

GESTURE RECOGNITION FOR DEAF AND DUMB

AN INDUSTRY ORIENTED-MINI PROJECT REPORT

**Submitted to
Jawaharlal Nehru Technological University Hyderabad**

*In partial fulfillment of the requirements
for the award of the degree of*

**BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

NAKKA AKSHAY (20E11A0527)

**Under the Supervision of
Ms. KIRANMAI
Assistant Professor**



Department of Computer Science and Engineering
**BHARAT INSTITUTE OF ENGINEERING
AND TECHNOLOGY**

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE)
Approved by AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpattanam-501 510, Hyderabad, Telangana.

DECEMBER 2023-2024



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpattam -501 510, Hyderabad, Telangana

Certificate

***This is to certify that the IO-Mini project work entitled Gesture Recognition
for The Deaf and Dumb “” is the bona fide work done***

By

NAKKA AKSHAY (20E11A0527)

***in the Department of Computer Science and Engineering, BHARAT INSTITUTE
OF ENGINEERING AND TECHNOLOGY, Ibrahimpattam is submitted to
Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of
the requirements for the award of B.Tech degree in Computer Science and
Engineering during 2020- 2024.***

Guide:

Ms. Saiba Jan

Professor
Dept. of IT/DS/CS,
Bharat Institute of Engineering and Technology,
Ibrahimpattam – 501510, Hyderabad.

Head of the Department:

Dr. Mahesh Lokhande

Associate Professor
Dept of CSE,
Bharat Institute of Engineering and Technology,
Ibrahimpattam – 501 510, Hyderabad.

Viva-Voce held on.....

Internal Examiner

External Examiner

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpattanam -501 510, Hyderabad, Telangana

Vision of the Institution

To achieve the autonomous & university status and spread universal education by inculcating discipline, character and knowledge into the young minds and mould them into enlightened citizens.

Mission of the Institution

Our mission is to impart education, in a conducive ambience, as comprehensive as possible, with the support of all the modern technologies and make the students acquire the ability and passion to work wisely, creatively and effectively for the betterment of our society.

Vision of CSE Department

Serving the high quality educational needs of local and rural students within the core areas of Computer Science and Engineering and Information Technology through a rigorous curriculum of theory, research and collaboration with other disciplines that is distinguished by its impact on academia, industry and society.

Mission of CSE Department

The Mission of the department of Computer Science and Engineering is

- To work closely with industry and research organizations to provide high quality Computer education in both the theoretical and applications of Computer Science and Engineering.
- The department encourages original thinking, fosters research and development, evolve innovative applications of technology.

A Mini Project Report



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpattanam -501 510, Hyderabad, Telangana

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Computer Science and Engineering program provides students with an in depth education in the conceptual foundations of computer science and in complex hardware and software systems. It allows them to explore the connections between computer science and a variety of other disciplines in engineering and outside. Combined with a strong education in mathematics, science, and the liberal arts it prepares students to be leaders in computer science practice, applications to other disciplines and research.

Program Educational Objective 1: (PEO1)

The graduates of Computer Science and Engineering will have successful career in technology or managerial functions.

Program Educational Objective 2: (PEO2)

The graduates of the program will have solid technical and professional foundation to continue higher studies.

Program Educational Objective 3: (PEO3)

The graduates of the program will have skills to develop products, offer services and create new knowledge.

Program Educational Objective 4: (PEO4)

The graduates of the program will have fundamental awareness of Industry processes, tools and technologies.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
 AICTE, Affiliated to JNTUH Hyderabad
 Ibrahimpatnam -501 510, Hyderabad, Telangana

PROGRAM OUTCOMES (POs)

PO1:	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2:	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3:	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4:	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5:	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6:	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7:	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8:	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9:	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10:	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation,make effective presentations,and give and receive clear instructions.
PO11:	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work,as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12:	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
AICTE, Affiliated to JNTUH Hyderabad
Ibrahimpattanam -501 510, Hyderabad, Telangana

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1:	Foundation of mathematical concepts: To use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm.
PSO2:	Foundation of Computer System: The ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems.
PSO3:	Foundations of Software development: The ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process. Familiarity and practical proficiency with a broad area of programming concepts and provide new ideas and innovations towards research.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC, Accredited by NBA (UG Programmes: CSE & ECE) Approved by
 AICTE, Affiliated to JNTUH Hyderabad
 Ibrahimpatnam -501 510, Hyderabad, Telangana

QUALITY OF THE PROJECT

I. Consideration to Factors

Factors (Environment, Safety, Ethics, Cost)	Type of Project (Application, Product, Research, Review, etc.)	Standards
This project has impact on the WHO based on public information.	This is a research based project which analyses the efficient Results using the algorithms of machine learning	

II. POs and PSOs addressed through the project with justification

S. No.	POs and PSOs Addressed	Justification
1.	PO1	Engineering knowledge: Machine Learning algorithms were used for the comparison of results.
2.	PO2	Problem analysis: The main drawbacks were observed, and led to implementation of three machine-learning algorithms.
3.	PO3	Design/Development of solutions: We had designed the solution that gives the accurate values.
4.	PO5	Modern Tool Usage: We had implemented all the algorithms using modern engineering and IT tools (i.e., Python language and Anaconda navigator).
5.	PSO1	Foundation of mathematical concepts: Calculating the accuracy and precision are done based on these three algorithms.

6.	PSO3	Foundation of Software Development: This project has the proper usage of Software Development Life Cycle.
----	------	--

DECLARATION

We hereby declare that this Project Work titled "Self diagnosable human disease prediction using machine learning" is a genuine project work carried out by us, in B.Tech (Computer Science and Engineering) degree course of Jawaharlal Nehru Technology University Hyderabad, Hyderabad and has not been submitted to any other course or university for the award of my degree by us.

Candidate Name(s) Roll Number Signature

NAKKA AKSHAY 20E11A0527

Date:

ABSTRACT

The Sign Language detection by technology is an overlooked concept despite there being a large social group that could benefit from it. Understanding sign language is essential in helping users of sign language communicate with the rest of society. Image classification and machine learning can be used to help computers recognize sign language, which could then be interpreted by other people. CNN's have been employed in this project to recognize sign language gestures. The image dataset used consists of static sign language gestures captured on an RGB camera. The images are preprocessed which then serve as the cleaned input. The results are obtained by retraining and testing this sign language gestures dataset on a CNN model which consists of multiple convolution filter inputs that are processed on the same input

KEYWORDS: *CNN, RGB Camera.*

PAGE INDEX

Chapter-1 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 PROBLEM STATEMENT... ..	1
1.3 PROBLEM OBJECTIVE... ..	2
Chapter-2 LITERATURE SURVEY	3
2.1 EXISTING SYSTEM	6
2.2 LIMITATIONS OF EXISITNG SYSTEM	6
Chapter-3 SOFTWARE REQUIREMENT SPECIFICATION	7
3.1 SOFTWARE REQUIREMENTS	7
3.2 HARDWARE REQUIREMENTS	7
3.3 FUNCTIONAL REQUIREMENTS.....	8
3.4 NON-FUNCTIONAL REQUIREMENTS	8
Chapter-4 DESIGN.....	10
4.1 SYSTEM DESIGN.....	10
4.2 UML DESIGN.....	11
4.3 SYSTEM ARCHITECTURE... ..	18
4.4 DATA FLOW DIAGRAM	19

