A Project Report on

Continuous Sign Language Recognition
Department of CSE

submitted in partial fulfillment for the award of

Bachelor of Technology

in

Computer Science & Engineering

by

V.Vineela (Y20ACS581) **T.Bhushaiah (Y20ACS576)**

R.Vamsi Lakshmi (Y20ACS550) **SK.Roshan Ali(Y20ACS563)**

Under the guidance of

Mr. K Ashok Babu, Asst. Prof.

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2023-2024

Department of
Computer Science & Engineering

CERTIFICATE

This is to certify that the project report entitled **Continuous Sign Language Recognition** that is being submitted by V.Vineela (Y20ACS581), T.Bhushaiah (Y20ACS576), R.Vamsi Lakshmi (Y20ACS550) and SK.Roshan Ali(Y20ACS563)) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide
K.Ashok Babu
Asst. Prof.

Signature of the HOD
M.Rajesh Babu
Assoc. Prof. & Head

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

V.Vineela (Y20ACS581)

T.Bhushaiah (Y20ACS576)

R.Vamsi Lakshmi (Y20ACS550)

SK.Roshan Ali(Y20ACS563)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr**.**K.Ashok Babu, Asst. Prof. ,** Department of CSE, for his valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Mr. M.Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr.Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

V.Vineela (Y20ACS581)
T.Bhushaiah (Y20ACS576)
R.Vamsi Lakshmi (Y20ACS550)
SK.Roshan Ali(Y20ACS563)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Communication is the ability that will influence how effectively human live their personal and professional life. It enables us to express ourselves. Speech is the most commonly used communication type even though we can communicate through speech, gestures, body language, reading, and writing or through visual aids. But the people with speaking and hearing minority, there is a communication gap between them and other normal people. Deaf-Dumb people can communicate with normal people with help of sign languages.

Sign Language is a means of communication which involves combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts. It consists of fingerspelling that means each character in a word is spelled out, word level signs, numbers and human relations etc., However it is very difficult for the deaf & dumb people to talk with the ordinary people. So it becomes impossible for them to communicate with the ordinary people unless and until ordinary people like us learn the sign language for the purpose of communication. The main objective of this project is to translate sign language into words.

The proposed system analyses and translate the sign language that is hand gestures into text by using deep learning and also by using OpenCV and Keras modules of python. We create a sign detector, which detects some signs that can very easily be extended to cover a vast multitude of other signs and hand gestures including the alphabets. This is divided into 3 parts: training a RESNET50 on the dataset ,Speech-text-sign and sign-text-speech.

1 INTRODUCTION

1.1 Introduction of Project Domain

One of the most important requirements for social survival is communication. Deaf and dumb peoples communicate with one another using sign language, but it is difficult for non-deaf and dumb people to understand them. Although sign languages have emerged naturally in deaf communities alongside or among spoken languages, they are unrelated to spoken languages and have different grammatical structures at their core.

Sign language is the natural language of the deaf and aphonic people. It is the basic method for the communication of deaf person. American Sign Language (ASL) is the language chosen by almost all the deaf communities of United States of America. Different Sign languages are evolved depending on the regions such as GSL (German Sign Language), CSL (Chinese Sign Language), Auslan (Australian Sign Language), ArSL (Arabic Sign Language), and many more.

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. It is a predominant sign language since the only disability D&M people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language. Communication is the process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. Deaf and dumb (D&M) people make use of their hands to express different gestures to express their ideas with other people. Gestures are the nonverbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exists in higher rates among the deaf population, especially when they are immersed in a hearing world. Large barriers that profoundly affect life quality stem from the communication disconnect between the deaf and the hearing. Some examples are information deprivation, limitation of social connections, and difficulty integrating in society.

Sign language is a visual language and consists of 3 major components

Table 1-1Sign Language Components

Fingerspelling	**Word level sign vocabulary**	**Non manual features**
Used to spell words letter by letter.	Used for the majority of communication.	Facial expressions and tongue, mouth and body position

1.1.1 Fingerspelling:

Fingerspelling is a method of spelling words using hand movements. The fingerspelling alphabet is used in sign language to spell out names of people and places for which there is not a sign. Fingerspelling can also be used to spell words for signs that the signer does not know the sign for, or to clarify a sign that is not known by the person reading the signer. Fingerspelling signs are often also incorporated into other signs. E.g. the sign for ‘gold’ is the fingerspelt ‘g’ and then moving your hands away in a shimmering motion (see the online dictionary for a demonstration). British Sign Language (BSL) uses a two-handed alphabet however some other sign languages, such as American Sign Language (ASL), use a one-handed alphabet.

1.1.2 Word Level Sign Vocabulary:

Sign languages vary in word-order typology. For example, Austrian Sign Language, Japanese Sign Language and Indo-Pakistani Sign Language are Subject-object-verb while ASL is Subject-verbobject. Influence from the surrounding spoken languages is not improbable. The entire gesture of words or alphabets is recognized through video classification. (Dynamic Input / Video Classification)

Sign languages tend to be incorporating classifier languages, where a classifier handshape representing the object is incorporated into those transitive verbs which allow such modification. For a similar group of intransitive verbs (especially motion verbs), it is the subject which is incorporated. Only in a very few sign languages (for instance Japanese Sign Language) are agents ever incorporated. In this way, since subjects of intransitives are treated similarly to objects of transitives, incorporation in sign languages can be said to follow an ergative pattern.

Brentari classifies sign languages as a whole group determined by the medium of communication (visual instead of auditory) as one group with the features monosyllabic and polymorphic. That means, that one syllable (i.e. one word, one sign) can express several morphemes, e.g., subject and object of a verb determine the direction of the verb's movement (inflection).

Another aspect of typology that has been studied in sign languages is their systems for cardinal numbers. Typologically significant differences have been found between sign languages.

