Project

On

Sign Language Recognition Using Python

Bachelor of Technology
in
Information Technology

SUBMITTED BY:

Tapaswini Haldar	2112100041
Selina Nayak	2001106516
Stuti Pattnaik	2001106520

UNDER THE GUIDANCE OF

MR. DEBI PRASAD MISHRA

Faculty, SCS

SCHOOL OF COMPUTER SCIENCES

ODISHA UNIVERSITY OF TECHNOLOGY & RESEARCH

(FORMERLY KNOWN AS COLLEGE OF ENGINEERING AND TECHNOLOGY)

BHUBANESWAR, ODISHA

Project
On
Sign Language Recognition Using Python

a project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology
in
Information Technology

SUBMITTED BY:

TAPASWINI HALDAR	2112100041
SELINA NAYAK	2001106516
STUTI PATTNAIK	2001106520

UNDER THE GUIDANCE OF

MR. DEBI PRASAD MISHRA

Faculty, SCS

SCHOOL OF COMPUTER SCIENCES

ODISHA UNIVERSITY OF TECHNOLOGY & RESEARCH

(FORMERLY KNOWN AS COLLEGE OF ENGINEERING AND TECHNOLOGY)

BHUBANESWAR, ODISHA

ODISHA UNIVERSITY OF TECHNOLOGY & RESEARCH

(FORMERLY KNOWN AS COLLEGE OF ENGINEERING AND TECHNOLOGY)

BHUBANESWAR, ODISHA

Certificate

This is to certify that the project entitled, “Sign Language Recognition Using Python and Open CV”, is a bonafide work done by**” Tapaswini Haldar , Selina Nayak, Stuti Pattnaik”** in partial fulfillment of requirements for the award of Bachelor of Technology Degree in Information Technology at “**Odisha University of Technology & Research**” is an authentic work carried out by them under my supervision and guidance. The matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree or Diploma to the best of my knowledge.

Date: (Guide)

Place: OUTR,Bhubaneswar

(Head of the School)

ODISHA UNIVERSITY OF TECHNOLOGY & RESEARCH

(FORMERLY KNOWN AS COLLEGE OF ENGINEERING AND TECHNOLOGY)

BHUBANESWAR, ODISHA

Acknowledgement

We would like to express our sincere regards and profound sense of gratitude to those who have helped us in completing the project successfully. At the very outset, our special and heartfelt thanks to our project supervisor **Mr. Debi Prasad Mishra. Faculty, SCS** for his/her precious guidance, assistance and constant supervision to bring this piece of work into the present form.

We also extend our humble gratitude to **Prof. Jibitesh Mishra, Head, School of Computer Sciences** for providing the opportunity to present this project in the university.

TAPASWINI HALDAR:	**2112100041**
SELINA NAYAK:	**2001106516**
STUTI PATTNAIK :	**2001106520**

ODISHA UNIVERSITY OF TECHNOLOGY & RESEARCH
(FORMERLY KNOWN AS COLLEGE OF ENGINEERING AND TECHNOLOGY)
BHUBANESWAR, ODISHA

Declaration

We declare that this written submission represents our ideas in our own words and wherever others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated any idea/data/fact/source in our submission. We understand that any violation of the above will cause for disciplinary action by the Institute as deemed fit.

TAPASWINI HALDAR 2112100041

SELINA NAYAK 2001106516

STUTI PATTNAIK 2001106520

Abstract

Sign language recognition is a project which utilizes advancements in technology to assist especially aided individuals. We planned to build a sign detector for this purpose. Through this paper, we are trying to recognize and display messages according to hand movements. The growth in the existing technologies, as well as extensive research, is used to assist the deaf and dumb. This can be extremely useful for deaf and a dumb person in interacting with others, as understanding sign language is not something that many possess. Physically challenged people can express their emotions and feelings through sign language Here we develop a sign detector that can detect signs and numbers and also other signs that are used in sign language with the help of OpenCV and Keras modules in python. By using this technology, we can understand what they want to convey through sign language which is not a common language to communicate with people. The proposed work proved to be a user-friendly approach to communication by using Python language to recognize sign languages for hearing-impaired persons.

Guided by
Mr. Debi Prasad Mishra

Submitted by:
Selina Nayak
Stuti Pattnaik
Tapaswini Haldar

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.INTRODUCTION

Sign languages vary throughout the world. There are around 300 different sign languages used across various parts of the world. This is because sign languages were developed naturally by people belonging to different ethnic groups.

According to the 2011 census, there are around 50 lakh people in India with speech/hearing impairments. But, there are only less than 300 educated and trained sign language interpreters in India. So, people with speech/hearing impairments tend to become isolated and lonely, as they face difficulties in communicating with other normal people.

.This system makes it easy for the specially challenged people to communicate effectively with the rest of the world. This could enhance their abilities and make them realize that they can do better in life. The proposed system performs Gesture to Text conversion . This has a tremendous effect on both their social and working life. Due to the above mentioned challenges that the specially challenged people face, an automated real-time system that could translate English words to ISL and vice versa has been proposed in this paper. This system makes it easy for the specially challenged people to communicate effectively with the rest of the world. This could enhance their abilities and make them realize that they can do better in life. The proposed system performs two major tasks: (i) Gesture to Text conversion and (ii) Speech to Gesture conversion. Gesture to text conversion is done using neural network classifiers. Speech to gesture conversion is done using Google Speech Recognition API. This paper focuses on conversion of standard Indian Sign Language gestures to English, and conversion of English words (spoken) to Indian Sign Language gestures with highest possible accuracy. For this, different neural network classifiers are developed, and their performance in gesture recognition is tested. The most accurate and efficient classifier is chosen and is used to develop an application that converts ISL gestures to their corresponding English text, and speech to the corresponding ISL gestures.

Sign Language Recognition (SLR) stands at the intersection of technology and inclusivity, offering a groundbreaking solution to enhance communication for individuals who are deaf or hard of hearing. Sign language is a rich and expressive form of communication, relying on intricate hand movements, facial expressions, and body language. Unfortunately, traditional communication tools often fall short in addressing the unique needs of the deaf community, necessitating the development of innovative technologies like SLR.

This project endeavors to create a system capable of understanding and translating sign language gestures into meaningful text or spoken language. Leveraging the advancements in computer vision, machine learning, and deep learning, the aim is to provide a reliable and efficient means for individuals with hearing impairments to communicate with the broader world. The potential impact of SLR is far-reaching, extending to improved accessibility, enhanced educational experiences, and increased social integration.

The significance of SLR lies in its ability to bridge the communication gap, fostering a more inclusive society. Deaf individuals often face challenges in expressing themselves and understanding others, hindering their access to education, employment, and social interactions. By developing a robust SLR system, we aim to empower individuals with hearing impairments, enabling them to participate more fully in various aspects of life.

Technically speaking, the main challenge of sign language recognition lies in developing descriptors to express hand-shapes and motion trajectory. In particular, hand-shape description involves tracking hand regions in video stream, segmenting hand-shape images from complex background in each frame and gestures recognition problems. Motion trajectory is also related to tracking of the key points and curve matching. Although lots of research works have been conducted on these two issues for now, it is still hard to obtain satisfying result for SLR due to the variation and occlusion of hands and body joints. Besides, it is a nontrivial issue to integrate the hand-shape features and trajectory features together. To address these difficulties, we develop a CNNs to naturally integrate hand-shapes, trajectory of action and facial expression. Instead of using commonly used color images as input to networks like [1, 2], we take color images, depth
images and body skeleton images simultaneously as input which are all provided by Microsoft Kinect.
Kinect is a motion sensor which can provide color stream and depth stream. With the public Windows SDK, the body joint locations can be obtained in real-time as shown in Fig.1. Therefore, we choose Kinect as capture device to record sign words dataset. The change of color and depth in pixel level are useful information to discriminate different sign actions. And the variation of body joints in time dimension can depict the trajectory of sign actions. Using multiple types of visual sources as input leads CNNs paying attention to the change not only in color, but also in depth and trajectory. It is worth mentioning that we can avoid the difficulty of tracking hands, segmenting

hands from background and designing descriptors for hands because CNNs have the capability to learn features automatically from raw data without any prior knowledge .
CNNs have been applied in video stream classification recently years. A potential concern of CNNs is time consuming. It costs several weeks or months to train a CNNs with million-scale in million videos. Fortunately, it is still possible to achieve real-time efficiency, with the help of CUDA for parallel processing. We propose to apply CNNs to extract spatial and temporal features from video stream for Sign Language Recognition (SLR). Existing methods for SLR use hand-crafted features to describe sign language motion and build classification model based on these features. In contrast, CNNs can capture motion information from raw video data automatically, avoiding designing features. We develop CNNs taking multiple types of data as input. This architecture integrates color, depth and trajectory information by performing convolution and sub sampling on adjacent video frames. Experimental results demonstrate that 3D CNNs can significantly outperform Gaussian mixture model with Hidden Markov model (GMM-HMM) baselines on some sign words recorded by ourselves.