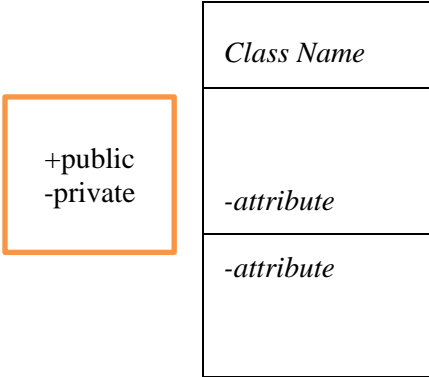
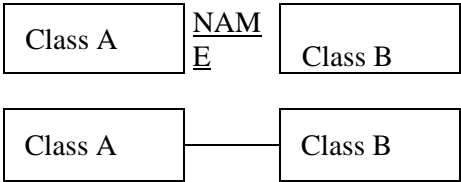
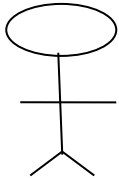
Chapter-5 IMPLEMENTATION.....	20
5.1 LIBRARIES.....	20
5.2 IMPLEMENTATION DESGIN.....	25
5.3 VARIOUS DATASETS.....	26
5.4 DATASET PREPARATION AND PREPROCESSING	27
5.5 EXECUTABLE CODE	29
Chapter-6 TESTING.	37
6.1 TESTING.....	39
6.2 TYPES OF TESTING	37
6.3 UNIT TESTING.....	38
6.4 TEST CASES	39
Chapter-7 RESULTS... ..	42
7.1 USER INTERFACE 1.....	42
7.2 USER INTERFACE 2.....	42
7.3 USER INTERFACE 3	43
7.4 USER INTERFACE 4... ..	43
Chapter-8 CONCLUSION... ..	44
Chapter-9 FUTURE SCOPE.....	45
Chapter-10 REFERENCES... ..	46

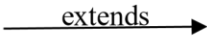

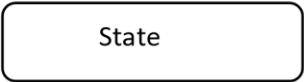
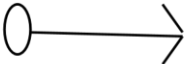
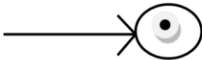
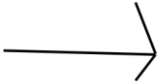
SNO.	LIST OF FIGURES	PAGENO.
4.2.1	UML Hierarchy Diagrams	13
4.2.1.1	Class Diagram	14
4.2.2.1	Use Case Diagram	15
4.2.3.1	Activity Diagram	16
4.2.4.1	Sequence Diagram	17
4.3.1	System Architecture	18
4.4.1	Data Flow Diagram	19
5.2.1	Implementation Design	25
5.3.1	Various Datasets	26
6.4.1.1	Load The Model Test Case	39
6.4.2.1	Capture Frames from Video	40
6.4.3.1	Hand Gesture Recognition	41

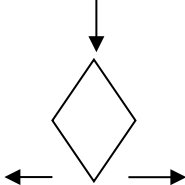
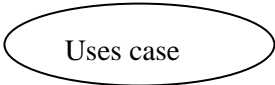
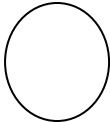


SNO.	LIST OF SCREENSHOTS	PAGE NO.
5.2.1	Implementation Design	25
5.3.1	Various Datasets	26
7.1.1	User Interface 1	42
7.1.2	User Interface 2	42
7.1.3	User Interface 3	43
7.1.4	User Interface 4	43

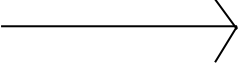
LIST OF ABBREVIATIONS		
S.NO	ABBREVIATION	EXPANSION
1	CNN	CONVOLUTIONAL NEURAL NETWORKS
2	RGB	RED GREEN BLUE
3	HSV	HUE SATURATION VALUE
4	RCNN	REGIONS WITH CONVOLUTIONAL NEURAL NETWORKS
5	RPN	REGION PROPOSAL NETWORK
6	HLD	HIGH LEVEL DESIGN
7	LLD	LOW LEVEL DESIGN
8	UML	UNIFIED MODEL LANGUAGE
9	GUI	GRAPHICAL USER INTERFACE
10	OPENCV	OPEN SOURCE COMPUTER VISION LIBRARY
11	ONEIROS	OPEN-ENDED NEURO-ELECTRONIC ROBOT OPERATING SYSTEM
12	GPU	GRAPHICS PROCESSING UNIT
13	TPU	TENSOR PROCESSING UNIT
14	MLP	MULTI-LAYERED PERCEPTRON

LIST OF SYMBOLS

S.NO	NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Association represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.

5.	<i>Relation</i> (uses)	<i>Uses</i>	Used for additional process communication.
6.	Relation (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.

B.Tech-IO	Mini Project		2023-2024
12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interact ion between the system and external environment.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.

19.	Message	Message 	Represents the message exchanged.
-----	---------	--	-----------------------------------

CHAPTER - 1

INTRODUCTION

1.1 PURPOSE OF THE PROJECT

To assist differently abled people with respect to speaking and hearing, translators were always a must. Since sign language is not known and understood by many, a translator was always necessary. However, it's not very convenient for a translator to be available at all times, and sometimes not convenient in particular settings. To overcome this, the Gesture Recognition for Deaf and Dumb is used to eliminate the need of a translator. The application translates sign language and enables the opposite person to understand what is to be conveyed.

1.2 PROBLEM STATEMENT

There are many applications where hand gestures can be used for interaction with systems like video Games, controlling UAVs, medical equipment, etc. System present and developed for hand gestures Recognition can be used by handicapped people to interact with the systems. Classical interactions tools like keyboard, mouse, touchscreen, etc. May limit the way we use the system. All these systems require Physical contact, to interact with the system. Gestures can interpret the same functionality without physically interacting with the interfacing devices. The problem lies in understanding these gestures, as for different people, the same gesture may look Different for performing the same task. This problem may be overthrown by the use of deep learning Approaches.

1.3 PROJECT OBJECTIVE

Sign Language Translator enables the hearing-impaired user to communicate efficiently in sign language. The model is trained using prerecorded sign language gestures where each sign language gesture is correctly labelled with its corresponding English alphabet. This model also allows the user to construct words or sentences using a sequence of sign language gestures. The user can use these gestures to communicate effectively.

CHAPTER-2

LITERATURE SURVEY

The Hasan applied multivariate Gaussian distribution to recognize hand gestures using non geometric features. The input hand image is segmented using two different methods; skin color-based segmentation by applying HSV color model and clustering based thresholding techniques. Kulkarni recognizes static posture of American Sign Language using neural networks algorithm. The input images are converted into HSV color model, resized into 80x64 and some image preprocessing operations are applied to segment the hand from a uniform background, features are extracted using histogram technique and Hough algorithm. Wysoski et al. presented rotation invariant postures using boundary histogram. Camera used for acquire the input image, filter for skin color detection has been used followed by clustering process to find the boundary for each group in the clustered image using ordinary contour-tracking algorithm.

Girshick et al. further improve upon Fast R-CNN introducing Faster RCNN (2017). They proposed a method using Region Proposal Network (RPN) which combines convolutional features with the detection network, providing extremely low cost region proposals. This model overcomes the bottleneck of region proposal computation. It also improves accuracy up to 73.2% mAP in the PASCAL VOC 2007 dataset. It has a small frame rate on GPU of 5fps which is expensive considering the real time detection.

Gestures are a natural form of human expression, and hands a natural mode of interaction with the physical world and objects in it (Zimmerman et al., 1987, Buchmann et al., 2004). Gestures are used for communication and accompany speech in many different forms. They range from gestures that do not convey a specific meaning and simply follow the rhythm of the speech, to those enriching its meaning and symbolizing specific concepts (McNeill, 1992, Quek, 2004).