1.1.3 Non-manual features:

Sign languages convey much of their prosody through non-manual elements. Postures or movements of the body, head, eyebrows, eyes, cheeks, and mouth are used in various combinations to show several categories of information, including lexical distinction, grammatical structure, adjectival or adverbial content, and discourse functions.

At the lexical level, signs can be lexically specified for non-manual elements in addition to the manual articulation. For instance, facial expressions may accompany verbs of emotion, as in the sign for angry in Czech Sign Language. Non-manual elements may also be lexically contrastive. For example, in ASL (American Sign Language), facial components distinguish some signs from other signs. An example is the sign translated as not yet, which requires that the tongue touch the lower lip and that the head rotate from side to side, in addition to the manual part of the sign.

Without these features the sign would be interpreted as late. Mouthing's, which are (parts of) spoken words accompanying lexical signs, can also be contrastive, as in the manually identical signs for doctor and battery in Sign Language of the Netherlands

While the content of a signed sentence is produced manually, many grammatical functions are produced non-manually (i.e., with the face and the torso). Such functions include questions, negation, relative clauses and topicalization. ASL and BSL use similar non-manual marking for yes/no questions, for example. They are shown through raised eyebrows and a forward head tilt.

Some adjectival and adverbial information is conveyed through non-manual elements, but what these elements are varies from language to language. For instance, in ASL a slightly open mouth with the tongue relaxed and visible in the corner of the mouth means 'carelessly', but a similar nonmanual in BSL means 'boring' or 'unpleasant'.

Discourse functions such as turn taking are largely regulated through head movement and eye gaze. Since the addressee in a signed conversation must be watching the signer, a signer can avoid letting the other person have a turn by not looking at them, or can indicate that the other person may have a turn by making eye contact.

1.1.4 Sign Language Detection

The process of converting the signs and gestures shown by the user into text is called sign language recognition. It bridges the communication gap between people who cannot speak and the general public. Image processing algorithms along with neural networks is used to map the gesture to appropriate text in the training data and hence raw images/videos are converted into respective text that can be read and understood.

Dumb people are usually deprived of normal communication with other people in the society. It has been observed that they find it really difficult at times to interact 4 with normal people with their gestures, as only a very few of those are recognized by most people. Since people with hearing impairment or deaf people cannot talk like normal people so they have to depend on some sort of visual communication in most of the time.

The importance of sign language is emphasized by the growing public approval and funds for international project. At this age of Technology the demand for a computer based system is highly demanding for the dumb community. However, researchers have been attacking the problem for quite some time now and the results are showing some promise. Interesting technologies are being developed for speech recognition but no real commercial product for sign recognition is actually there in the current market.

The idea is to make computers to understand human language and develop a user friendly human computer interfaces (HCI). Making a computer understand speech, facial expressions and human gestures are some steps towards it. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers.

1.2 Existing System

There are some existing systems of Sign Language detection by different image acquisition devices such as Data glove, Kinect, Leap Motion Controller and different segmentation techniques such as thresholding method, edge based method, clustering method and sign language classifiers such as k nearest neighbors, support vector machine, hidden markov model, fuzzy logic and ensemble learning.

1.2.1 Segmentation Technique

Thresholding Method

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images. Thresholding methods are categorized into six groups based on the information the algorithm manipulates, in this paper we focus on different clustering-based Thresholding methods. It is a fast and straightforward approach. It does not require prior information to operate. It has a low computation cost.

Disadvantage:

It is highly dependent on peaks, while spatial details are not considered. Sensitive to noise. Selection of an optimal threshold value is difficult.

Edge Based Method

Edge-based segmentation relies on edges found in an image using various edge detection operators. These edges mark image locations of discontinuity in gray levels, color, texture, etc. When we move from one region to another, the gray level may change. Suitable for images having better contrast between objects. It is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Detects and links edge pixels to form contours. Result from edge detection cannot be used directly.

Disadvantage:

It is not suitable for images with too much noise or too many edges.

Region-Based Method

The region-based segmentation method looks for similarities between adjacent pixels. That is, pixels that possess similar attributes are grouped into unique regions. It is less susceptible to noise and more useful when defining similarity criteria is easy. It has some predefined rules are present which have to be obeyed by a pixel in order to be classified into similar pixel

regions. Region-based segmentation methods are preferred over edge-based segmentation methods in case of a noisy image.

There are two types of segmentation: Top-down approach, Bottom-Up approach:

Disadvantage:

It is quite expensive in terms of computation time and memory consumption.

Clustering Method

The method of identifying similar groups of data in a dataset is called clustering. It is one of the most popular techniques in data science. Entities in each group are comparatively more similar to entities of that group than those of the other groups. It's more useful for real-world challenges due to the fuzzy partial membership employed. In other words, the clusters are regions where the density of similar data points is high. It is generally used for the analysis of the data set, to find insightful data among huge data sets and draw inferences from it. Generally, the clusters are seen in a spherical shape, but it is not necessary as the clusters can be of any shape.

It depends on the type of algorithm we use which decides how the clusters will be created. The inferences that need to be drawn from the data sets also depend upon the user as there is no criterion for good clustering.

Disadvantage:

Determining membership functions is not easy.

1.2.2 Sign language Classifier Methods

K-Nearest Neighbours

KNN is one of the classification methods. The KNN algorithm is used to determine which class a new observation to be included in the sample from the observation values in a sample set with certain classes. In this method the similarity of test data is to be classified with the education data. Classification is done according to the threshold value determined with average of the k data that appears to be the closest. The performance of this process is depends on the closest neighbour number, threshold value, similarity measurement and sufficient number of normal behaviours. It was trained with MNIST data and then tested. While creating the KNN classifier with Scikit Learn, the number of neighbour was set to 5. And all points in each neighbourhood are weighted equally as

'weight' parameter was set to uniform. MNIST data set is used here. It is effortless to implement. It is a simple algorithm to interpret.