Sign languages are nearly as old as the spoken languages. Plato, in his work Cratylus (fifth century BC), quotes Socrates, “If we hadn’t a voice or a tongue, and wanted to express things to one another, wouldn’t we try to make signs by moving our hands, head, and the rest of our body, just as dumb people do at present? [2]” Traditional Indian dances like Bharatnatyam strive to tell a tale by the positions of the hands and the body, the fluidity of motion and facial expressions. By some estimates it is more than 2000 years old.Though sign languages have existed in local cultures, standardization began only in the seventeenth century.

Sign languages have their own morphological structure and grammatical nuances. They exist in a variety of forms and features. Manual signing or finger-spelling refers to using a sequence of ‘alphabet signs’ for making an entire word and making a sentence by recursion. In such cases a wide vocabulary is not needed. The more traditional and widely used system has an entry in the sign dictionary for every word. A lot of variations occur in these signs from region to region but a few standardised systems like American Sign Language, British Sign Language, French Sign Language, etc. have emerged. There is also an alternative form of signed communication among the deaf—namely Cued Speech .Cued Speech is not sign language. It is a phonemic speechreading system in which a small number of hand gestures are used in conjunction with speechreading.

Signs may be categorised as being static and dynamic. Static signs involve a fixed hand pose, while dynamic signs involve movement of hands to imply a suggestive motion. Most sign language users use emotion and timing to express themselves better. This makes the language more malleable and hence more comprehensible. Expressions and timing make a world of difference when a non-signer is at the other end of the communication channel.

The journey to achieve effective Sign Language Recognition involves overcoming technical challenges related to diverse signing styles, variations, and real-time processing demands. This report will delve into the methodologies employed, including data collection, preprocessing, and model training. The ultimate goal is to create a system that not only recognizes a broad spectrum of sign language gestures but is also adaptable to different signing styles and regional variations.

Through this exploration, we aim to contribute to the broader conversation about accessibility and inclusivity in technology. The findings and outcomes of this project have the potential to influence future developments in assistive technologies, educational tools, and communication platforms, ultimately paving the way for a more inclusive and connected world for individuals with hearing impairments.

2.MOTIVATION:

Communication is one of the basic requirement for survival in society. Deaf and dumb people communicate among themselves using sign language but normal people find it difficult to understand their language. Extensive work has been done on American sign language recognition but Indian sign language differs significantly from American sign language.ISL uses two hands for communicating(20 out of 26) whereas ASL uses single hand for communicating. Using both hands often leads to obscurity of features due to overlapping of hands. In addition to this, lack of datasets along with variance in sign language with locality has resulted in restrained efforts in ISL gesture detection. Our project aims at taking the basic step in bridging the communication gap between normal people and deaf and dumb people using Indian sign language. Effective extension of this project to words and common expressions may not only make the deaf and dumb people communicate faster and easier with outer world, but also provide a boost in developing autonomous systems for understanding and aiding them.

The 2011 Indian census cites roughly 1.3 million people with “hearing impairment”. In contrast to that, numbers from India’s National Association of the Deaf estimates that 18 million people – roughly 1 percent of the Indian population are deaf. These statistics formed the motivation for our project. As these speech impairments and deaf people need a proper channel to communicate with normal people, there is a need for a system .Not all normal people can understand sign language of impaired people.

3.LITERATURE SURVEY

1) A Mobile Application of American Sign Language Translation via Image Processing Algorithms

AUTHORS: Cheok Ming Jin, Zaid Omar, Mohamed

Due to the relative lack of pervasive sign language usage within our society, deaf and other verbally-challenged people tend to face difficulty in communicating on a daily basis. Our study thus aims to provide research into a sign language translator applied on the smartphone platform, due to its portability and ease of use. In this paper, a novel framework comprising established image processing techniques is proposed to recognise images of several sign language gestures. More specifically, we initially implement Canny edge detection and seeded region growing to segment the hand gesture from its background. Feature points are then extracted with Speeded Up Robust Features (SURF) algorithm, whose features are derived through Bag of Features (BoF). Support Vector Machine (SVM) is subsequently applied to classify our gesture image dataset; where the trained dataset is used to recognize future sign language gesture inputs. The proposed framework has been successfully implemented on smartphone platforms, and experimental results show that it is able to recognize and translate 16 different American Sign Language gestures with an overall accuracy of 97.13%.

2) Hand gesture recognition based on ISL

Authors: Mr. Sanket Kadam, Mr. Aakash GhodkeProf. Sumitra Sadhukhan

Hand gesture recognition are a powerful environment for communicating with communities with intellectual disability. It is useful for connecting people and computers. The expansion potential of this system can be known in public places where deaf people are communicating with ordinary people to send messages. In this article, we have provided a system of recognizing gestures continuously with the Indian Sign Language (ISL), which both hands are used to make every gesture. Gesture recognition continues to be a daunting task. We tried to fix this problem using the key download method. These key tips are useful for breaking down the sign language gestures into the order of the characters, as well as deleting unsupported frameworks. After the splitting gear breaks each character is regarded as a single and unique gesture. Pre-processing gestures are obtained using histogram (OH) with PCA to reduce the dimensions of the traits obtained after OH. The experiments were performed on our live ISL dataset, which was created using an existing camera.

3) Real-time Indian Sign Language (ISL) Recognition

Authors: Karthik Shenoy, Tejas Dastane, Varun Rao, Devendra Vyavaharkar.

This paper presents a system which can recognise hand poses & gestures from the Indian Sign Language (ISL) in real-time using grid-based features This system attempts to bridge the communication gap between the hearing and speech impaired and the rest of the society. The existing solutions either provide relatively low accuracy or do not work in real-time. This system provides good results on both the parameters. It can identify 33 hand poses and some gestures from the ISL. Sign Language is captured from a smartphone camera and its frames are transmitted to a remote server for processing. The use of any external hardware (such as gloves or the Microsoft Kinect sensor) is avoided, making it user-friendly. Techniques such as Face detection, Object stabilisation and Skin Colour Segmentation are used for hand detection and tracking. The image is further subjected to a Grid-based Feature Extraction technique which represents the hand's pose in the form of a Feature Vector. Hand poses are then classified using the k-Nearest Neighbours algorithm. On the other hand, for gesture classification, the motion and intermediate hand poses observation sequences are fed to Hidden Markov Model chains corresponding to the 12 pre-selected gestures defined in ISL. Using this methodology, the system is able to achieve an accuracy of 99.7% for static hand poses, and an accuracy of 97.23% for gesture recognition.

4) A depth based ISL Recognition using Microsoft Kinect

Authors: T Raghuveera, R Deepthi, R Mangalashri And R Akshaya

Recognition of sign language by a system has become important to bridge the communication gap between the abled and the Hearing and Speech Impaired people. This paper introduces an efficient algorithm for translating the input hand gesture in Indian Sign Language (ISL) into meaningful English text and speech. The system captures hand gestures through Microsoft Kinect (preferred as the system performance is unaffected by the surrounding light conditions and object colour). The dataset used consists of depth and RGB images (taken using Kinect Xbox 360) with 140 unique gestures of the ISL taken from 21 subjects, which includes singlehanded signs, double-handed signs and fingerspelling (signs for alphabets and numbers), totaling to 4600 images. To recognize the hand posture, the hand region is accurately segmented and hand features are extracted using Speeded Up Robust Features, Histogram of Oriented Gradients and Local Binary Patterns. The system ensembles the three feature classifiers trained using Support Vector Machine to improve the average recognition accuracy up to 71.85%. The system then translates the sequence of hand gestures recognized into the the best approximate meaningful English sentences. We achieved 100% accuracy for the signs representing 9, A, F, G, H, N and P.

4.SYSTEM ANALYSIS

4.1Existing System:

The existing system for sign language interpretation relies heavily on human interpreters or the presence of individuals proficient in sign language. While this approach is effective, it can be challenging to provide real-time interpretation in various situations. Additionally, human interpreters may not always be readily available, leading to communication barriers for the hearing-impaired community. Therefore, there is a need for an automated and accurate system that can detect and interpret sign language gestures in real-time.

4.1.1 Disadvantages of Existing System:

1.Limited Accuracy: Traditional methods for sign language recognition may have limited accuracy in detecting and interpreting complex and subtle hand gestures, leading to errors in communication.
2. Lack of Real-time Processing: Many existing systems may not offer real-time processing capabilities, making it challenging to have natural and fluid sign language conversations in real-world scenarios.
3.Limited Vocabulary: Some systems may have a limited vocabulary and may struggle to recognize a wide range of signs and gestures, limiting their practical use.

4.High Dependency on Manual Features: In the existing system, feature extraction and selection are often performed manually, making it challenging to adapt to diverse signing styles and variations.
5.Scalability Issues: Traditional systems may encounter scalability issues when dealing with a large dataset of signs and gestures, especially for continuous sign language communication.

4.2.Proposed System:

The Sign Language Detection Using ML project proposes an intelligent solution to address the limitations of the existing system. By implementing machine learning algorithms, the system can recognize hand gestures from video input, allowing for immediate and accurate interpretation. The proposed system will utilize image processing techniques to extract relevant features from the hand gestures, which will then be fed into a machine learning model for classification. This enables the system to provide real-time interpretation of sign language, promoting inclusive communication.

