinks," *Proceedings of the Network and Distributed System Security Symposium*, 2014.
- [3] M. Whitney, H. Richter Lipford, B. Chu, and T. Thomas. "Embedding Secure Coding Instruction into the IDE: Complementing Early and Intermediate CS Courses with ESIDE" In press, *Journal of Educational Computing Research*, 2017
- [4] B. Fisseha Demissie, D. Ghio, M. Ceccato, A. Avancini Identifying Android Inter-app Communication Vulnerabilities Using Static and Dynamic Analysis, *IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2016
- [5] T. Dallas, "13 Must-Know Tips for Keeping Your Phone Secure," [Online]. Available: <https://android.gadgethacks.com/how-to/android-security-13-must-know-tips-for-keeping-your-phone-secure-0162723/>. [Accessed 11 October 2017].
- [6] Khokhlov and L. Reznik, "Data Security Evaluation for Mobile Android Devices," in *IEEE Information Security and Protection of Information Technologies Conference*, St. Petersburg, 2017.
- [7] Kaur and D. Upadhyay, "PeMo: Modifying application's permissions and preventing information stealing on smartphones," *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Sep. 2014.
- [8] Z. Fang, W. Han, and Y. Li, "Permission based Android security: Issues and countermeasures," *Computers & Security*, vol. 43, pp. 205–218, Jun. 2014.
- [9] L. Xing, X. Pan, R. Wang, K. Yuan, and X. Wang, "Upgrading Your Android, Elevating My Malware: Privilege Escalation Through Mobile OS Updating," *IEEE Symp. Secur. Priv.*, 2014.
- [10] L. Whitney, "Almost no one is using Android marshmallow, still," *CNET*, 2016. [Online]. Available: <http://www.cnet.com/news/almost-no-one-is-using-androidmarshmallow- still>.

- [11] J.-K. Park and S.-Y. Choi, "Studying security weaknesses of Android system," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 7–12, Mar. 2015.
- [12] Security and P. Foundation, "Gimme Rat – Android Malware In The Wild," XSec Technologies Pvt Ltd and Cyber Security & Privacy Foundation, India, p.5, mar 2014.
- [13] S. Calzavara, I. Grishchenko, and M. Maffei, "Horndroid: Practical and sound static analysis of android applications by smt solving," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 4762.
- [14] M. I. Gordon, D. Kim, J. H. Perkins, L. Gilham, N. Nguyen, and M. C. Rinard, "Information flow analysis of Android applications in droidsafe," in *Proc. NDSS*, 2015, pp. 116.
- [15] G. Betarte, J. Campo, M. Cristiá, F. Gorostiaga, C. Luna, and C. Sanz, "Towards formal model-based analysis and testing of Android's security mechanisms," in *Proc. 43rd Latin Amer. Comput. Conf. (CLEI)*, Sep. 2017, pp. 110.
- [16] G. Betarte, J. D. Campo, C. Luna, and A. Romano, "Verifying Android's permission model," in *Proc. Int. Colloq. Theor. Aspects Comput.*, 2015, pp. 485504.