This algorithm is one of the most preferred unsupervised learning algorithms. Clusters are created by looking at the similarity rates of the data. The number of clusters to be created in the algorithm is already determined in advance. The number of the clusters is expressed by "k". Among the algorithms compared within the scope of the study, the only algorithm that is an unsupervised learning algorithm is the K-means algorithm. While creating K-Means clustering with Scikit

Learn, the number of clusters was set to 10. In the tests performed with the Scikit Learn on the MNIST data set, it was observed that the accuracy of the clusters was high.

Disadvantage:

Very sensitive to irrelevant features. It does not work well with a large dataset. It does not work well with high dimensions.

Support Vector Machine (SVM)

Support Vector Machine, is a supervised machine learning used for the classification method and the regression challenge. However, it is mainly used for the classification methods. The main aim of the SVM algorithm is to form the best line or a decision boundary which can be able to segregate n-dimensional space into the classes so that it can be easy to put the new data in the right category further. SVM will choose the extreme point or the extreme vectors which helps in creating a hyper plane. These extremes are called support vectors and hence this algorithm is called Support Vector Machine (SVM). SVM is of two types Linear SVM and Non-linear SVM.

Linear SVM is used only for the linearly separable data that means if a dataset is classified into two classes by using the single straight line, then that data is termed as the linearly separable data and the classifier is called as the Linear SVM Classifier. Non-linear SVM is used only for the nonlinearly separable data that means if a dataset is could not be classified using a single straight line then such data is called as non-linearly separable data and the classifier is known as Non-linear SVM Classifier.

The algorithm begins with the previously created training set. During the training, SVM learns the relationship of each data in the existing training set. Kernel function selection is an important step in SVM to solve any type of problem. Polynomial kernel function was used in the tests performed on the MNIST data set. It performs better when dealing with multi dimensions and

continuous features. It is applicable in numerous domains. Tolerance to irrelevant attributes.

Disadvantage:

It requires a large sample of the dataset to achieve its maximum prediction accuracy. Hyper parameters are often challenging while interpreting their impact.

Fuzzy logic

Fuzzy Logic (FL) is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO. The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO. It is an expert base technique that provides solutions to complex solutions.

It deals with complicated problems in a simple way.

Disadvantage:

It is completely dependent on human intelligence and expertise. It has low accuracy, and its predictors are not always correct.

Ensemble Learning

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.)

performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting. This article focuses on classification related applications of ensemble learning, however, all principle ideas described below can be easily generalized to function approximation or prediction type problems as well. It improves the average prediction performance. It provides high accuracy and a more stable model. It reduces the variance of predictive errors.

Disadvantage:

It can be more difficult to interpret. Sometimes the model can be over fitted or under fit using the ensemble learning method

1.3 Objectives of the Project

The main objective of the project is to develop a system that can translate sign language into its corresponding word equivalent that includes letters, numbers and sentences to familiarize the users with the fundamentals of sign language.

Sign Language is a means of communication which involves combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts. The system analyses and translate the sign language that is hand gestures into text by using deep learning and also by using OpenCV and Keras modules of python. We will create a sign detector, which detects signs (hand gestures) and convert them into words and sentences.

- Producing a model which can recognize Fingerspelling-based hand gestures in order to form a complete word

- Producing a model which can recognize Fingerspelling-based hand gestures in order to form a complete sentence
- To train the model with the manual dataset
- To recognize signs of sign language by using hand gestures.
- To store and converts alphabetic signs into words.
- To recognize the space between the words by the blank screen.
- To convert words into sentences with the help of blank screen (space between words)
- To make sign language detection easier
- To convert words into corresponding signs

1.4 Organization of Project

1.4.1 Introduction

The first chapter gives the introduction to the project area and the domain that is being used in the project.

1.4.2 Requirement Analysis

The second chapter focuses on the requirements that are needed by the system in order to accomplish the aim of the project.

1.4.3 Literature Survey

The third chapter focuses on literature review which aims at a survey of different issues related to the existing topics along with that it also acts as basics in achieving current knowledge.

1.4.4 Modules

The fourth chapter highlights the modules in which the project is being divided thus making the implementation easier in order to accomplish the aim of the project.

1.4.5 System Design

The fifth chapter highlights the design issues related to the system along with the actual design of various modules in the form of UML diagrams.

1.4.6 System Implementation

The sixth chapter focuses on the implementation of modules which are described in the Chapter-4 along with the technology used to accomplish the aim of the project.

1.4.7 Testing

The seventh chapter highlights about testing and the need of testing along with various kinds of testing. The test cases related to each of the input design issues are written.

1.4.8 Screens & Reports

The eighth chapter focuses on the results and the reports obtained from the project**.** It contains screenshots and outputs of the project.

1.4.9 Conclusion and Future Scope

The ninth chapter highlights conclusion from the project along with the scope of the project in future.

1.4.10 References

This chapter represents the references and books that are used in the project.

2 REQUIREMENTS ANALYSIS

2.1 Introduction

Requirement Analysis which is also known as Requirement Engineering, is the process of defining the user expectations for a new software which is in the process of building or modified. In the software engineering, sometimes it is referred to loosely by the names such as the requirements gathering or the requirements capturing. Requirements analysis encompasses those tasks that go into determining the needs or the conditions to meet for a new or an altered product or project, taking in account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing system or software requirements.

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. It involves all the tasks which are conducted to identify the needs of the different stakeholders. Therefore the requirements analysis means it is to analyze, make document, validate and manage the software or the system requirements. High-quality requirements are actionable, documented, traceable, testable, measurable, helps to identify the business opportunities, and are defined to facilitate the system design.

A software requirement is the capability needed by the user to solve any problem or to achieve an objective. In other words, the requirement is the software capability that must be met or possessed by a system or the system component to satisfy a specification, standard, contract, or other formally imposed documentation.

Ultimately, what we want to achieve is to develop the high quality software that meets the customers' real needs on a certain time and within the budget. Perhaps the greatest challenge or hurdle that is being faced by the software developers is to share the vision of the final product with the customer.