4.2.1.Advantages Of Proposed System:

1.Improved Accuracy: Leveraging machine learning techniques, the proposed system can achieve higher accuracy in recognizing and interpreting a broader range of sign language gestures, reducing communication errors.
2.Real-time Processing: The system is designed for real-time processing, enabling fluid and natural sign language conversations, making it suitable for practical and dynamic communication scenarios.
3.Extensive Vocabulary: The proposed system can potentially recognize and interpret a more extensive vocabulary of signs and gestures, allowing for a richer and more versatile communication experience.
4.Automated Feature Learning: Machine learning models automate feature extraction and learning, adapting to diverse signing styles and variations more effectively than manual methods.
5.Scalability: The system can efficiently scale to handle a large dataset of signs and gestures, accommodating the needs of continuous sign language communication and expanding its usability.

6.Enhanced Accessibility: By accurately recognizing sign language, the proposed system promotes accessibility and inclusivity, bridging the communication gap between individuals with hearing impairments and those who do not understand sign language.

4.2.2. ALGORITHM

CNN Convolutional Neural Network:

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers**.**

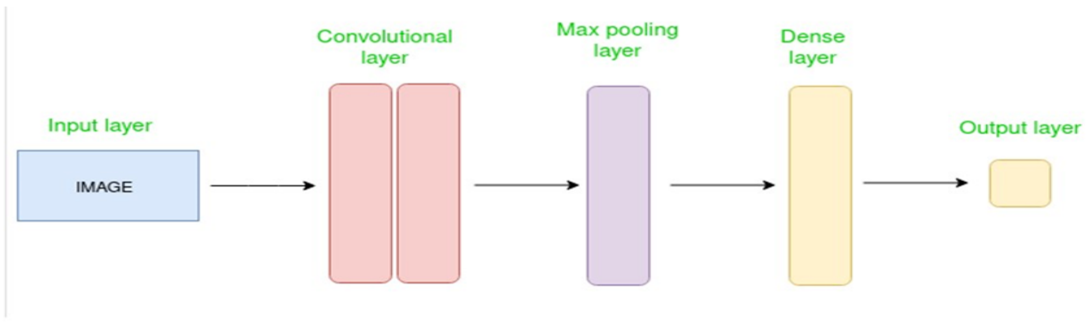

Fig 4.1.Convolutional Neural Network

The Sign Language Recognition System consists of several components:

•Data Collection: A dataset of sign language gestures is collected, including various hand shapes.

•Preprocessing: The collected data is pre-processed to enhance the quality, remove noise, and extract relevant features.

•Model Training: Machine learning models, i.e. convolutional neural networks (CNNs) is trained on the pre-processed data to learn the mapping between input gestures and their corresponding meanings.

•Real-time Recognition: The trained model is deployed in a real-time environment, where it takes video input and performs gesture recognition on the fly.

Convolutional Neural Networks (CNNs) represent a pivotal advancement in the field of deep learning, particularly in image and video analysis. These neural networks are designed to automatically and adaptively learn hierarchical features from input data, making them highly effective in tasks such as image recognition, object detection, and even natural language processing. Below are key concepts and characteristics of CNNs:

Convolutional Layers:

CNNs derive their name from the "convolutional" layers, where filters or kernels are applied to input data to extract relevant features.

These layers enable the network to automatically learn spatial hierarchies of features, capturing local patterns.

Pooling Layers:

Pooling layers follow convolutional layers, reducing spatial dimensions while retaining essential information.

Common pooling operations include max pooling, where the maximum value in a region is selected, aiding in translation invariance.

Activation Functions:

Non-linear activation functions like Rectified Linear Unit (ReLU) introduce non-linearity to the model, allowing it to learn complex relationships in the data.

Fully Connected Layers:

After feature extraction, CNNs often include fully connected layers for classification or regression tasks.

These layers connect all neurons from one layer to every neuron in the next layer, facilitating high-level abstraction.

Weight Sharing:

CNNs employ weight sharing, meaning the same set of parameters (weights and biases) is used for different parts of the input data.

This sharing of weights makes CNNs particularly efficient for processing grid-like data, such as images.

Transfer Learning:
CNNs are well-suited for transfer learning, where pre-trained models on large datasets (e.g., ImageNet) can be fine-tuned for specific tasks with smaller datasets.
This leverages the learned features from one task to improve performance on another.

Data Augmentation:
To prevent overfitting and improve generalization, CNNs often incorporate data augmentation techniques.
This involves applying transformations like rotation, scaling, and flipping to artificially expand the training dataset.

Applications:
CNNs find applications in various fields, from image and speech recognition to medical image analysis and autonomous vehicles.
Their ability to automatically learn hierarchical representations makes them versatile and powerful tools in machine learning.

In summary, CNNs have revolutionized the field of deep learning, particularly in tasks related to visual and spatial data. Their architecture, incorporating convolutional layers, pooling, and weight sharing, has proven highly effective in capturing intricate patterns and representations, making CNNs a cornerstone in contemporary artificial intelligence applications.

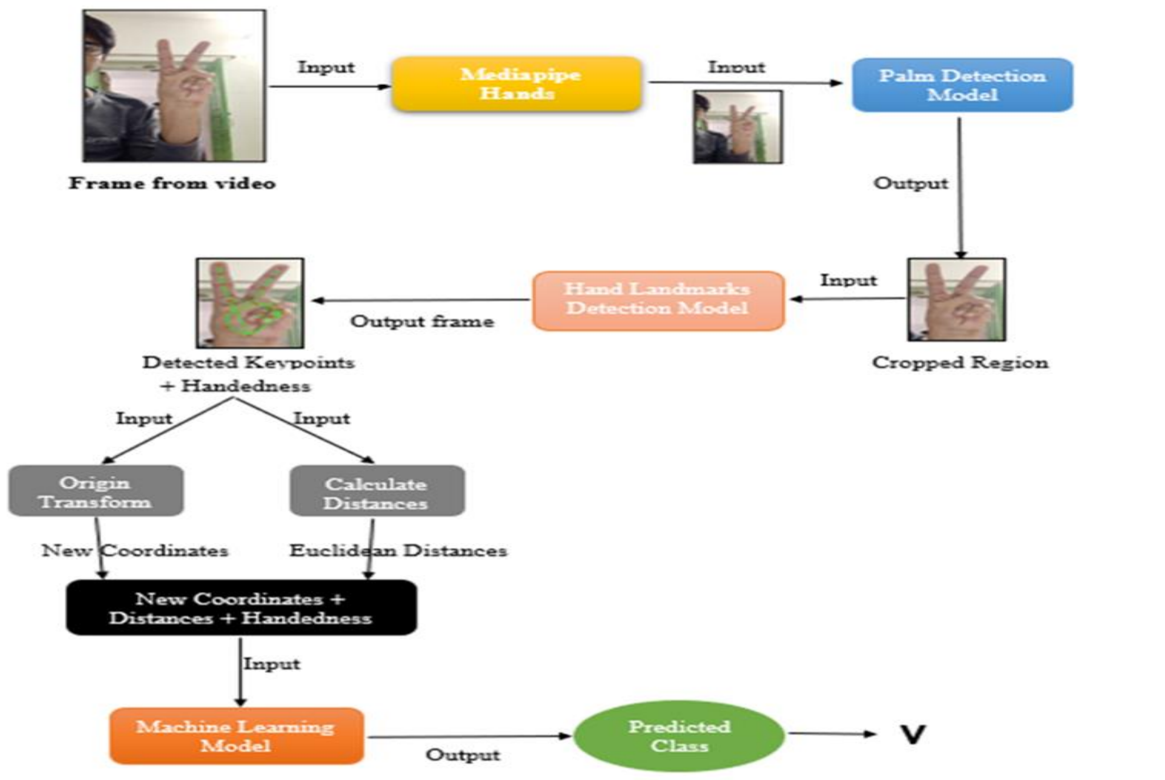

Fig 4.2.Proposed Model

4.3System Environment:

4.3.1Python:

Python is a general-purpose interpreted, interactive, object-oriented, and high- level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a community- based development model, as do nearly all of its variant implementations. Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.
•Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
•Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
•Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
•Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

4.3.2.Python Features

Python's features include:
•Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
•Easy-to-read: Python code is more clearly defined and visible to the eyes.
•Easy-to-maintain: Python's source code is fairly easy-to-maintain.
•A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
•Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
•Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

•Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
•Databases: Python provides interfaces to all major commercial databases.
•GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
•Scalable: Python provides a better structure and support for large programs than shell scripting.
Python has a big list of good features:
•It supports functional and structured programming methods as well as OOP.
•It can be used as a scripting language or can be compiled to byte-code for building large applications.
•It provides very high-level dynamic data types and supports dynamic type checking.
•It supports automatic garbage collection.
•It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

4.4. SYSTEM STUDY:

4.4.1.FEASIBILITY STUDY:

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

Economical Feasibility:

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

Technical Feasibility:

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

Social Feasibility:

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

4.5Requirement Analysis:

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

4.5.1Requirement Specification:

Functional Requirements: Graphical User interface with the User.

Software Requirements:

For developing the application the following are the

Software Requirements:

Operating Systems supported: Windows 10 64 bit OS

Technologies and Languages used to Develop:Python

Debugger and Emulator:Any Browser (Particularly Chrome)

Hardware Requirements:

For developing the application the following are the Hardware Requirements:

Processor: Intel i5

RAM: 8 GB

Space on Hard Disk: minimum 1 TB

HARDWARE REQUIREMENTS:

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

•System: Pentium IV 2.4 GHz.