Hand gestures in particular, including use of fingers and arms, are widely explored as a natural and intuitive interaction modality for a variety of applications. They are used as a sole, or one of the modes for interaction interfaces. It is believed that gesture based interfaces can reduce the complexity of interaction between humans and computers (New et al., 2003). Motivations behind the decision to use gestures in an interface can be varied. Gesture based interfaces used for computer applications can be more intuitive than established WIMP (Windows Icon Mouse Pointer) based interfaces, and allow inexperienced users to interact with computer applications, without undertaking extensive training (Buchmann et al., 2004, Kim et al., 2005, Beyer and Meier, 2011). In medical applications or industrial environments, they enable touchless operation guaranteeing sterility or safer interaction. Gesture interfaces for Virtual Reality (VR) or Augmented Reality (AR) environments provide better immersion and do not require conscious attention dedicated to the specific gestures being performed (Deller et al., 2006). Spatial concepts can be expressed using gestures, and they are used in design and engineering, when externalising ideas (Vinayak et al., 2013). Interaction with comfort functions in a car can be achieved without taking the eyes off the road (Riener et al., 2013). Gestures can be used to help older population achieve easier interaction with electronic devices (Bhuiyan and Picking, 2011). These are just some of the examples, and new applications are constantly being developed. Use of gestures for these applications is supported by a variety of technologies. Development of Kinect (Kinect, 2018) and LEAP sensors (LEAP MOTION INC., 2018), which are portable and supported by Software Development Kits (SDKs) enabling simpler implementation, seems to have contributed significantly to the expansion of the field on gesture based interfaces since 2013.

While the gesture-based interfaces are being developed for various applications clear standards which could guide their further development are not apparent. For example, while interfaces supporting three-dimensional (3D) object manipulation exploring use of intuitive, affordable and nonintrusive interfaces are ubiquitous, none of the approaches used have been established as the baseline for future development (Vinayak et al., 2013). Investigation of patterns of gesture use, identifying commonalities and differences between different fields would be an initial step towards development of a standard framework for gesture elicitation for

interaction interface development.

Review by Rautaray and Agrawal (2015), provides a survey of the gesture based research published up to and including 2012, and the content covered is largely that published prior to the uptake of Kinect and LEAP in the gesture research community. Reviews by Hasan and Mishra (2012), Suarez and Murphy (2012), and Pisharady and Saerbeck (2015) focus on recognition approaches. Three systematic reviews have been identified: one that focuses on usability guidelines for “health serious games” (Milani et al., 2017), one that focuses on data exchange formats (Santos¹ et al., 2015), and one that focuses on vision based gesture systems and algorithms for gesture recognition (AlShamayleh et al., 2018). The first one is in Portuguese and reports on only 16 studies. Reviews identified in the literature either cover recognition based research questions, rather than gestures themselves, or are not systematic, and information on patterns of gesture use cannot be extracted from them. Further to this, the underpinning gesture theory found in the literature is heavily based on gestures observed as speech aid, gestures used in parallel with verbal communication.

2.1 Existing System

To assist differently abled people with respect to speaking and hearing, translators were always a must. Since sign language is not known and understood by many, a translator was always necessary. However, it's not very convenient for a translator to be available at all times, and sometimes not convenient in particular settings. To overcome this, the Gesture Recognition for Deaf and Dumb is used to eliminate the need of a translator. The application translates sign language and enables the opposite person to understand what is to be conveyed.

2.2 Limitations of the system

- Gesture are difficult in understanding, informal etiquette, information might get distorted, etc.
- It is not precise and sometimes it is vague and plain.
- It is one of the informal types of communication, where it is not suited for official purposes
- One cannot make long explanation or conversation through gesture

CHAPTER-3

SOFTWARE REQUIREMENT SPECIFICATION

This chapter gives an overview of the software and hardware components required for our project.

3.1 SOFTWARE REQUIREMENTS

- Operating System : Windows 10 or MAC OS
- Platform : Anaconda, Spyder
- Software : Python, OpenCV, NumPy, Pandas

3.2 HARDWARE REQUIREMENTS

- Processor : Intel i5 and above
- RAM : 4GB and above
- Hard Disk : 8GB and above

3.3 FUNCTIONAL REQUIREMENTS:

For documenting the functional requirements, the set of functionalities supported by the system are to be Specified. A function can be specified by identifying the state at which data is to be input to the system, Its input data domain, the output domain, and the type of processing to be carried on the input data to Obtain the output data. Functional requirements define specific behavior or function of the application. Following are the functional requirements:

1. The main feature is to translate the recognized gesture into the textual meaning of the gesture and Display the translated text to the user.
2. System will recognize the appropriate movement of the hands and will search patterns to match the Movement with the pre-defined gestures of the ASL. Student should be able to view his/her complaint details and complaint history.
3. After matching system, will add confidence of the meaning of the gesture i.e., it's values from A-Z.
4. Formation of words using the given characters to generate meaningful words and phrases.

3.4 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Especially these are the constraints the system must work within. Following are the non-functional requirements:

1. Performance: Correct alphabet of the hand gesture in the input image will be recognized with an accuracy of about 95% and more.
2. Availability: This system will retrieve the regions only if the image contains a hand gesture in it.

3. Flexibility: It provides the system to understand the region even if the room is dimly lit.
4. Learnability: The software is very easy to use and reduces the learning work.
5. Reliability: This software will work reliably for low resolution images and not for graphical images.

CHAPTER-4

SYSTEM DESIGN

4.1 System Design

In this phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Low-Level Design (LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

4.2 UML Design:

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modeling Language as a standard in 1997. It's been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

Do we really need UML?

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non programmer's essential requirements, functionalities and processes of the system.
- A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
- UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.

- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

Types of UML Diagrams:

Structural Diagrams:

Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

Behavior Diagrams:

Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

The image below shows the hierarchy of diagrams according to UML

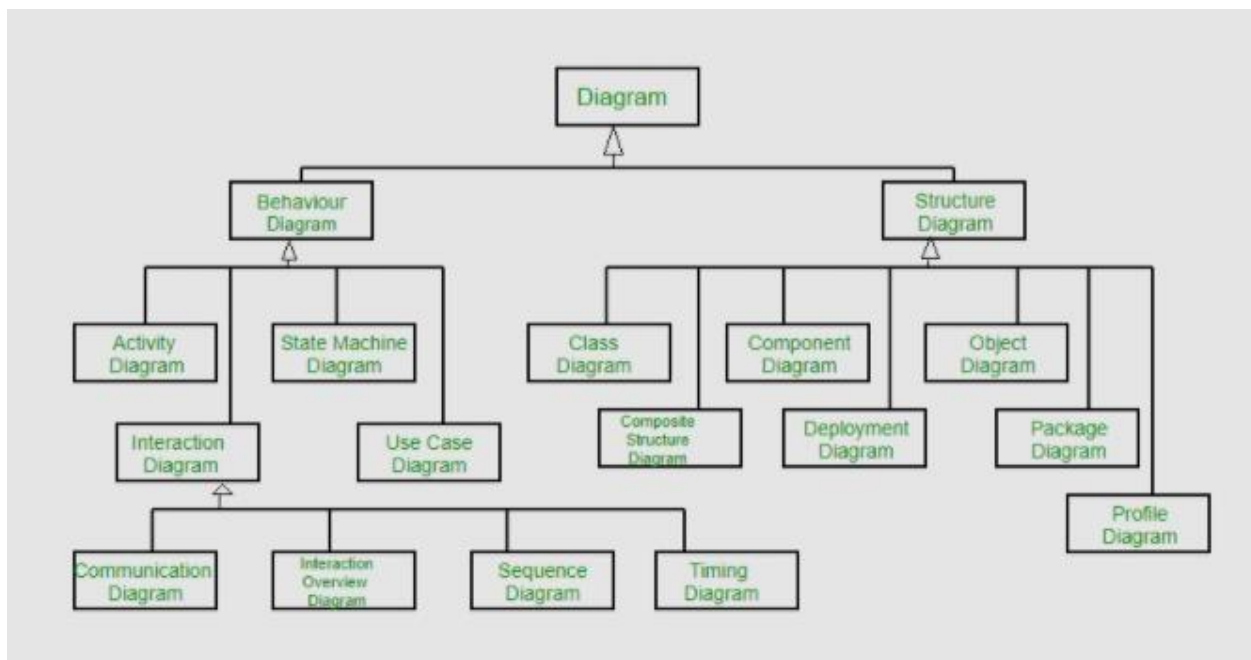


Figure-4.2.1 UML Hierarchy diagrams

4.2.1 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

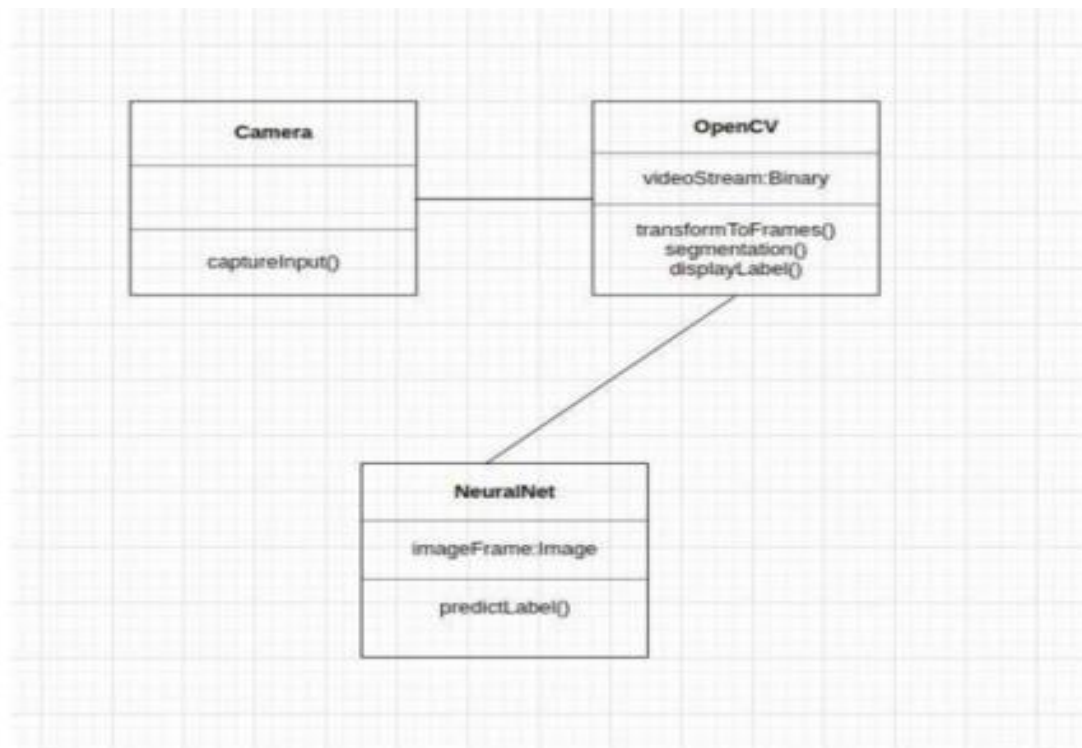


Figure-4.2.1.1 Class Diagram

4.2.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

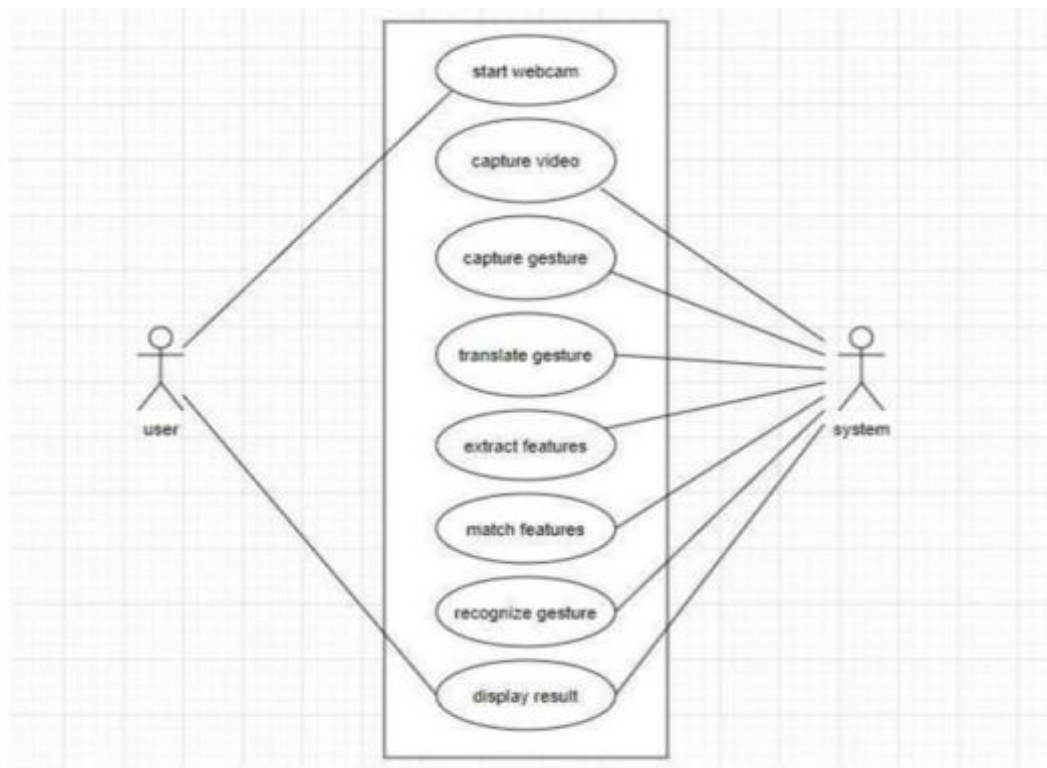


Figure-4.2.2.1 Use Case Diagram

4.2.3 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

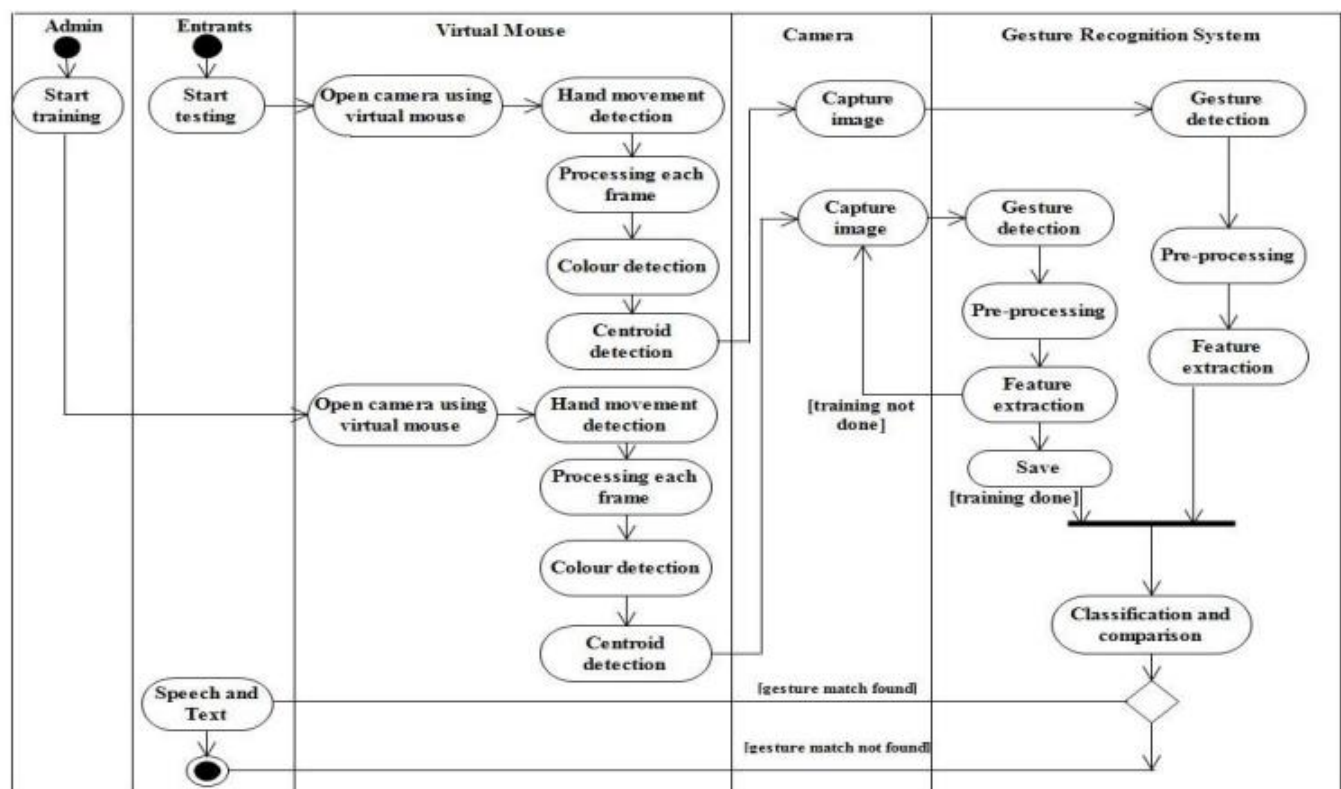


Figure-4.2.3.1 Activity Diagram

4.2.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

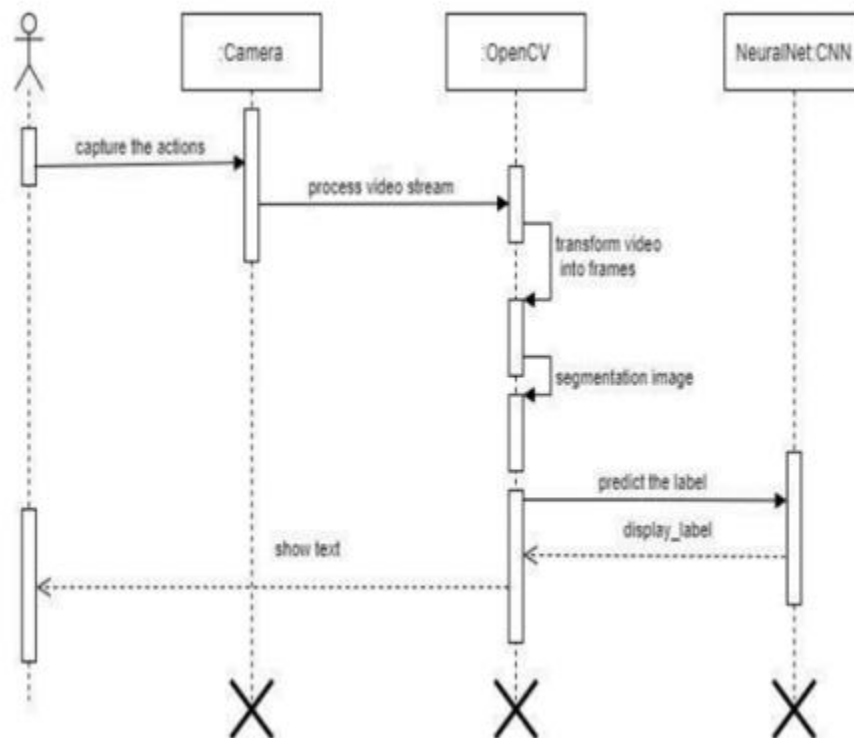


Figure-4.2.4.1 Sequence Diagram