All stakeholders in a project such as developers, end users, software managers, customer managers must achieve some common understanding of what the product will be like and what we do, or someone will be surprised when it is delivered. Surprises in software are not always a good news. Therefore, we need some ways to accurately capture, interpret, and then represent the voice of the customers when specifying the requirements for a software product.

Requirement analysis will encompasses those tasks which are meant to go into determining the needs or conditions to achieve a new or an altered product or a project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements. Here are the objectives for performing requirement analysis in the early stage of a software project.

The main aim of requirement analysis is to fully understand the main objective of the requirement that includes why it is necessary, Does it add the value to the product?

- Will it be the beneficial?
- Does it increase the quality of the project?
- Does it will have any other effect?

All these points are to be fully recognized in the problem recognition so that the requirements which are essential can be fulfilled to solve the business problems.

Evaluation means judging about something whether it is worth or not and the synthesis means to create or form something. Here are some of the tasks that are given that is important in the evaluation and the synthesis of the software requirement:

- To define all the functions of the software that are necessary.
- To define all the data objects that are present externally and are easily observable.
- To evaluate that the flow of data is worth or not.
- To fully understand overall behavior of the system that means overall working of the system.
- To identify and discover the constraints that are meant to be designed.
- To define and establish the character of system interface to fully understand how the system interacts with two or more components or with one another.

After complete the gathering of information from the above tasks, the functional and the behavioral models are established after checking the function and the behavior of the system using a domain model that also known as the conceptual model. The software requirement specification (SRS) which means to specify the requirement whether it is functional or nonfunctional should be developed. After developing the SRS document, it must be reviewed to check whether it can be improved or not and must be refined to make it better and increase the quality.

2.1.1 Activities for Requirement Analysis

Requirements analysis is critical to the success or the failure of a systems or the software project. The requirements should be actionable, documented, testable, measurable, traceable, related to identified business needs or opportunities, and

defined to a level of detail sufficient for system design. Conceptually, requirements analysis includes four types of activity:

1. **Eliciting requirements**: The task of communicating with the customers and the users to determine what their requirements are. This is sometimes also called as requirements gathering.
2. **Analyzing requirements**: Determining whether the stated requirements are incomplete, contradictory, unclear or ambiguous and then resolving these issues.
3. **Requirements modeling**: the requirements might be documented in various forms, such as process specifications, use cases, user stories or natural languages.
4. **Review and retrospective**: Team members need to reflect on what happened in the iteration and identifies actions for the improvement of going forward.

Requirements analysis is a team of effort that demands the combination of hardware, software and human factors engineering expertise as well as skills in dealing with the people. Here are the main activities involve in requirement analysis:

- Identify the customer's needs.
- Evaluate the system for feasibility.
- Perform the economic and the technical analysis.
- Allocate the functions to the system elements.
- Establish the schedule and the constraints.
- Create the system definitions.

2.1.2 Business Requirement vs Software Requirements

A business plan or project requires a variety of requirements to help for defining the goals and to establish the scope for the work that is to be undertaken. Requirements also provide the context and the objective ways to measure the progress and the success. Once the business requirements are established, then the software requirements are defined and are developed in order to move the project forward.

Business Requirements

Business requirements relate to a business' objectives, vision and goals. They also provide the scope of a business need or problem that needs to be addressed through a specific activity or project. For example, if a trade association has an objective to promote the services offered by its members, the business requirements for a project might include creating a member directory that increases awareness of members. Good business requirements must be:

- Clear and are typically defined at a very high level.
- Provide enough information and guidance to help ensure that the project fulfils the identified need.
- Understanding an organization's mandate, objectives or goals, a specific business need or problem that is being tackled.
- Should be clearly defined and understood before developing business requirements.
- The need or problem can relate to the organization or business in general or focus on a stakeholder group, such as customers, clients, suppliers, employees or another group.

2.1.3 Software Requirements

Software requirements break-down the steps needed to meet the business requirement or requirements. Whereas a business requirement states the 'why' for a project, a software requirements outline the 'what'. For example, if the business requirement is to create a member directory for a trade association, the software requirements will outline who has access to the directory, how member register with the directory, who will have ownership of the data, what vehicle or vehicle will be used such as a website or paper-based directory, and so on.

2.1.4 Hardware Requirements

The hardware requirements are the requirements of a hardware device. Most of the hardware only has the operating system requirements or its compatibility. For example, a printer may be compatible with Windows XP operating system but not compatible with the newer versions of the Windows like Windows 10, Linux, or the Apple macOS.

2.2 Hardware & Software Requirements

Architecture

All computer operating systems are designed for some particular computer architecture. Most of the software applications are only limited to the particular operating systems running on some particular architectures. Although the architecture independent operating systems and the applications exist are mostly need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

Processing power

The power of the central processing unit (CPU) is the fundamental system requirement for any software. Most of the software running on x86 architecture define the processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence is speed and power, like bus speed, cache, and MIPS are often ignored. This definition of the power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

Memory

All the software, when they run, will resides in the random access memory (RAM) of the computer. Memory requirements are defined after considering some demands of the application, operating system, supporting software and files, and some other running processes.

Secondary storage

Hard-disk requirements may vary, depending on the size of the software installation, temporary files are created and are maintained while installing or running the software, and the possible use of swap space (if RAM is insufficient).

Software Specifications

Software requirements deals with the defining of software resource requirements and the prerequisites that are needed to be installed on a computer to provide the maximum and optimal functioning of any application. These requirements or the prerequisites are generally

will not be included in the software installation package and is need to be installed separately before the software is installed.

Platform

In the computing system, a platform will describe some sort of the frameworks, either in a hardware or a software, which allows the software to run. Some of the typical platforms include the computer's architecture, the operating system, or the programming languages and their runtime libraries.