•Hard Disk: 100 GB.

•Monitor: 15 VGA Color.

•Mouse: Logitech.

•RAM: 1 GB.

SOFTWARE REQUIREMENTS:

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

•Operating system: Windows XP/7/10

•Coding Language: Python 3.7

5.SYSTEM DESIGN

5.1.SYSTEM ARCHITECTURE:

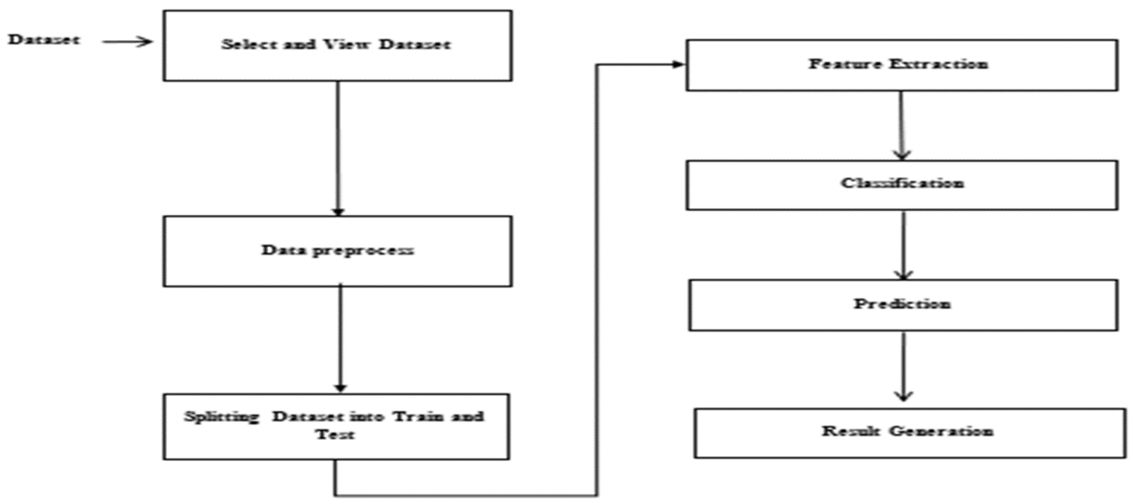

Fig5.1.System Architecture

5.2.UML Diagrams:

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object- oriented computer software. In its current form UML is comprised of two major components: a Meta-model

and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:
The Primary goals in the design of the UML are as follows:
1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6.Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.2.1Use 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.

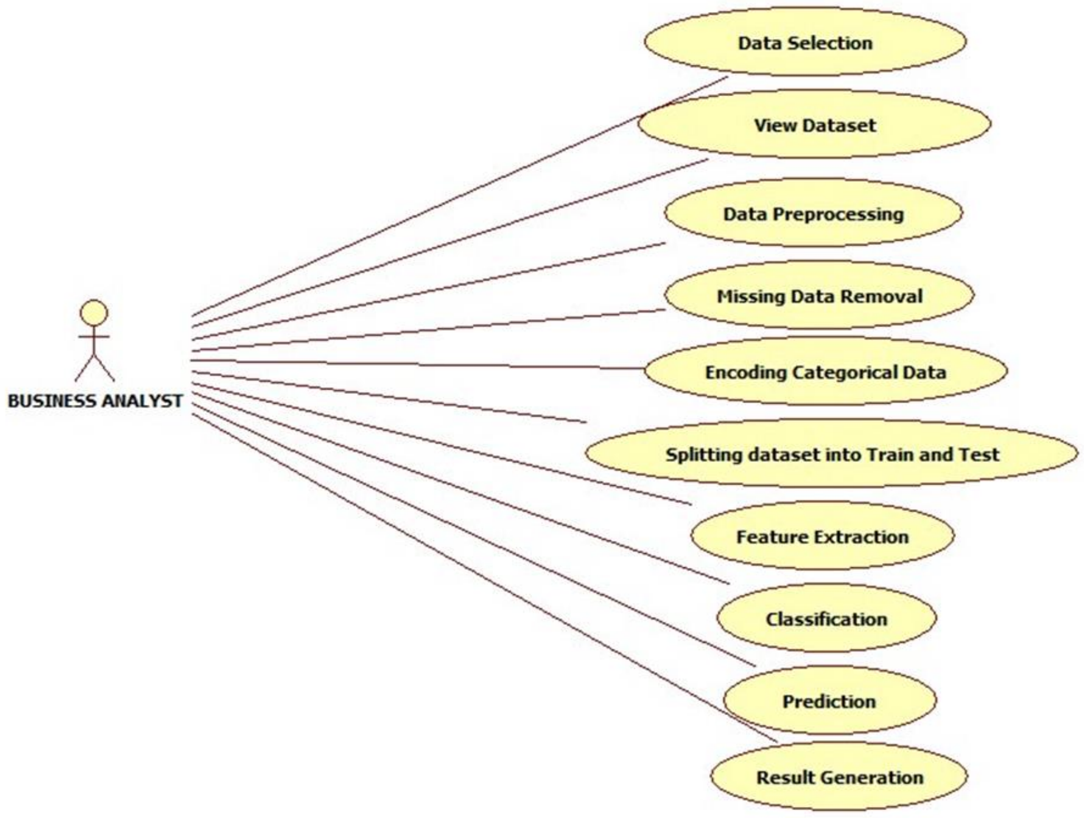

Fig5.2. Use case diagram

5.2.2.CLASS DIAGRAM

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 .

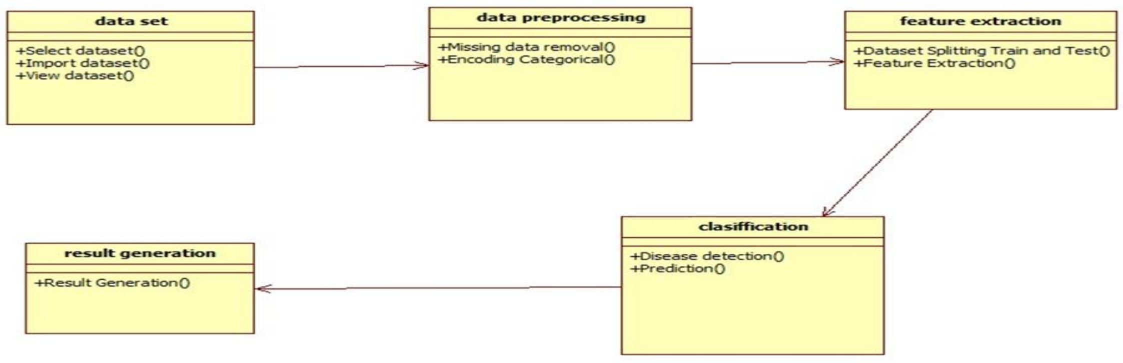

Fig5.3.Class Diagram

5.2.3.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.

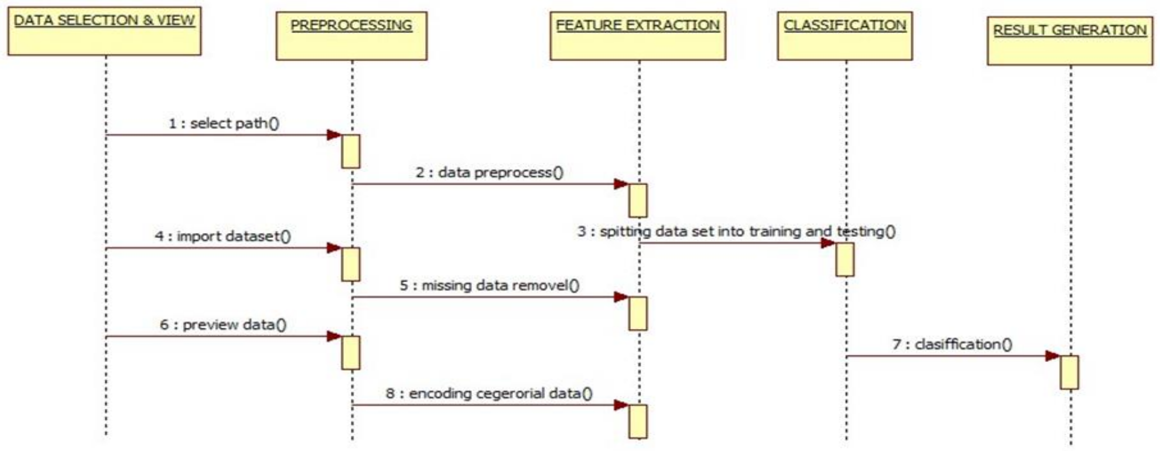

Fig 5.4.Sequence Diagram

5.2.4. Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

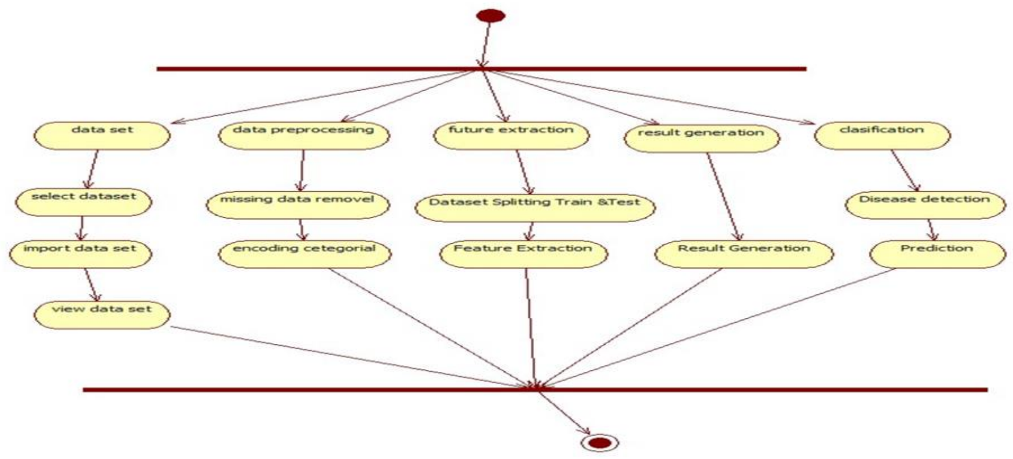

Fig 5.5.Activity Diagram

6.IMPLEMENTATION:

6.1.MODULES

1. Data Collection and Preprocessing:
 Gather a diverse dataset of sign language gestures.
Preprocess the data by resizing, normalizing, and augmenting images or video frames.

2. Feature Extraction: The feature extraction module extracts meaningful information from the hand gesture, such as hand shape, position, and movement.

3. Machine Learning Classification: The extracted features are fed into a machine learning model, such as a convolutional neural network (CNN), for classification into corresponding sign language gestures.

4. Model Training:
Train the selected machine learning model on the training dataset.
Fine-tune hyperparameters and monitor training performance.

5. User Interface: The user interface module provides a user-friendly interface for interacting with the system, allowing users to input sign language gestures through a webcam or video feed.

Sample code:

```
import os

import cv2

cap=cv2.VideoCapture(0)

directory='Image/'

while True:

    _,frame=cap.read()
```

```
count = {
        'a': len(os.listdir(directory+"/A")),
        'b': len(os.listdir(directory+"/B")),
        'c': len(os.listdir(directory+"/C")),
        'd': len(os.listdir(directory+"/D")),
        'e': len(os.listdir(directory+"/E")),
        'f': len(os.listdir(directory+"/F")),
        'g': len(os.listdir(directory+"/G")),
        'h': len(os.listdir(directory+"/H")),
        'i': len(os.listdir(directory+"/I")),
        'j': len(os.listdir(directory+"/J")),
        'k': len(os.listdir(directory+"/K")),
        'l': len(os.listdir(directory+"/L")),
        'm': len(os.listdir(directory+"/M")),
        'n': len(os.listdir(directory+"/N")),
        'o': len(os.listdir(directory+"/O")),
        'p': len(os.listdir(directory+"/P")),
        'q': len(os.listdir(directory+"/Q")),
        'r': len(os.listdir(directory+"/R")),
        's': len(os.listdir(directory+"/S")),
        't': len(os.listdir(directory+"/T")),
        'u': len(os.listdir(directory+"/U")),
        'v': len(os.listdir(directory+"/V")),
        'w': len(os.listdir(directory+"/W")),
        'x': len(os.listdir(directory+"/X")),
        'y': len(os.listdir(directory+"/Y")),
            'z': len(os.listdir(directory+"/Z"))
            }
    # cv2.putText(frame, "a : "+str(count['a']), (10, 100), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "b : "+str(count['b']), (10, 110), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "c : "+str(count['c']), (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "d : "+str(count['d']), (10, 130), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "e : "+str(count['e']), (10, 140), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "f : "+str(count['f']), (10, 150), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "g : "+str(count['g']), (10, 160), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "h : "+str(count['h']), (10, 170), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "i : "+str(count['i']), (10, 180), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "k : "+str(count['k']), (10, 190), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "l : "+str(count['l']), (10, 200), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "m : "+str(count['m']), (10, 210), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255),
1)
    # cv2.putText(frame, "n : "+str(count['n']), (10, 220), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "o : "+str(count['o']), (10, 230), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "p : "+str(count['p']), (10, 240), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "q : "+str(count['q']), (10, 250), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "r : "+str(count['r']), (10, 260), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "s : "+str(count['s']), (10, 270), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "t : "+str(count['t']), (10, 280), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "u : "+str(count['u']), (10, 290), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "v : "+str(count['v']), (10, 300), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    # cv2.putText(frame, "w : "+str(count['w']), (10, 310), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255),
1)
```

```
# cv2.putText(frame, "x : "+str(count['x']), (10, 320), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
# cv2.putText(frame, "y : "+str(count['y']), (10, 330), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
# cv2.putText(frame, "z : "+str(count['z']), (10, 340), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
row = frame.shape[1]
col = frame.shape[0]
cv2.rectangle(frame,(0,40),(300,400),(255,255,255),2)
cv2.imshow("data",frame)
cv2.imshow("ROI",frame[40:400,0:300])
frame=frame[40:400,0:300]
interrupt = cv2.waitKey(10)
if interrupt & 0xFF == ord('a'):
    cv2.imwrite(directory+'A/'+str(count['a'])+'.png',frame)
if interrupt & 0xFF == ord('b'):
    cv2.imwrite(directory+'B/'+str(count['b'])+'.png',frame)
if interrupt & 0xFF == ord('c'):
    cv2.imwrite(directory+'C/'+str(count['c'])+'.png',frame)
if interrupt & 0xFF == ord('d'):
    cv2.imwrite(directory+'D/'+str(count['d'])+'.png',frame)
if interrupt & 0xFF == ord('e'):
    cv2.imwrite(directory+'E/'+str(count['e'])+'.png',frame)
if interrupt & 0xFF == ord('f'):
    cv2.imwrite(directory+'F/'+str(count['f'])+'.png',frame)
if interrupt & 0xFF == ord('g'):
    cv2.imwrite(directory+'G/'+str(count['g'])+'.png',frame)
if interrupt & 0xFF == ord('h'):
    cv2.imwrite(directory+'H/'+str(count['h'])+'.png',frame)
```

```
if interrupt & 0xFF == ord('i'):
    cv2.imwrite(directory+'I/'+str(count['i'])+'.png',frame)
if interrupt & 0xFF == ord('j'):
    cv2.imwrite(directory+'J/'+str(count['j'])+'.png',frame)
if interrupt & 0xFF == ord('k'):
    cv2.imwrite(directory+'K/'+str(count['k'])+'.png',frame)
if interrupt & 0xFF == ord('l'):
    cv2.imwrite(directory+'L/'+str(count['l'])+'.png',frame)
if interrupt & 0xFF == ord('m'):
    cv2.imwrite(directory+'M/'+str(count['m'])+'.png',frame)
if interrupt & 0xFF == ord('n'):
    cv2.imwrite(directory+'N/'+str(count['n'])+'.png',frame)
if interrupt & 0xFF == ord('o'):
    cv2.imwrite(directory+'O/'+str(count['o'])+'.png',frame)
if interrupt & 0xFF == ord('p'):
    cv2.imwrite(directory+'P/'+str(count['p'])+'.png',frame)
if interrupt & 0xFF == ord('q'):
    cv2.imwrite(directory+'Q/'+str(count['q'])+'.png',frame)
if interrupt & 0xFF == ord('r'):
    cv2.imwrite(directory+'R/'+str(count['r'])+'.png',frame)
if interrupt & 0xFF == ord('s'):
    cv2.imwrite(directory+'S/'+str(count['s'])+'.png',frame)
if interrupt & 0xFF == ord('t'):
    cv2.imwrite(directory+'T/'+str(count['t'])+'.png',frame)
```

6.2.Input and Output Design:

6.2.1.Input Design:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

6.2.2Output Design: A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1.Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

•Convey information about past activities, current status or projections of the Future.

•Signal important events, opportunities, problems, or warnings.

•Trigger an action.

•Confirm an action.