4.3 SYSTEM ARCHITECTURE:

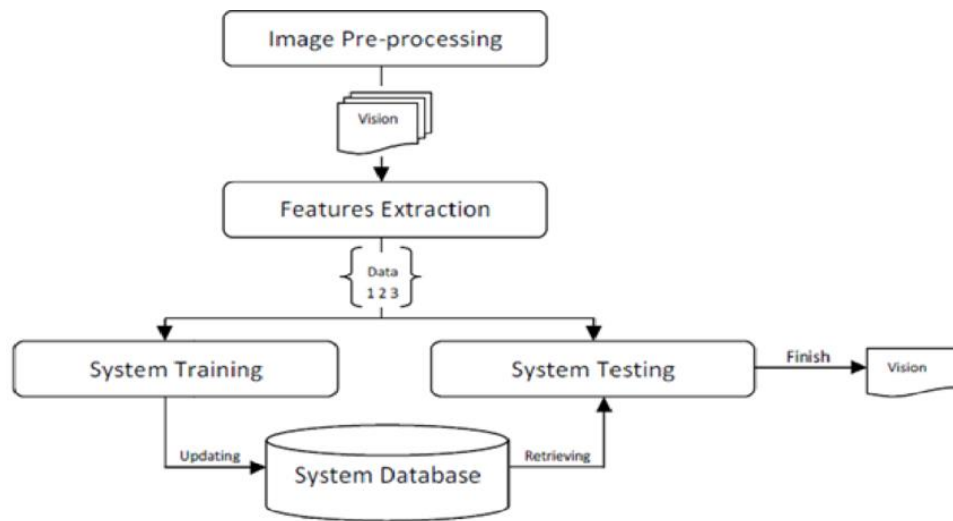


Figure-4.3.1 System Architecture

4.1 DATA FLOW DIAGRAM

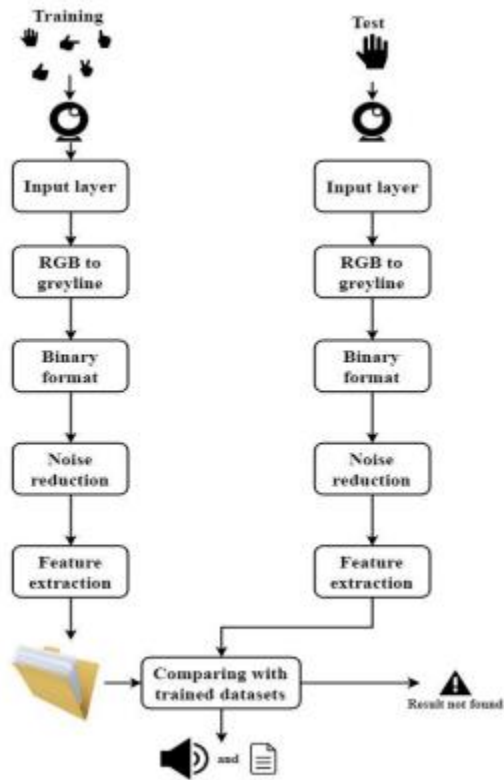


Figure 4.4.1 Data Flow Diagram

CHAPTER-5

IMPLEMENTATION

5.1 Libraries:

The Python programming language is an Open Source, cross-platform, high level, dynamic, interpreted language.

The Python 'philosophy' emphasizes readability, clarity and simplicity, whilst maximizing the power and expressiveness available to the programmer. The ultimate compliment to a Python programmer is not that his code is clever, but that it is elegant. For these reasons Python is an excellent 'first language', while still being a powerful tool in the hands of the seasoned and cynical programmer.