Operating system

Operating systems are one of the first crucial requirements mentioned and used when defining the system requirements i.e., software. Software may so not be compatible with various versions of the same line of the operating systems, although some of the measures of the backward compatibility is neatly maintained. For instance, most of the software is designed for the Microsoft Windows XP and it does not run on the Microsoft Windows 98, although the converse is not may be always true. Similarly, the software which is designed is using newer features of Linux Kernel v2.6 generally will not run or compile properly or sometimes not at all on the Linux distributions when using Kernel v2.2 or v2.4.

APIs and drivers

Software is making extensive use of the special hardware devices, such as high-end display adapters, it needs a special API or a new device drivers. One of the good examples is DirectX, which is one of the collections of the APIs for

handling the tasks related to the multimedia, especially of game programming, on Microsoft platforms.

Web browser

Most of the web applications and the software depending is heavily on the Internet technologies which make use of the default browser installed on the system. Microsoft Internet Explorer is the frequent choice of the software running on the Microsoft Windows, which makes use of the ActiveX controls, in spite of their vulnerabilities.

Software requirements

- Python 3.6.6
- Tensor flow 1.11.0
- OpenCV 3.4.3.18
- NumPy 1.15.3
- Matplotlib 3.0.0
- Hunspell 2.0.2
- Keras 2.2.1
- PIL 5.3.0

Hardware requirements

- GB RAM (8 GB RAM Recommended)
- I3 PROCESSOR (I5 PROCESSOR Recommended)

2.3 Software Requirements Specification

Software Requirement Specification (SRS) as the name suggests, it is complete specification and the description of the requirements of the software

that it needs to be fulfilled for the successful development of the software system. These mentioned requirements can be of functional and also as non-functional depending upon the type of the requirement. The interaction between the several customers and the contractor is done because it is necessary to completely understand the needs of stakeholders or customers.

Depending upon the information gathered after the interaction with the customers, SRS is developed which will describe the requirements of software which may include changes and modifications that are needed to be done to increase the quality of product and to satisfy the customer's demand.

The structure of SRS is as follows:

2.3.1 Introduction

This document is a software requirement specification of Sign Language Detection Using RESNET50. Through this document, we are going to provide all specifications and functionalities of the project. This document will mention the functionality, that is what the resulting application is supposed to do, external interfaces which interacts the users, performance, attributes, that is if the application is portable, or maintainable, and design constraints imposed on the implementation such as implementation language, input specifications and output expectations.

Purpose of this document

The aim of this document is to specify the features, requirements of the final product and the interface of Sign Language Detection Using RESNET50. It will explain the scenario of the desired project and necessary steps in order to succeed in the task. To do this throughout the document, overall description of the project,

the definition of the problem that this project presents a solution and definitions and abbreviations that are relevant to the project will be provided. The preparation of this SRS will help consider all of the requirements before design begins, and reduce later redesign, recoding, and retesting. If there will be any change in the functional requirements or design constraints part, these changes will be stated by giving reference to this SRS in the following documents.

Scope of this document

The scope of this project is as follows:

- To recognize signs of sign language by using hand gestures.
- To store and converts alphabetic signs into words.
- To recognize the space between the words by the blank screen.
- To convert words into sentences with the help of blank screen (space between words)

2.3.2 General description

The general description of sign language detection model is the user make a sign in front of the camera so we will train and detect the sign that user made one by one character which form the word and by using blank we can convert into sentence. After that if we want to detect another sign again we will execute the code which opens web camera so that user can make the sign and system will detect the sign.

2.3.3 Functional requirements

The functional requirements here are some processing steps before getting the possible outcomes, they are explained below briefly.

Functional Req. ID #	Functional Requirement Name	Functional Requirement Description
1	Image acquisition	Image acquisition is the first stage in sign language recognition that can be acquired through self-created or available public datasets.
2	preprocessing	The second stage is preprocessing to eliminate unwanted noise and enhanced the quality of the image.
3	segmentation	after preprocessing step is to segment and extract the region of interest from the entire image
4	Feature extraction	The fourth stage is feature extraction, which transforms the input image region into feature vectors for recognition
5	Classification	The last involves matching the features of the new sign image with the stored features in the database for recognition of the given sign.

Interface Requirements

In the interface requirements the software program/code is interlinked to one another. One code file will helps to train the model, the second code file is used to test the model and the final code file is used to create the dataset. By making use of all these three it can detect, recognize and display the output.

2.3.4 Performance Requirements

The performance is measured in the terms of classification accuracy, sensitivity to training the sample size of data, ambiguity and useless rejection, and the outlier resistance. The outlier resistance of neural classifiers is enhanced by the training with the synthesized outlier data.

2.3.5 Design Constraints

The hardware and software limitations for this model are processor is of any process above 500MHz, RAM should be 4GB and more, hard disk is about 4GB and more, input devices are web camera and mouse, output devices are monitor with high resolution. Operating system must be Windows 7 and high, programming languages used is python 3.6 and libraries related to the algorithm, platforms are visual studio and anaconda prompt.

2.3.6 Appendices

The information for this SRS is collected from some of the references and mentioned below:

https://www.sciencedirect.com/science/article/pii/S2667305321000454#:~:text=The%20stages% 20involved%20in%20vision,5.

https://air.imag.fr/index.php/SRS_-_Sign2Speech

https://senior.ceng.metu.edu.tr/2012/dbug/documents/SRS.pdf

3 LITERATURE SURVEY

In Literature survey we have gone through other similar works that are implemented in the domain of sign language recognition. The summaries of each of the project works are mentioned below

Pavlovic et al. [1] discussed about the visual interpretation of hand gestures for Human-Computer Interaction. The paper published on 1997 emphasizes on the advantages and shortcomings and important differences in the gesture interpretation approaches depending on whether a 3D model of the human hand or an image appearance model of the human hand is used. As of the time, this survey was done 3D hand models offered a way of more elaborate modeling of hand gestures but lead to computational hurdles that had not been overcome given the real-time requirements of HCI. They also discussed implemented gestural systems as well as other potential applications of visionbased gesture recognition.