7.RESULTS:

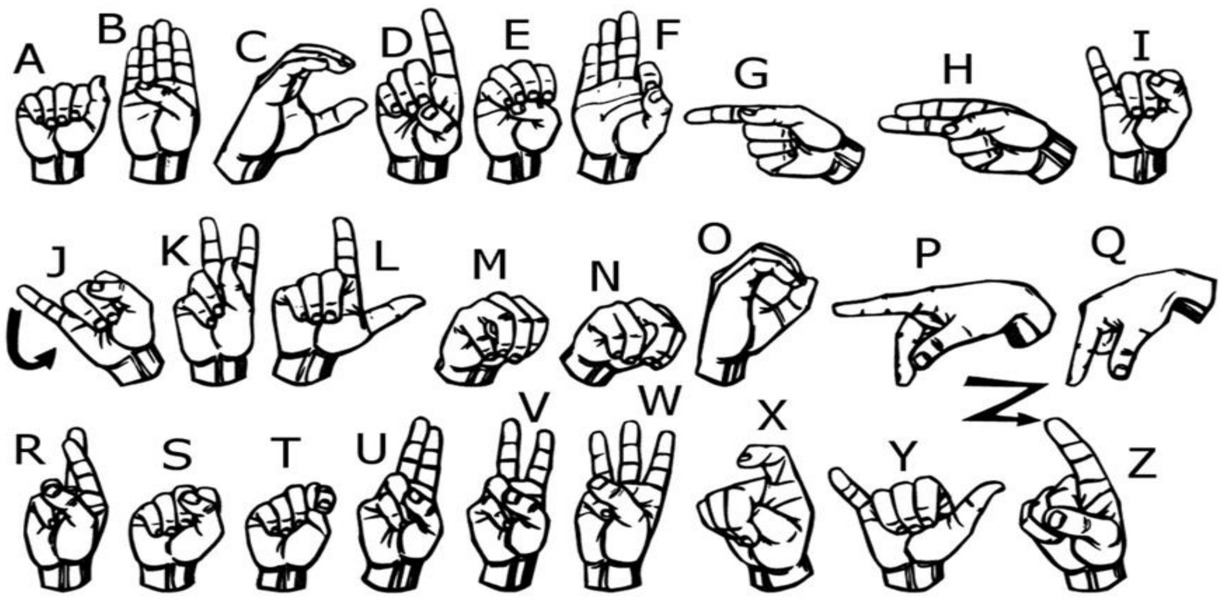

Fig .7.1 Specific Sign Of Each Alphabet

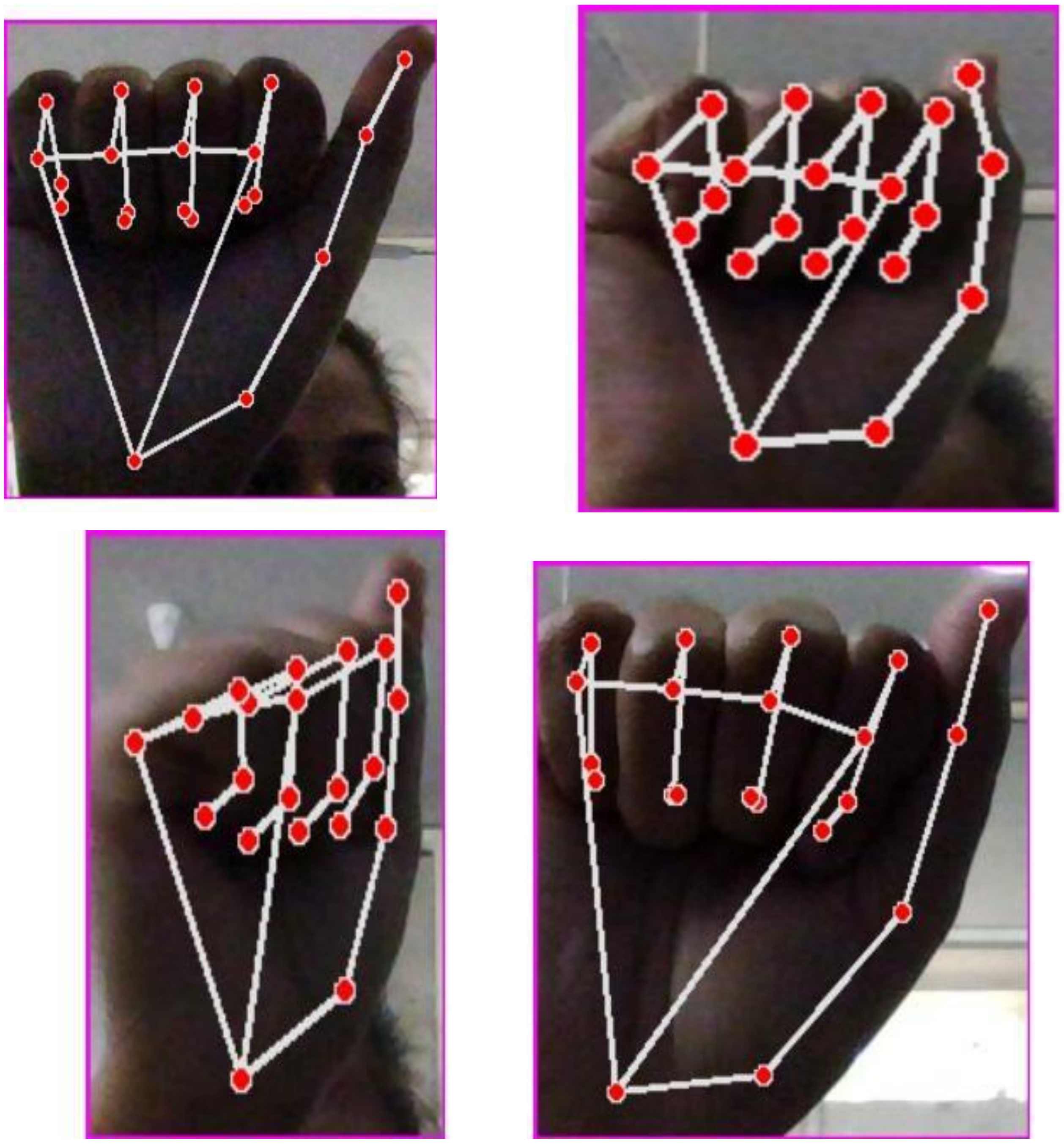

Fig 7.2 Sign Of Alphabet A

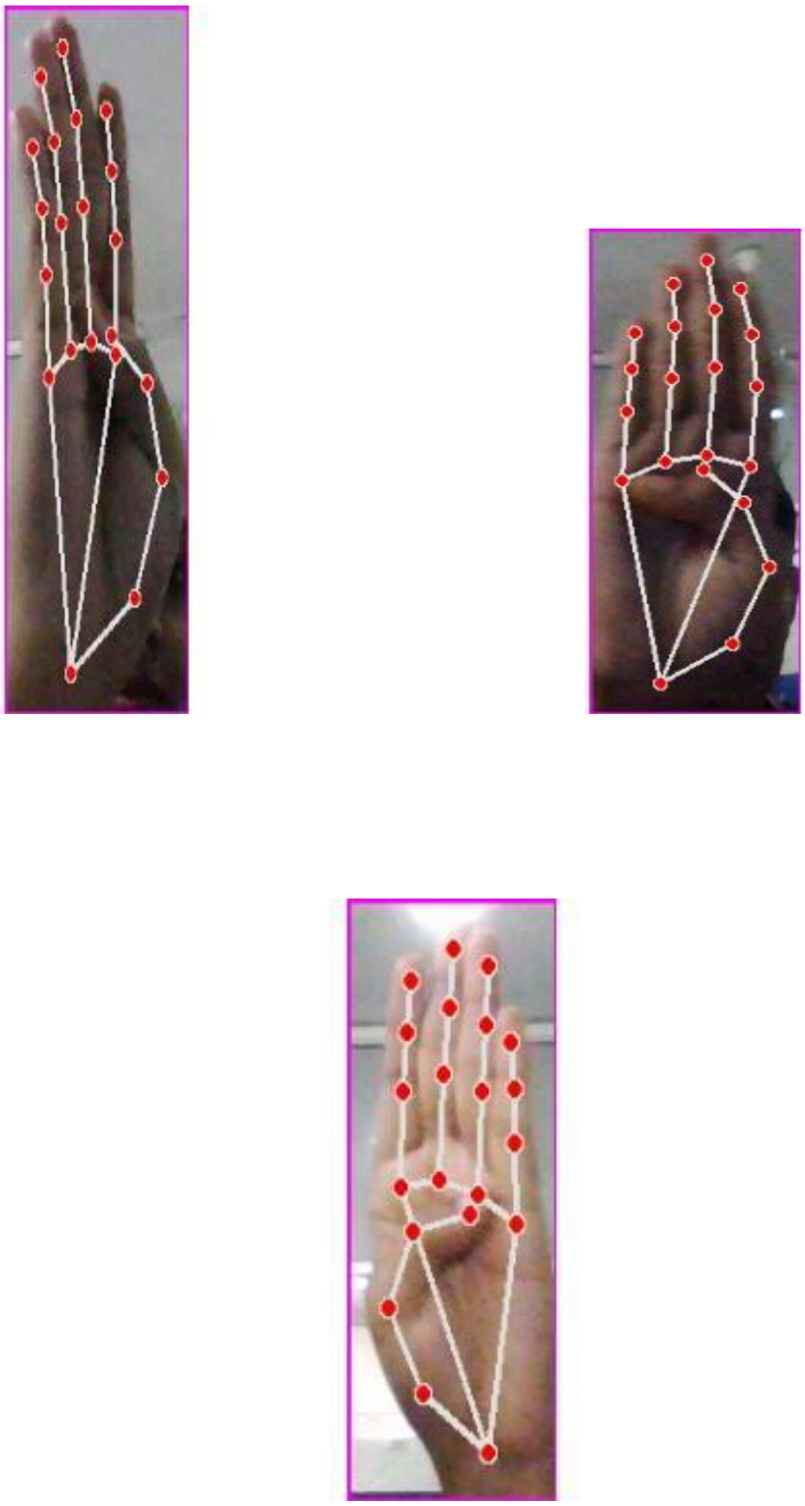

Fig 7.3 Sign Of Alphabet B

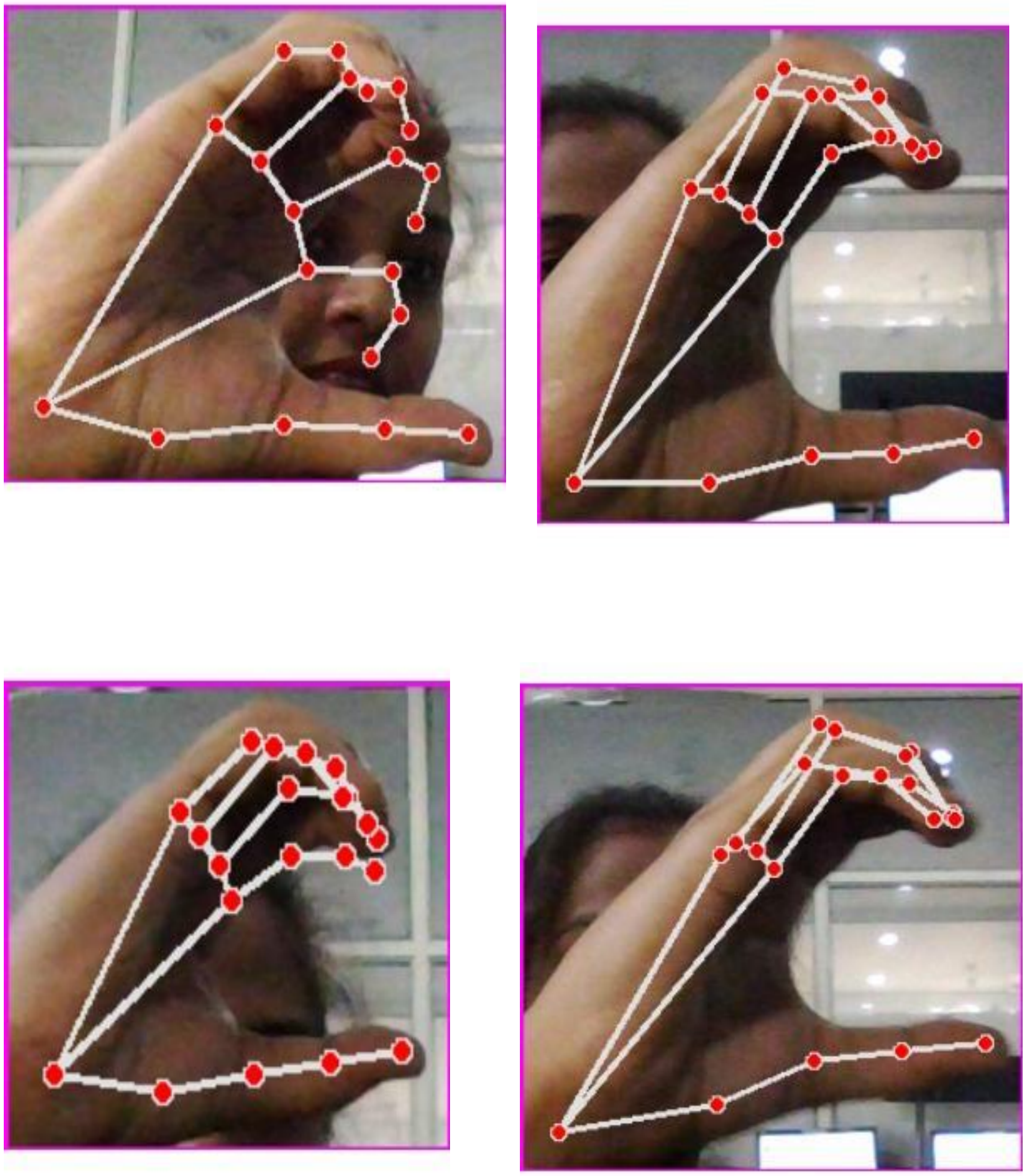
Fig 7.4 Sign Of Alphabet C

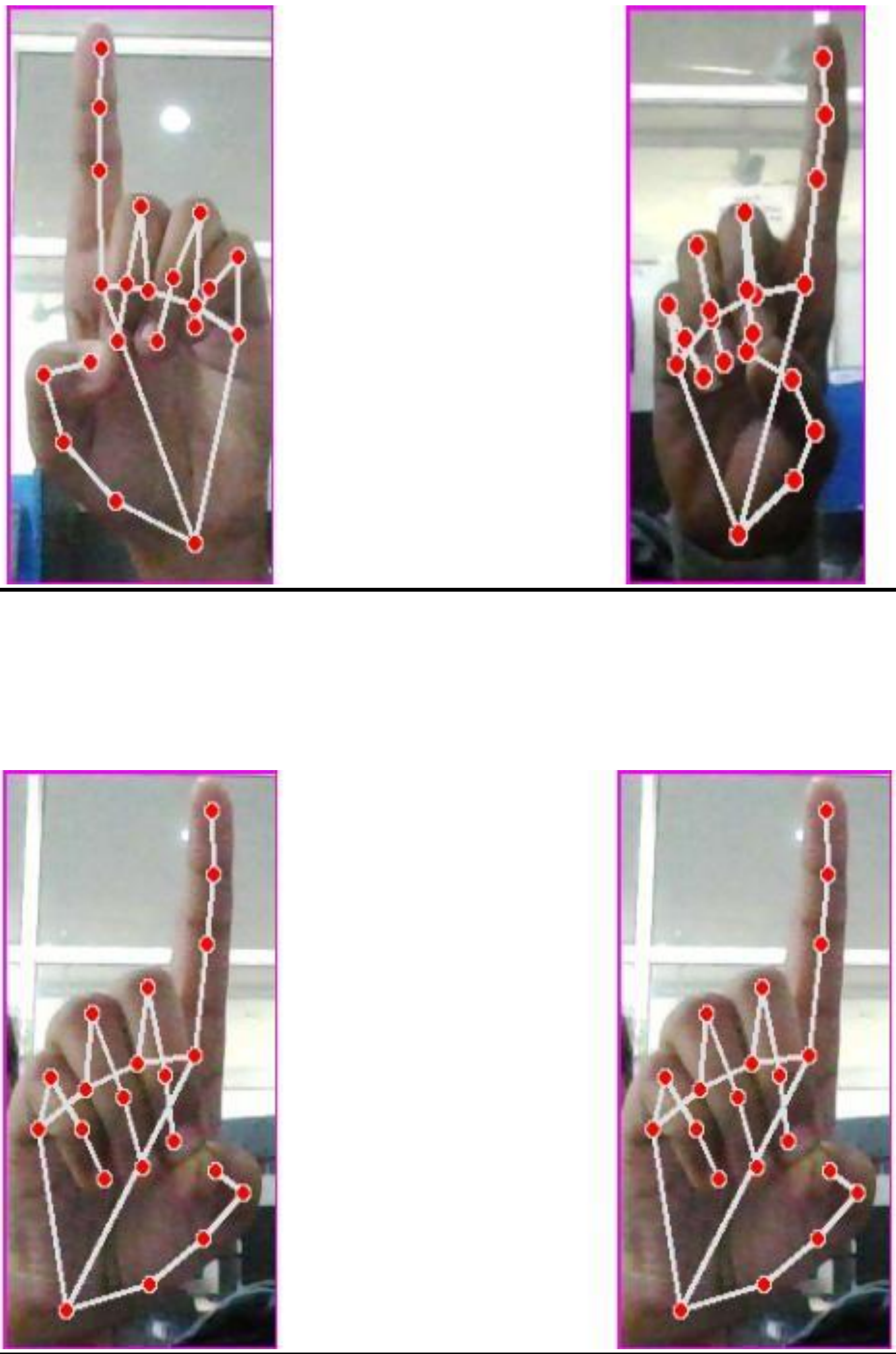

Fig 7.5 Sign Of Alphabet D

8.SAMPLE TEST CASES

8.8.1. Test Cases for Hand Detection and Tracking Module:

Test Case ID	Description	Expected Result	Result
TC_HDT_01	Verify hand detection in a well-lit environment.	Hands are detected.	Pass
TC_HDT_02	Test hand tracking as hands move across the frame.	Hands are tracked accurately as they move.	Pass
TC_HDT_03	Check hand detection in low-light conditions.	Hands are still detected, albeit with reduced accuracy.	Pass
TC_HDT_04	Verify the module's ability to handle occluded hands.	Hands behind objects are not detected.	Pass

Test Cases for Feature Extraction Module:

Test Case ID	Description	Expected Result	Result
TC_FE_01	Verify extraction of hand shape features.	Hand shape features are extracted correctly.	Pass
TC_FE_02	Test the extraction of hand position features.	Hand positions are accurately extracted.	Pass
TC_FE_03	Check if hand movement features are extracted.	Hand movement features, such as gestures, are correctly extracted.	Pass

Test cases of User interface

Test Case ID	Description	Expected Result	Result
TC_UI_01	Verify the availability of webcam input option.	Users can choose to input sign language gestures through a webcam.	Pass
TC_UI_02	Test the user interface's ease of use and navigation.	The user interface is intuitive and straightforward to use.	Pass
TC_UI_03	Check the responsiveness	The interface responds	Pass

Test Case ID	Description	Expected Result	Result
	of the user interface.	promptly to user interactions and inputs.	