CODE

CODE:

```
# Load pickled data
import pickle

# TODO: Fill this in based on where you saved the training and testing data

training_file="train.p"
validation_file="test.p"
testing_file = "test.p"

with open(training_file, mode='rb') as f:
    train = pickle.load(f)
with open(validation_file, mode='rb') as f:
    valid = pickle.load(f)
with open(testing_file, mode='rb') as f:
    test = pickle.load(f)

X_train, y_train = train['features'], train['labels']
X_valid, y_valid = valid['features'], valid['labels']
X_test, y_test = test['features'], test['labels']
import numpy as np
from matplotlib import pyplot

# TODO: Number of training examples
n_train = len(X_train)

# TODO: Number of validation examples
n_validation = len(X_valid)

# TODO: Number of testing examples.
n_test = len(X_test)

# TODO: What's the shape of an traffic sign image?
image_shape = np.array(X_train[0]).shape

# TODO: How many unique classes/labels there are in the dataset.
n_classes = len(set(y_train))
sign_classes, class_indices, class_counts = np.unique(y_train, return_index = True, return_counts = True)
```

```

pyplot.bar( np.arange( 43 ), class_counts, align='center' )
pyplot.xlabel('Class')
pyplot.ylabel('Number of training examples')
pyplot.xlim([-1, 43])
pyplot.show()
print("Number of training examples =", n_train)
print("Number of testing examples =", n_test)
print("Image data shape =", image_shape)
print("Number of classes =", n_classes)
print("Number of validation examples=",n_validation)
import random
import numpy as np
import matplotlib.pyplot as plt

index = random.randint(0, len(X_train))
image = X_train[index].squeeze()

plt.figure(figsize=(1,1))
plt.imshow(image, cmap="gray")
print(y_train[index])
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.utils import shuffle
from skimage import exposure

def preprocess_dataset(X, y = None):
    #print(X.shape)
    #Convert to grayscale, e.g. single Y channel
    X = 0.299 * X[:, :, :, 0] + 0.587 * X[:, :, :, 1] + 0.114 * X[:, :, :, 2]
    #Scale features to be in [0, 1]
    X = (X / 255.).astype(np.float32)
    # Add a single grayscale channel
    X = X.reshape(X.shape + (1,))
    return X, y

X_test, y_test = preprocess_dataset(X_test,y_test)
X,y = X_train,y_train
X_train,y_train = preprocess_dataset(X_train,y_train)
#print(X_train.shape)
X_train, y_train = shuffle(X_train, y_train)
#X_train = (X_train -128)/128
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=0.20)
import tensorflow as tf

EPOCHS = 40
BATCH_SIZE = 128
dropout1 = 0.90 # Dropout, probability to keep units
dropout2 = 0.80
dropout3 = 0.70
dropout4 = 0.50
from tensorflow.contrib.layers import flatten
mu = 0
sigma = 0.1
layer1_weight = tf.Variable(tf.truncated_normal([5, 5, 1, 6],mean = mu,stddev=sigma))
layer1_bias = tf.Variable(tf.zeros(6))
layer2_weight = tf.Variable(tf.truncated_normal([5, 5, 6, 16],mean = mu,stddev=sigma))
layer2_bias = tf.Variable(tf.zeros(16))
flat_weight = tf.Variable(tf.truncated_normal([400, 120],mean = mu,stddev=sigma))
bias_flat = tf.Variable(tf.zeros(120))
flat_weight2 = tf.Variable(tf.truncated_normal([120, 84],mean = mu,stddev=sigma))
bias_flat2 = tf.Variable(tf.zeros(84))
flat_weight3 = tf.Variable(tf.truncated_normal([84, 43],mean = mu,stddev=sigma))
bias_flat3 = tf.Variable(tf.zeros(43))

def LeNet(x,train=True):
    # Arguments used for tf.truncated_normal, randomly defines variables for the weights and biases for each layer
    x = tf.nn.conv2d(x, layer1_weight, strides=[1, 1, 1, 1], padding='VALID')
    x = tf.nn.bias_add(x, layer1_bias)
    x = tf.nn.relu(x)
    x = tf.nn.max_pool(x,ksize=[1, 2, 2, 1],strides=[1, 2, 2, 1],padding='SAME')
    if(train):
        x = tf.nn.dropout(x, dropout1)

```

```

x = tf.nn.conv2d(x, layer2_weight, strides=[1, 1, 1, 1], padding='VALID')
x = tf.nn.bias_add(x, layer2_bias)
x = tf.nn.relu(x)
conv2 = tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
if(train):
    conv2 = tf.nn.dropout(conv2, dropout2)

fc0 = flatten(conv2)
fc1 = tf.add(tf.matmul(fc0, flat_weight), bias_flat)
fc1 = tf.nn.relu(fc1)
if(train):
    fc1 = tf.nn.dropout(fc1, dropout3)

fc1 = tf.add(tf.matmul(fc1, flat_weight2), bias_flat2)
fc1 = tf.nn.relu(fc1)
if(train):
    fc1 = tf.nn.dropout(fc1, dropout4)
fc1 = tf.add(tf.matmul(fc1, flat_weight3), bias_flat3)
logits = tf.nn.relu(fc1)

return logits
x = tf.placeholder(tf.float32, (None, 32, 32, 1))
y = tf.placeholder(tf.int32, (None))
keep_prob = tf.placeholder(tf.float32) #dropout (keep probability)
one_hot_y = tf.one_hot(y, 43)
rate = 0.001

logits = LeNet(x)
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(labels=one_hot_y, logits=logits)
loss_operation = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate = rate)

training_operation = optimizer.minimize(loss_operation)
logits_2 = LeNet(x)
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(one_hot_y, 1))
accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

saver = tf.train.Saver()
session = tf.Session()
session.run(tf.global_variables_initializer())
def evaluate(X_data, y_data):
    num_examples = len(X_data)
    total_accuracy = 0
    sess = tf.get_default_session()
    for offset in range(0, num_examples, BATCH_SIZE):
        batch_x, batch_y = X_data[offset:offset+BATCH_SIZE], y_data[offset:offset+BATCH_SIZE]
        accuracy = sess.run(accuracy_operation, feed_dict={x: batch_x, y: batch_y})
        total_accuracy += (accuracy * len(batch_x))
    return total_accuracy / num_examples
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    num_examples = len(X_train)

    print("Training...")
    print()
    for i in range(EPOCHS):
        X_train, y_train = shuffle(X_train, y_train)
        for offset in range(0, num_examples, BATCH_SIZE):
            end = offset + BATCH_SIZE
            batch_x, batch_y = X_train[offset:end], y_train[offset:end]
            sess.run(training_operation, feed_dict={x: batch_x, y: batch_y})

        validation_accuracy = evaluate(X_validation, y_validation)
        print("EPOCH {} ...".format(i+1))
        print("Validation Accuracy = {:.3f}".format(validation_accuracy))
        print()

        saver.save(sess, './lenet')
        print("Model saved")
    with tf.Session() as sess:
        saver.restore(sess, './lenet')
        test_accuracy = evaluate(X_test, y_test)
        print("Test Accuracy = {:.3f}".format(test_accuracy))
from skimage import io
X_custom = np.empty([0, 32, 32, 3], dtype = np.int32)