Jiyong et al. [2] presented as input to the Hidden Markov Models (HMMs) for a real-time system designed to recognize continuous Chinese Sign Language (CSL). Raw Data was collected from two Cyber-Gloves and a 3-D tracker. Dynamic programming (DP) technique was used to segment the training sentence into basic units, then; estimating was done by the Welch-Baum algorithm. Test results using 220 words and 80 sentences, and the system showed 94.7% recognition rates.

Volger et al. [3] used Parallel Hidden Markov models (PaHMMs) for American Sign Language recognition. They stated that phonemes could be used instead of

whole signs for a continuous recognition system. Used Two channels to the right and left hands, assuming any word can be broken down into fundamental phonemes the same as words in speech recognition. A single channel of the HMM model was tested for a small vocabulary number (22 signs) with results showing an 87.88% accuracy rate. The system cannot recognize a larger vocabulary, hand configuration, orientation, and facial expressions.

Gunasekaran et al. [4] in their paper discussed about the recognition of sign language and translation into speech using a microcontroller system. The proposed system played a pre-recorded voice every time a sign language gesture was detected. The proposed model consisted of four modules, they are sensing unit, processing unit, voice storage unit, and a wireless communication unit. It was achieved by integrating the flux sensor and APR9600 with PIC16F877A. The flux sensors are placed in gloves, which respond to the gesture. By using a suitable circuit response of the sensor was given to the microcontroller based on the response microcontroller played the recorded voice using APR9600. This system offered high reliability and fast response.

The paper by Kalidolda et al. [5] described the project, which aimed to develop an interpreting robotic system of sign language. The project had two core objectives of facilitating fingerspelling recognition in real-time and conducting performance testing „in the wild“ and to get feedback on the NAO robotic interpreting system from deaf-mute individuals. The robotic system comprises a number of software and hardware components, particularly: Microsoft Kinect SDK for human detection and tracking, Leap Motion SDK for hands detection and tracking. A humanoid stationary NAO robot (NaoQi SDK) acting as a social interface. NAO also takes the role of a robotic tutor for children, A humanoid

stationary Pepper robot (Pepper SDK for An-droid) acting as a social interface, A computer monitor for virtual robot and Android tablet (Android SDK) is used for tutoring applications.

Tanuj Bohra et al. [6] proposed a real-time two-way sign language communication system built using image processing, deep learning and computer vision. Techniques such as hand detection, skin color segmentation, median blur and contour detection are performed on images in the dataset for better results. RESNET50 model trained with a large dataset for 40 classes and was able to predict 17600 test images in 14 seconds with an accuracy of 99%.

Joyeeta Singha and Karen Das [7] proposed a system for Indian sign language recognition from a live video. The system comprises of three stages. Preprocessing stage includes skin filtering and histogram matching. Eigen values and Eigen vectors are being considered for feature extraction stage and Eigen value weighted Euclidean distance for classification. Dataset consisted 480 images of 24 signs of ISL signed by 20 people. System was tested on 20 videos and achieved an accuracy of 96.25%.

Muthu Mariappan H. and Dr. Gomathi V. [8] designed a real time sign language recognition system as a portable unit using contour detection and fuzzy c-means algorithm. Contours are used for detecting face, left and right hand. While fuzzy c-means algorithm is used to partition the input data into specified number of clusters. System was implemented on a dataset that contained videos recorded from 10 signers for several words and sentences. It was able to achieve accuracy of 75%.

Salma Hayani et al. [9] proposed an Arab sign language recognition system based on RESNET50, inspired from LeNet-5. Dataset contained 7869 images of Arab signs of numbers and letters. Various experiments were conducted by varying the number of training sets from 50% to 80%. 90% accuracy was obtained with 80% training dataset. The author has also compared the results obtained with machine learning algorithms like KNN (k-nearest neighbor) and SVM (support vector machine) to show performance of the system. This model was purely image based and it can be extended to video based recognition.

Kshitij Bantupalli and Ying Xie [10] worked on American Sign Language recognition system which works on video sequences based on RESNET50, LSTM and RNN. A RESNET50 model named Inception was used to extract spatial features from frames, LSTM for longer time dependencies and RNN to extract temporal features. Various experiments were conducted with varying sample sizes and dataset consists of 100 different signs performed by 5 signers and maximum accuracy of 93% was obtained. Sequence is then fed to a LSTM for longer time dependencies. Outputs of softmax layer and maxpooling layer are fed to RNN architecture to extract temporal features from softmax layer.

Mahesh Kumar [11] proposed a system which can recognize 26 hand gestures of Indian sign language based on Linear Discriminant Analysis (LDA). Preprocessing steps such as skin segmentation and morphological operations are applied on the dataset. Skin segmentation was carried out using Otsu algorithm. Linear discriminant analysis is used for feature extraction. Each gesture is represented as a column vector in training phase which is then normalized with respect to average gesture. The algorithm finds the eigenvectors of the covariance matrix of normalized gestures. In recognition phase, subject vector is

normalized with respect to average gesture and then projected onto gesture space using eigenvector matrix. Euclidean distance is computed between this projection and all known projections. The minimum value of these comparisons is selected.

Suharjito et al. [12] tried to implement a sign language recognition system with I3D inception model through transfer learning method. Public dataset LSA64 [15] was used for 10 vocabularies with 500 videos. For training the dataset is distributed to 6:2:2 ratio, 300 videos for training, 100 for validation and 100 for testing set. The model has good training accuracy but very low validation accuracy.

Oscar Kellar et al. [13] introduced a hybrid RESNET50-HMM for sign language recognition. They conducted experiments on three datasets namely RWTH-PHOENIX-Weather 2012 [19], RWTHPHOENIX-Weather Multisigner 2014 and SIGNUM single signer. Training and validation set have a ratio of 10 to 1. After the RESNET50 training is finished a softmax layer is added and results are used in HMM as observation probabilities.