Python is a very flexible language. It is widely used for many different purposes. Typical uses include :

- Web application programming with frameworks like Zope, Django and Turbogears
- System administration tasks via simple scripts
- Desktop applications using GUI toolkits like Tkinter or wxPython (and recently Windows Forms and IronPython)
- Creating windows applications, using the Pywin32 extension for full windows integration and possibly Py2exe to create standalone programs
- Scientific research using packages like Scipy and Matplotlib

5.1.1 Tensor Flow :

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. Features: TensorFlow provides stable Python

(for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB, R, Scala, Rust, O. Caml, and Crystal. “New language support should be built on top of the C API. However, not all functionality is available in C yet.” Some more functionality is provided by the Python API. Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as Deep Dream

5.1.2 OpenCV:

OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding

5.1.3 Keras:

Keras is an open-source neural-network library written in Python. It is capable of running on top of Tensor Flow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the Xception deep neural network model.

Features: Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the L1 coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA. Keras applications module is used to provide pre-trained model for deep neural networks. Keras models are used for prediction, feature extraction and fine tuning. This chapter explains about Keras applications in detail.

5.1.4 NumPy:

NumPy is a library for the Python programming language, adding support for large, multidimensional arrays And matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, 12 was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Features: NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode

interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

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

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as threedimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

5.1.5 Neural Networks:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

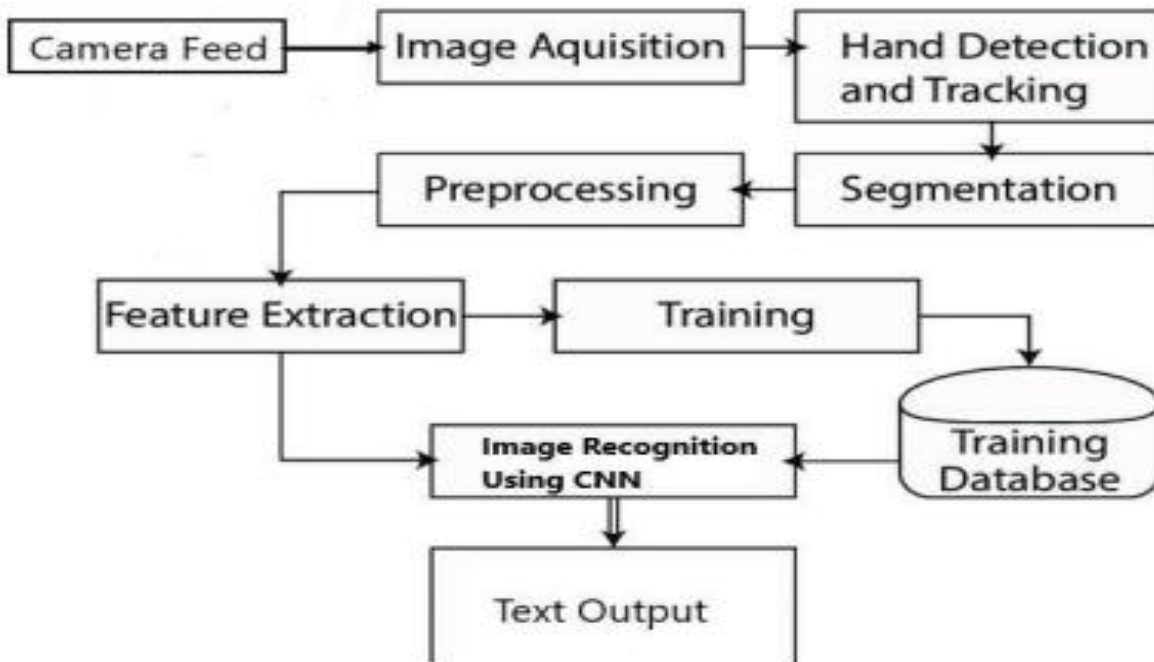
In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a

mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear. 14 In a multi-layered perceptron (MLP), perceptron's are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

5.2 IMPLEMENTATION DESGIN:



5.2.1 Implementation Design

5.3 Various Datasets:



5.3.1 Various Datasets

5.4 Dataset preparation and preprocessing

Data is the foundation for any machine learning project. The stage of project implementation is complex and involves data collection, selection, preprocessing, and transformation. Each of these phases can be split.

Data collection

Sources of collecting relevant and comprehensive data, interpreting it, and analyzing results with the help of statistical techniques.

Data Visualization

A large amount of information represented in graphic form is easier to understand and analyze. Some companies specify that a person must know how to a gesture.

Labeling

Data labeling takes much time and effort as datasets sufficient for machine learning may require thousands of records to be labeled. For instance, if your image recognition algorithm must classify types of bicycles, these types should be clearly defined and labeled in a dataset.

Data selection

After having collected all information, it chooses a subgroup of data to solve the defined problem. For instance, if you save your customers' the selected data includes attributes that need to be considered when building a predictive model.

Data preprocessing

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a person to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

- **Data formatting:** The importance of data formatting grows when data is acquired from various sources by different people. The first task for a person is to standardize record formats.
- **Data cleaning:** This set of procedures allows for removing noise and fixing inconsistencies in data. A data scientist can fill in missing data using imputation techniques, e.g. substituting missing values with mean attributes.
- **Data anonymization:** Sometimes a person must anonymize or exclude attributes representing sensitive information.
- **Data sampling:** Big datasets require more time and computational power for analysis. If a dataset is too large, applying data sampling is the way to go. A person uses this technique to select a smaller but representative data sample to build and run models much faster, and at the same time to produce accurate outcomes

5.5 Executable Code

Part 1 - Building the CNN

#importing the Keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense, Dropout

from keras import optimizers

Initialing the CNN

classifier = Sequential()

Step 1 - Convolutio Layer

classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

#step 2 - Pooling

classifier.add(MaxPooling2D(pool_size =(2,2)))

Adding second convolution layer

classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))

classifier.add(MaxPooling2D(pool_size =(2,2)))

#Adding 3rd Concolution Layer

classifier.add(Convolution2D(64, 3, 3, activation = 'relu',padding='same'))

classifier.add(MaxPooling2D(pool_size =(2,2),padding='same'))

#Step 3 - Flattening

classifier.add(Flatten())

#Step 4 - Full Connection

```
classifier.add(Dense(256, activation = 'relu'))  
classifier.add(Dropout(0.5))  
classifier.add(Dense(26, activation = 'softmax'))
```

#Compiling The CNN

```
classifier.compile(  
    optimizer = optimizers.SGD(lr = 0.01),  
    loss = 'categorical_crossentropy',  
    metrics = ['accuracy'])
```