```

```

for i in range(38):
    raw_path = "/Users/sai_ganesh_reddy/Desktop/traffic_sign_classifier-master/custom/"
    image = io.imread(raw_path + "example_{0:0>5}".format(i + 1) + '.png')
    X_custom = np.append(X_custom, [image[:, :, :3]], axis = 0)
y_custom = np.array([21,39,17,17,17,39,39,40,40,34,28,39,0,17,38,13,40,13,38,38,11,0,28,0,99,99,99,32,40,28,40,40,28,24,0,0,0,0])

X_custom, _ = preprocess_dataset(X_custom)
new_pred = tf.nn.softmax(LeNet(X_custom,False))

with tf.Session() as sess:
    saver.restore(sess, './lenet')
    test_preds = sess.run(new_pred, feed_dict={x:X_custom,y:y_custom})
    print(test_preds)
    top_k_predictions = tf.nn.top_k(test_preds, 5)
    print(top_k_predictions)
import pandas as pd

sign_names = pd.read_csv('signnames.csv')

def get_sign_name(class_index):
    row = sign_names[sign_names.ClassId == int(class_index)]
    return row.SignName.values[0]

def get_sign_names(indices):
    return [get_sign_name(index) for index in indices]
output = test_preds
with tf.Session() as sess:
    saver.restore(sess, tf.train.latest_checkpoint('.'))
    sess.run(tf.global_variables_initializer())
    session=sess
    for i, image in enumerate(X_custom):
        # Plot the original image
        plt.subplot(121);
        plt.imshow(image.reshape(image_shape[0], image_shape[1]), cmap='gray');
        plt.title('Image');

        # Get the top k probabilities and the classes to which they correspond
        top_k = tf.nn.top_k(output[i], k=5)
        indices = top_k.indices.eval(session=session)
        values = top_k.values.eval(session=session)

        # Sort in ascending order
        idx = values.argsort()
        values = values[idx]
        indices = indices[idx]

        # Show a bar chart of the probabilities
        y_pos = range(0, 200, 40)

        plt.subplot(122)
        plt.bar(y_pos, values, align='center', alpha=0.5, color='red', width=20)

        plt.xticks(y_pos, get_sign_names(indices))
        locs, labels = plt.xticks()
        plt.setp(labels, rotation=90)

        plt.ylabel('Probability')
        plt.title('Probability of top 5 classes')
        plt.show()

```

OUTPUT FOR 40 EPOCHS:

```
Number of training examples = 39209
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
Number of validation examples= 12630
5
WARNING:tensorflow:From C:\Users\sai ganesh reddy\Desktop
ated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients
into the labels input on backprop by default.

See @tf.nn.softmax_cross_entropy_with_logits_v2}.

Training...

EPOCH 1 ...
Validation Accuracy = 0.365

EPOCH 2 ...
Validation Accuracy = 0.549

EPOCH 3 ...
Validation Accuracy = 0.654

EPOCH 4 ...
Validation Accuracy = 0.703

EPOCH 5 ...
Validation Accuracy = 0.749

EPOCH 6 ...
Validation Accuracy = 0.759
```

```

EPOCH 34 ...
Validation Accuracy = 0.866

EPOCH 35 ...
Validation Accuracy = 0.868

EPOCH 36 ...
Validation Accuracy = 0.865

EPOCH 37 ...
Validation Accuracy = 0.863

EPOCH 38 ...
Validation Accuracy = 0.860

EPOCH 39 ...
Validation Accuracy = 0.863

EPOCH 40 ...
Validation Accuracy = 0.873

Model saved
Test Accuracy = 0.820
[[2.3255814e-02 2.3255814e-02 2.3255814e-02 ... 2.3255814e-02
  2.3255814e-02 2.3255814e-02]
 [1.3796603e-17 6.1904339e-16 1.3796603e-17 ... 1.3574685e-16
  1.3796603e-17 1.3796603e-17]
 [5.1604860e-07 5.1604860e-07 5.1604860e-07 ... 5.1604860e-07
  5.1604860e-07 5.1604860e-07]
 ...
 [2.3255814e-02 2.3255814e-02 2.3255814e-02 ... 2.3255814e-02
  2.3255814e-02 2.3255814e-02]
 [1.7598109e-02 2.6087973e-01 1.7598109e-02 ... 1.7598109e-02
  1.7598109e-02 1.7598109e-02]
 [2.3255814e-02 2.3255814e-02 2.3255814e-02 ... 2.3255814e-02
  2.3255814e-02 2.3255814e-02]]
TopKV2(values=<tf.Tensor 'TopKV2:0' shape=(38, 5) dtype=float32>, indices=<tf.Tensor 'TopKV2:1' shape=(38, 5) dtype=int32>

```