Mengyi Xie and Xin Ma [14] proposed an end-to-end system using residual neural network to implement recognition of American Sign Language. Dataset contained 2524 images for 36 classes. Data enhancement is used to expand the dataset to 17640 images. These images are converted to CSV file format and after applying one-hot encoding are given as input to ResNet50 network for training. Model gives an accuracy of 96.02% without data enhancement and accuracy improves with data enhancement to 99.4%.

G. Anantha Rao et al. [15] proposes an Indian sign language gesture recognition using convolutional neural network. This system works on videos captured from a mobile's front camera. Dataset is created manually for 200 ISL signs. RESNET50 training is performed with 3 different datasets. In the first batch, dataset of only one set is given as input. Second batch has 2 sets of training data and third batch respectively has 3 sets of training data. Average recognition rate of this RESNET50 model is 92.88%.

Aditya Das et al. [16] trained a convolutional neural network using Inception v3 model for American Sign Language. Data augmentation is applied on the images before training them to avoid overfitting. This model gives more than 90% accuracy on Sreehari sreejith dataset for 24 class labels with 100 images per class.

The literature surveys can be easily understood by the following table in which it consists of the reference paper title, their purpose and their drawbacks.

Table 3-1 Literature Survey Table

S.No	Title (Method Used)	Drawbacks
1	Hand Gesture Recognition System For the Dumb People (SIFT Algorithm)	Mathematically complicated and computationally heavy
2	An Automated System for Indian Sign Language Recognition (Otsu's thresholding algorithm)	It doesn't perform very well when the dataset has more noise

3	Sign Language Interpreter using a Smart Glove (Portable Smart Glove)	High Latency, Expensive, High Propagation
4	Hand Sign Language Recognition for Bangla Alphabet using Support Vector Machine (Support Vector Machine)	It is not suitable for large data sets

4 MODULES

4.1 Introduction

A module is a collection of source files and build settings that allow you to divide project into discrete units of functionality. Project can have one or many modules, and one module may use another module as a dependency. You can independently build, test, and debug each module. In general, the software comprises of many systems and each system is further divides into further sub-systems and each sub-system is again divided into further sub-systems and this goes on and so it is not possible to design the entire systems in a single flow. In order to solve this kind of problem, we need to breakdown the entire software into multiple parts. These parts are known as “Modules”. In shorter, a module is defined as the components which are modified and solved without effecting the other modules of the software. Thus each and every software design must and should follow the modularity. The process of breaking the entire software system into multiple modules which are independent and each and every module can be developed individually without depending on the other modules of the software is called as “Modularization”. The modular design can be made effective, if the modules which are divided can be modified and compiled independently. Thus even if there are any changes in the modules, after applying the changes there is no need to compile the entire system, instead we can compile the module itself which have undergone changes.

4.1.1 Benefits of Modules in Software Design:

- As entire software has been divided into multiple individual modules, it is very easy for the developers to understand the requirements of the customer and to implement them thus designing can become easy.
- Since the modules are independent on one another, it became easy to make the changes without affecting the entire system.
- Breaking down the entire system into modules saves time in both during testing and debugging.
- The independence of the modules in the software system can be measured using
  "Cohesion and Coupling".
- A good software design requires high coupling and low cohesion.

Cohesion is the measure of the degree to which the elements of the modules are functionally related. It is the degree to which all the elements are directed towards performing a single task which contains in the component. Basically, the cohesion is the internal glue which keeps the module together. A good software design will have a high cohesion. Cohesion describes the strength in relationship between different functions in a module. Cohesion is further classified into seven different types. They are

1. **Functional Cohesion:** Every important element for a single computation is contained in this component. A functional cohesion performs the task and the functions. It is an ideal situation.
2. **Sequential Cohesion:** An element gives output as some data that becomes the input for the other element, i.e., data flow between the parts. It occurs basically in the functional programming languages.
3. **Communicational Cohesion:** It is defined as two elements operate on the same input data or contribute towards the same output data. For example, update a record in the database and send it to the printer.

4. **Procedural Cohesion:** Elements of the procedural cohesion ensure about the order of the execution. Actions are still weakly connected and are unlikely to be reusable. For example, calculate a student GPA, print that student record, calculate the cumulative GPA, print the cumulative GPA.
5. **Temporal Cohesion:** in temporal cohesion, the elements are related by their timing involved. A module connected with the temporal cohesion all the tasks must be executed in the same time span. This cohesion contains the code for initializing all the parts of the system. Many different activities occur, all at the unit time.
6. **Logical Cohesion:** In logical cohesion, the elements are logically related and are not functionally. For example, a component reads the inputs from a tape, a disk and network. All the codes for these functions are in the same component. Operations are related, but the functions are significantly different.
7. **Co-incidental Cohesion:** In co-incidental cohesion, the elements are not related (unrelated). The elements have no conceptual relationship other than its location in the source code. It is an accidental and the worst form of cohesion. For example, print the next line and then reverse the characters of a string in a single component.

Coupling is the measure of the degree of the interdependence between the modules. A good software will have the low coupling. Describes the strength of the relationship between different modules in the software. Coupling is further broadly classified into six types in the order of high to low coupling. They are

1. **Data Coupling:** If the dependency between the modules is based on the fact that they communicate by passing only the data, then the modules are said to be data coupled. In the data coupling, the components are independent of each other and they communicate through data. Module communications do not contain tramp data. For an example, customer billing system.
2. **Stamp Coupling:** In the stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves the

tramp data. It may be necessary due to the efficiency factors like this choice was made by the insightful designer, not a lazy programmer.