#Part 2 Fitting the CNN to the image

```
from keras.preprocessing.image import ImageDataGenerator  
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
```

```
training_set = train_datagen.flow_from_directory(  
    'mydata/training_set',  
    target_size=(64, 64),  
    batch_size=32,  
    class_mode='categorical')
```

```
test_set = test_datagen.flow_from_directory(  
    'mydata/test_set',  
    target_size=(64, 64),  
    batch_size=32,  
    class_mode='categorical')
```

```
model = classifier.fit_generator(
    training_set,
    steps_per_epoch=800,
    epochs=25,
    validation_data = test_set,
    validation_steps = 6500
)

# model.compile(loss="categorical_crossentropy", metrics=["acc"])

"""#Saving the model
import h5py
classifier.save("Trained_model.h5")"""

print(model.history.keys())
import matplotlib.pyplot as plt
# summarize history for accuracy
plt.plot(model.history['accuracy'])
plt.plot(model.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss

plt.plot(model.history['loss'])
plt.plot(model.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
```

```
plt.legend(['train', 'test'], loc='upper left')  
plt.show()
```

```
import cv2  
import numpy as np
```

```
def nothing(x):  
    pass
```

```
image_x, image_y = 64,64
```

```
from keras.models import load_model  
from keras.utils import load_img  
from keras.utils import img_to_array  
classifier = load_model('Trained_model.h5')
```

```
def predictor():  
    import numpy as np  
    from keras.preprocessing import image  
    test_image = load_img('1.png', target_size=(64, 64))  
    test_image = img_to_array(test_image)  
    test_image = np.expand_dims(test_image, axis = 0)  
    result = classifier.predict(test_image)  
  
    if result[0][0] == 1:  
        return 'A'  
    elif result[0][1] == 1:
```

```
        return 'B'
    elif result[0][2] == 1:
        return 'C'
    elif result[0][3] == 1:
        return 'D'
    elif result[0][4] == 1:
        return 'E'
    elif result[0][5] == 1:
        return 'F'
    elif result[0][6] == 1:
        return 'G'
    elif result[0][7] == 1:
        return 'H'
    elif result[0][8] == 1:
        return 'I'
    elif result[0][9] == 1:
        return 'J'
    elif result[0][10] == 1:
        return 'K'
    elif result[0][11] == 1:
        return 'L'
    elif result[0][12] == 1:
        return 'M'
    elif result[0][13] == 1:
        return 'N'
    elif result[0][14] == 1:
        return 'O'
    elif result[0][15] == 1:
        return 'P'
    elif result[0][16] == 1:
        return 'Q'
    elif result[0][17] == 1:
```

```
        return 'R'
    elif result[0][18] == 1:
        return 'S'
    elif result[0][19] == 1:
        return 'T'
    elif result[0][20] == 1:
        return 'U'
    elif result[0][21] == 1:
        return 'V'
    elif result[0][22] == 1:
        return 'W'
    elif result[0][23] == 1:
        return 'X'
    elif result[0][24] == 1:
        return 'Y'
    elif result[0][25] == 1:
        return 'Z'
```

```
cam = cv2.VideoCapture(0)
```

```
text='Default'
```

```
while True:
```

```
    ret, frame = cam.read()
```

```
    frame = cv2.flip(frame,1)
```

```
    cv2.putText(frame,text,(100, 400),cv2.FONT_HERSHEY_SIMPLEX,1.5,(0, 0, 255),5)
```

```
    img = cv2.rectangle(frame, (425,100),(625,300), (0,0,255), thickness=3, lineType=8, shift=0)
```

```
    if not ret:
```

```
        print("failed to grab frame")
```

```
        break
```

```
cv2.imshow("Detect Hand Gestures", frame)

k = cv2.waitKey(1)
if k%256 == 27:
    # ESC pressed
    print("Escape hit, closing...")
    break
elif k%256 == 32:
    # SPACE pressed
    img_name = "1.png"
    imcrop = img[103:297, 428:622]
    bw_img=cv2.cvtColor(imcrop.copy(),cv2.COLOR_BGR2GRAY)
    gaussian=cv2.GaussianBlur(bw_img,(11,11),0)

    thresholded=cv2.adaptiveThreshold(gaussian,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
    imgCopy = thresholded.copy()
    contours,
    hierarchy=cv2.findContours(thresholded,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    # print(contours)
    #sort the contours as per the area
    contours=sorted(contours,key=cv2.contourArea,reverse=True)
    contours=[contours[0]]

    mask=np.zeros(imcrop.shape,dtype='uint8')
    mask=cv2.drawContours(mask,contours,-1,(255,255,255),thickness=cv2.FILLED)
    mask=cv2.bitwise_not(mask)
    mask=cv2.resize(mask,(image_x,image_y))
    cv2.imshow("Mask",mask)
    cv2.imwrite(img_name,mask)

    text=predictor()
```

```
# cv2.putText(frame,text,(100, 400),cv2.FONT_HERSHEY_TRIPLEX,1,(255, 0, 0),20,cv2.LINE_AA)  
print(text)
```

```
cam.release()
```

```
cv2.destroyAllWindows()
```


CHAPTER-6

TESTING

6.1 TESTING DEFINATION:

- Software testing is the process of valuating and verifying that a software product or application does what it is supposed to do.
- The benefits of testing include preventing bugs, reducing development costs and improving performance.

TESTING AND TEST CASES:

- Software testing is the process of valuating and verifying that a software product or application does what it is supposed to do.
- The benefits of testing include preventing bugs, reducing development costs and improving performance.

6.2 TYPES OF TESTING:

White Box Testing

In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers.

Black Box Testing

Another type of manual testing is **black-box testing**. In this testing, the test engineer will analyze the software against requirements, identify the defects or bug, and sends it back to the development team.

Functional Testing

The test engineer will check all the components systematically against requirement specifications is known as **functional testing**. Functional testing is also known as **Component testing**.

Non-function Testing

The next part of black-box testing is **non-functional testing**. It provides detailed information on software product performance and used technologies.

Grey Box Testing

Another part of **manual testing** is **Grey box testing**. It is a **collaboration of black box and white box testing**.

Since, the grey box testing includes access to internal coding for designing test cases. Grey box testing is performed by a person who knows coding as well as testing.

6.3 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

The unit test unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown

code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

To achieve this, unit test supports some important concepts in an object-oriented way:

6.4 Test Cases:

6.4.1 Load the Model

Test scenario ID		Load-1		Test case ID		Model_Load-1A	
Test case description		Load model		Test case priority		High	
Pre-requisite		Convolutional neural network file		Post-requisite		model	
Test Execution steps							
S.No	Action	Inputs	Expected output	Actual Output	Test Environment	Test Result	Test comments
1	Load model	.h5 file	Model parameters stored in model	Model parameters stored in model	Anaconda	Pass	Model loaded successfully
2	Display the weights and perform recognition on sample data for verification.	Sample hand gesture images	Translated English alphabets	Translated English alphabets	Anaconda	Pass	Model setup successful

Test Case:6.4.1.1

6.4.2 Capture Frames from Video:

Test scenario ID	Capture -1		Test case ID	Capture frames-1A			
Test case description	Capture frames from video		Test case priority	High			
Pre-requisite	Input video stream		Post-requisite	Pre-processed gesture frames.			
Test Execution steps							
S.No	Action	Inputs	Expected output	Actual Output	Test Environm ent	Test Result	Test comments
1	Capture frames from input video stream with gestures in ROI.	Video stream	Cropped frame with gesture	Cropped frame with gesture	Anaconda	Pass	Gesture frame captured successfully
2	Pre-processing of gesture frame.	Cropped gesture frame	Pre-processed gesture image.	Pre-processed gesture image.	Anaconda	Pass	Pre-processed image generated successfully.

Test Case:6.4.2.1

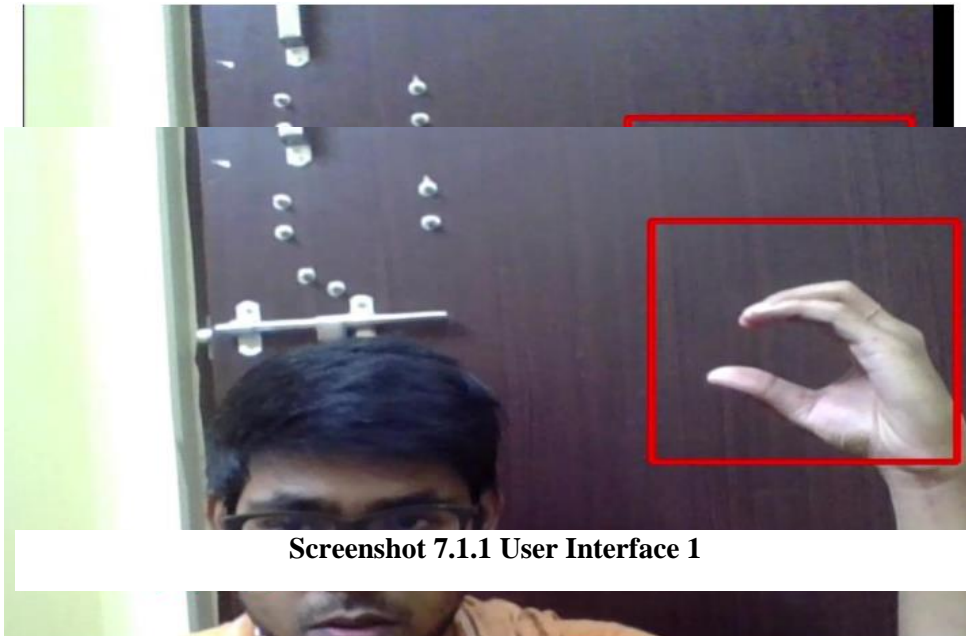
6.4.3 Hand Gesture Recognition:

Test scenario ID		Recognition-1		Test case ID		Recognition-1A	
Test case description		Translation of English equivalent for ASL.		Test case priority		High	
Pre-requisite		Pre-processed image.		Post-requisite		Translation of the ASL's English equivalent on the live input stream.	
Test Execution steps							
S.No	Action	Inputs	Expected output	Actual Output	Test Environment	Test Result	Test comments
1	Convert ASL to English	Pre-processed image	Confidence matrix of the gesture	Confidence matrix of the gesture	Anaconda	Pass	Alphabet recognized successfully.
2	Display the alphabet with maximum confidence on the input stream	NA	Alphabet around surrounding box	Alphabet around surrounding box	Anaconda	Pass	Alphabet displayed successfully

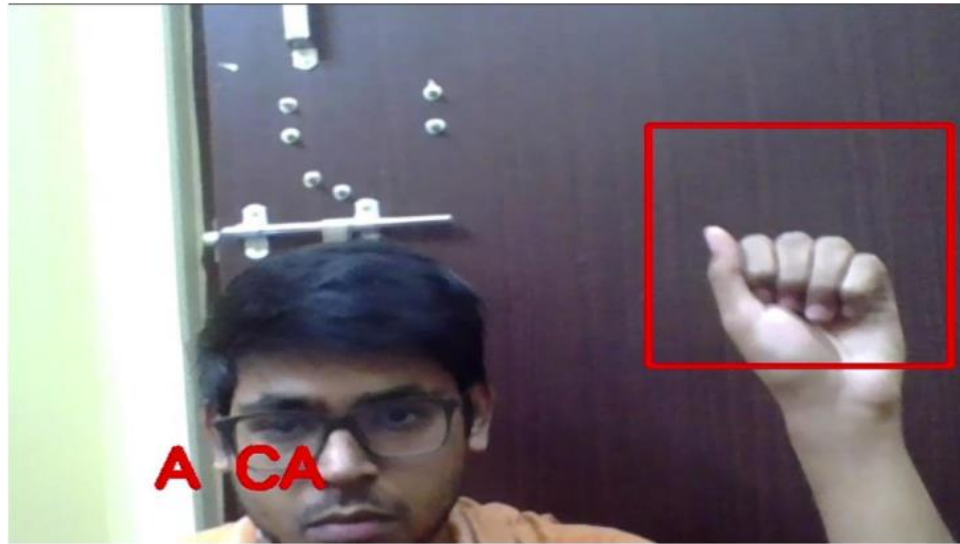
Test Case:6.4.3.1

CHAPTER-7

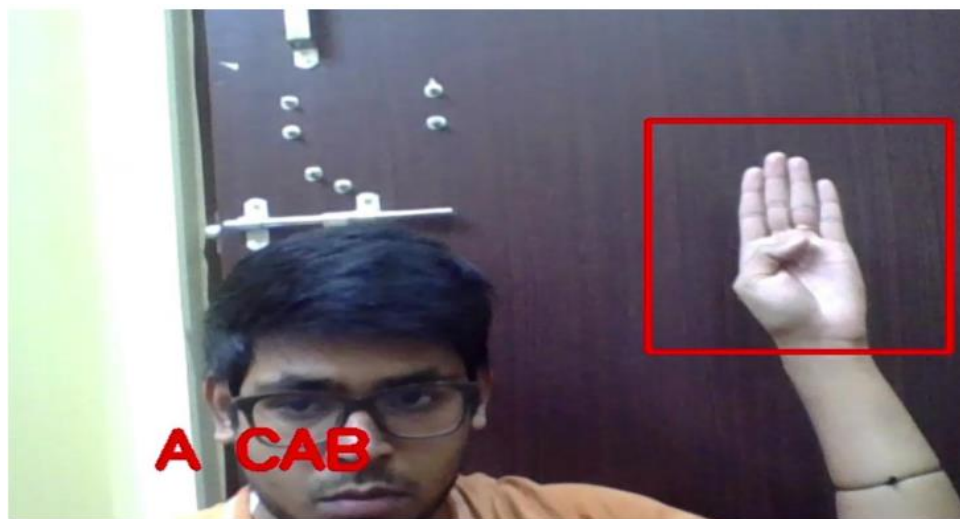
RESULTS



Screenshot 7.1.2 User Interface 2



Screenshot 7.1.3 User Interface 3



Screenshot 7.1.4 User Interface 4

CHAPTER-8

CONCLUSION

8.1 Conclusion

Nowadays, applications need several kinds of images as sources of information for elucidation and analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore, the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image. Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using various methods to make the image more readable by the computer.

Sign language recognition system is a powerful tool to prepare an expert knowledge, edge detect and the combination of inaccurate information from different sources. the intend of convolution neural network is to get the appropriate classification Future work. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language.

Hand Gesture Recognition and Text conversion helps Hearing Impaired person to communicate with others. CNN helps to converts hand gestures to text with good performance with maximum accuracy. As the system is trained for various affine transformations and simple backgrounds. As the input images do not have self-occlusion or inter object occlusion the recognition rate is maximum. The performance of Convolutional Neural Network algorithm is evaluated by changing the optimizers and number of filters. It is observed that the training accuracy increases with the Adam optimizer and increase in number of filters. In future the efficiency and accuracy of the training can be studied by varying CNN hyper parameters and architectures.

CHAPTER-9

FUTURE SCOPE

9.1 Future scope

There are some significant changes which can be brought to the existing system.

1. Scalable: It can be integrated with various search engines and texting application such as google, WhatsApp. So that even the illiterate people could be able to chat with other persons, or query something from web just with the help of gesture.

2. Extension to other modes of detection: This project is working on image currently; further development can lead to detecting the motion of video sequence and assigning it to a meaningful sentence with TTS assistance. The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. The scope of different sign languages can be increased. This project can further be extended to convert the signs to speech.

3. Variance of data sources and classes: More training data can be added to detect the letter with an increased accuracy that can be differentiated even if they look alike.

CHAPTER –10

REFERENCES

1. <https://www.nidcd.nih.gov/health/american-sign-language-fingerspelling-alphabets-image>
2. Farnaz D. Notash and Elahe Elhamki. “Comparing loneliness, depression and stress in students with hearingimpaired and normal students studying in secondary schools of Tabriz”. In: International Journal of Humanities and Cultural Studies February 2016 Special Issue (2016). issn: 2356-5926.
3. “The Cognitive, Psychological and Cultural Impact of Communication Barrier on Deaf Adults”. In: Journal of Communication Disorders, Deaf Studies Hearing Aids 4 (2 2016). doi: 10.4172/2375-4427.1000164.
4. Vivek Bheda and Dianna Radpour. “Using Deep Convolutional Networks for Gesture Recognition in American Sign Language”. In: CoRR abs/1710.06836 (2017). arXiv: 1710.06836. url: <http://arxiv.org/abs/1710.06836>