Test Cases for Real-Time Interpretation Model

Test Case ID	Description	Expected Result	Result
TC_RTI_01	Verify accurate interpretation of basic signs.	Basic signs are interpreted correctly and converted to text/speech.	Pass
TC_RTI_02	Test interpretation accuracy for complex signs.	Complex signs are interpreted with reasonable accuracy.	Pass

Test Cases for Machine Learning Classification Module:

Test Case ID	Description	Expected Result	Result
TC_MLC_01	Verify correct classification of basic signs.	Basic signs are classified accurately.	Pass
TC_MLC_02	Test classification accuracy for complex signs.	Complex signs are classified with reasonable accuracy.	Pass
TC_MLC_03	Check if the module can handle rapid signing.	Signs produced at a rapid pace are classified with minimal errors.	Pass

9.CONCLUSION

The Sign Language Detection Using ML project demonstrates the potential of machine learning and image processing in bridging communication gaps for individuals with hearing impairments. By accurately recognizing and interpreting sign language gestures in real-time, the proposed system empowers the hearing-impaired community to communicate more effectively in various settings. The successful implementation of the project's modules showcases the feasibility of building an automated sign language detection system using machine learning techniques. This project represents a significant advancement in inclusive technology and has the potential to positively impact the lives of individuals with hearing impairments, fostering a more inclusive and accessible society.Gesture recognition is a complex and challenging task. We designed a software stack which could be easily used and extended. There is a dearth of Kinect depth datasets for Indian Sign Language signs (figure 18). ISL has hardly received any attention from the research community compared to ASL and other such sign language systems worldwide. We shall release all the data and code under an open-source license for the research community. In the fingerspelling category of our dataset, we achieved above 90% recognition rates for 13 signs and 100% recognition for three signs with overall 16 distinct alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) recognised with an average accuracy rate of 90.68%. Many approaches were tried out in this regard and would constitute a useful platform to engage in the future [52]. Future directions to further develop this system are as follows:

1. In the current work, we tried to tackle only static signs. But in majority of the communication through sign languages, much of the semantic content lies in movement, facial expressions and stress placed on the sign. Thus there is scope for development of a complete system which could be used as an accompaniment for the disabled to communicate, in which all elements of a sign language would be used.
2. Language and context models can be applied to generate complete sentences. This might be done using HMMs.
3. Gesture recognition could also be used in recognition of Cued Speech – a sign system in which both lipreading and hand gestures are used in conjunction.

4. This system could be ported to run on a portable embedded system that could be taken by a disabled person and used anywhere for interpretation.

10.FUTURE WORK:

At this moment, results showed high accuracy on the captured dataset with symbols for letters and numbers, but the CNN has been designed based on my own experience with some tests and tutorials I did before and following the architecture presented in . It may be that adding more symbol classes to recognize, the model precision decreases and thus the system will not be as good as before. My idea to improve system's design is to start with a group of gestures and test diverse configurations of the CNN to observe which configurations are the best for that case. The group of gestures will be increased with additional classes and, using the evaluation methods, identify the best configuration. From the relation number of classes and best configurations, my goal is to find a relation between input number of classes and design of the CNN so the system can construct an effective model for each case. Once the letter recognition system has achieved high recognition rate, a logical step towards the SL transcription is the concatenation of predicted letters to construct words. Since the CNN for letter recognition is not perfectly precise, errors exist at the recognition step, consequently, errors can accumulate over the transcription 70 process. My current knowledge about language processing is not high enough to define exact features of this future work but one idea is to use Hidden Markov Models for the creation of finger-spelled words and correct errors caused by the CNN. SL has symbols to represent words instead of finger-spelling them, therefore if a word gesture is detected the word creation step is skipped. Recognized words will then be used to construct sentences in the final step of the transcription process. SL has no direct translation to natural language due to the elusion of some of the words; as an example, stop words or prepositions. Techniques used in query suggestion systems may be useful to ensure correctness of sentence forming, but they have to be trained in already transcripted SL texts. The proposed system was trained only using static images; such a case excludes some of the letters in American Sign Language alphabet (J and Z). The inclusion of these letters and gestures done by movement is crucial for proper transcription, thus methods to recognize gestures will be required in the future. I mentioned the significance of face expressions. I would like to add this feature to the system because it may be helpful for understanding what the users are saying, and thus produce more precise

transcriptions. The use of another CNN should be useful to detect face expressions, that is one area I would like to study to improve the transcription system.
Some of other fields where Sign language recognition can be applied are:

Deep Learning Models:

- Explore and implement more advanced deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), or even transformer models like BERT for sign language recognition.
- Experiment with transfer learning using pre-trained models on large datasets to boost performance.

Real-time Detection:

- Optimize your model for real-time performance to make it practical for applications like live interpretation or interactive sign language learning tools.
- Consider deploying your model on edge devices or using frameworks like TensorFlow Lite or ONNX for efficient inference.

Gesture Recognition:

- Expand your system to recognize specific gestures within sign language, as different gestures can convey different meanings.
- Incorporate hand pose estimation to capture the fine details of hand movements.

Multimodal Approaches:

- Combine video and depth data for a more comprehensive understanding of sign language gestures.
- Integrate facial expressions and body language recognition to enhance the overall interpretation of signed messages.

Data Augmentation and Diversity:

- Augment your dataset with variations in lighting conditions, backgrounds, and hand orientations to improve the robustness of your model.
- Include a diverse set of signers to account for different signing styles and accents.

User Interface and Accessibility:

- Develop a user-friendly interface for your sign language detection system, making it accessible to a wide range of users.

- Consider integrating your system with existing communication tools or assistive technologies.

Continuous Learning:

- Implement techniques for continuous learning to adapt the model to new signs and variations over time.
- Explore online learning approaches to update the model with new data without retraining from scratch.

Localization and Cultural Adaptation:

- Extend your model to support different sign languages and regional variations.
- Consider cultural context and adapt your system to different signing styles and cultural nuances.

Evaluation and Metrics:

- Develop comprehensive evaluation metrics, considering factors such as accuracy, precision, recall, and F1 score. Include user studies for subjective evaluations.
- Conduct thorough benchmarking against existing sign language recognition systems.

Open Source and Collaboration:

- Consider open-sourcing your project to encourage collaboration and contributions from the community.
- Collaborate with organizations and communities working on accessibility and inclusion to gather insights and improve your system.

11.REFERENCES:

[1] Cheok Ming Jin, Zaid Omar, Mohamed Hisham Jaward, ”A Mobile Application of American Sign Language Translation via Image Processing Algorithms”, in IEEE Region 10 Symposium, on IEEEexplore, 2016.

[2] Mr. Sanket Kadam, Mr. Aakash GhodkeProf. Sumitra Sadhukhan, ”Hand Gesture Recognition Software based on ISL”, IEEE Xplore, 20 June 2019.

[3] Kartik Shenoy, Tejas Dastane, Varun Rao, Devendra Vyavaharkar, ” Real-time Indian Sign Language (ISL) Recognition”, IEEE Xplore: 18 October 2018.

[4] T Raghuveera, R Deepthi, R Mangalashri And R Akshaya, “A depth based ISL Recognition using Microsoft Kinect”, ScienceDirect, 2018.

[5] Muthu Mariappan H, Dr Gomathi V, “Real time Recognition of ISL”, IEEE Xplore: 10 October 2019.

[6] G. Ananth Rao a, P.V.V. Kishore, “Selfie video based continuous Indian sign language recognition system”, ScienceDirect, 2018.

[7] G. Ananth Rao a, P.V.V. Kishore, “Sign Language Recognition Based On Hand And Body Skeletal Data”, IEEE 2018.

[8] Suharjitoa, Ricky Andersonb, Fanny Wiryanab, Meita Chandra Ariestab, Gede Putra Kusuma, “Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process- Output”, ScienceDirect, 2017.

[9] Panwar.M. Hand Gesture Recognition System based on Shape parameters. In Proc. International Conference, Feb 2012

[10] Christopher Lee and Yangsheng Xu. Online, interactive learning of gestures for human robot interfaces. Carnegie Mellon

University, The Robotics Institute, Pittsburgh, Pennsylvania, USA, 1996.
[11] Hyeon-Kyu Lee and Jin H. Kim. An HMM-Based Threshold Model Approach for Gesture Recognition. IEEE transactions on
pattern analysis and machine intelligence, Volume 21, October 1999

[12] P. Subha Rajam and Dr. G. Balakrishnan. Real Time Indian Sign Language Recognition System to aid Deaf-dumb People. ICCT,
IEEE, 2011.

[13] Olena Lomakina. Development of Effective Gesture Recognition System. TCSET'2012, Lviv-Slavske, Ukraine, February 21-24,
2012.

[14] Etsuko Ueda, Yoshio Matsumoto, Masakazu Imai, Tsulasa Ogasawara. Hand Pose Estimation for Vision based Human Interface.
In Proc. 10th IEEE International Workshop on Robot and Human Communication (Roman 2001) , pp. 473-478, 2001

[15] Claudia Nölker and Helge Ritter. Visual Recognition of Hand Postures. In Proc. International Gesture Workshop on
Gesture-Based Communication in Human Computer Interaction, pp. 61-72, 1999