3. **Control Coupling:** If the modules can communicate by passing control information, then they are said to be control coupled. It can be bad if the parameters indicate completely different behavior and indicate good if the parameters allow the factoring and the reuse of functionality. For example, the sort function that takes comparison function as an argument.
4. **External Coupling:** In an external coupling, the modules depend on the other modules, external to the software being developed or to a particular type of hardware. For example, device format, external file, protocol etc.
5. **Common Coupling:** The modules have shared data such as the global data structures. The changes in the global data means tracing back to all the modules which access that the data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses, and reduced maintainability.
6. **Content Coupling:** In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

Sign language detection using RESNET50 algorithm has following modules.

- Artificial neural networks
- Convolutional neural networks
- Tensor flow
- Open CV
- Keras
- Numpy

4.2 Description of Modules

Artificial Neural Networks:

Artificial Neural Network is a connections of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another

neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer.

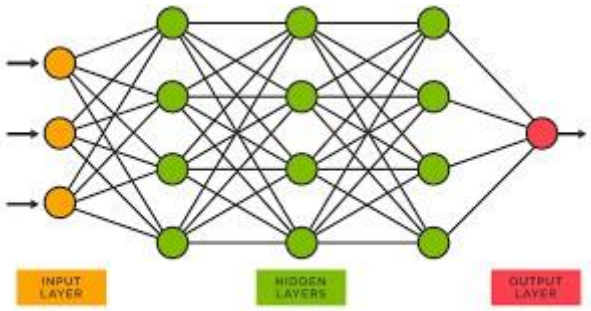

Figure 4-1 Layers in ANN

There are capable of learning and they have to be trained. There are different learning strategies:

1. Unsupervised Learning
2. Supervised Learning
3. Reinforcement Learning

Convolution Neural Network:

Unlike regular Neural Networks, in the layers of RESNET50, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the RESNET50 architecture we will reduce the full image into a single vector of class scores.

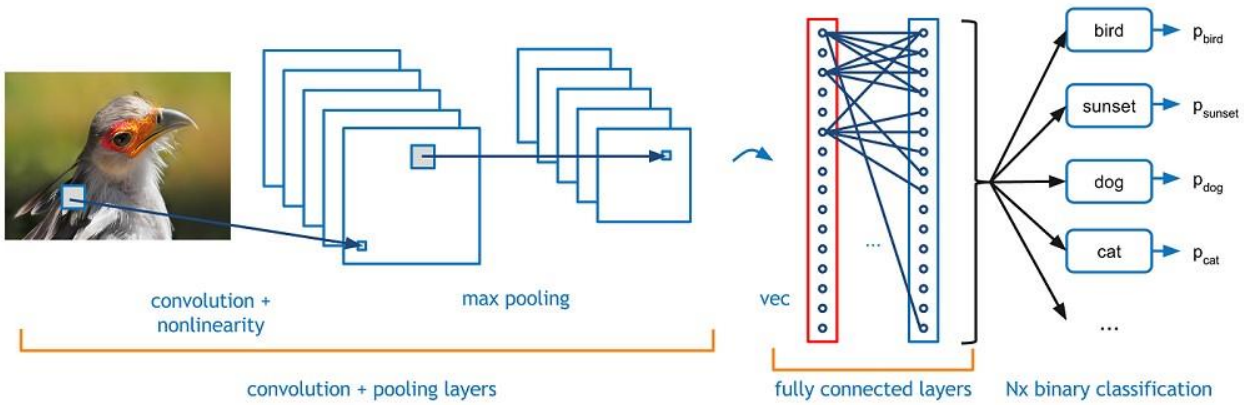

Figure 4-2 Layers in RESNET50

There are different layers in RESNET50

1. Input Layer

Input layer in RESNET50 should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input. If you have

"m" training examples then dimension of input will be (784, m).

2. Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer.

Convo layer also contains ReLU activation to make all negative

value to zero. **3. Pooling Layer**

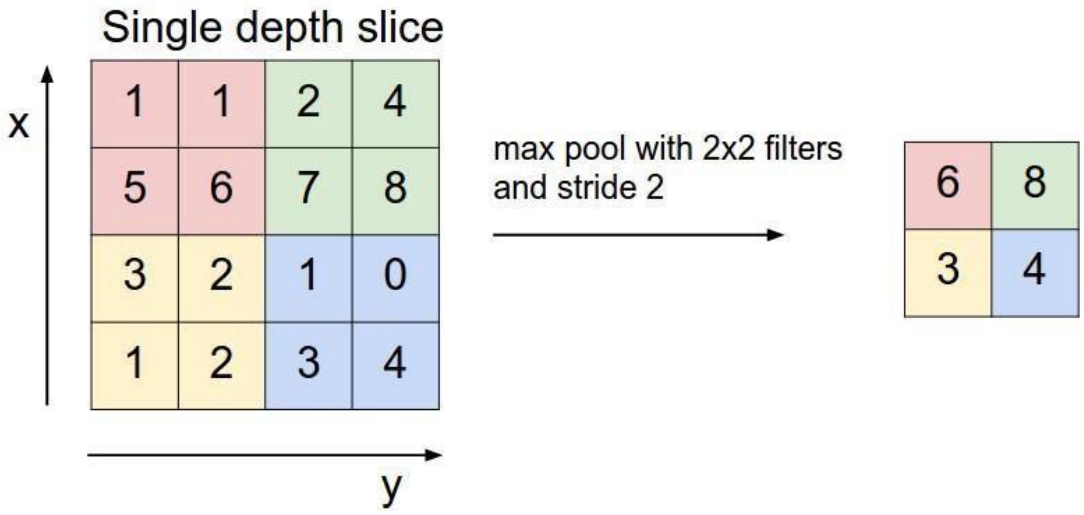

Figure 4-3 Pooling Layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don't want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

There is no parameter in pooling layer but it has two hyperparameters — Filter(F) and Stride(S).

In general, if we have input dimension W1 x H1 x D1, then

$W2 = (W1-F)/S+1$

$H2 = (H1-F)/S+1$

$D2 = D1$

Where W2, H2 and D2 are the width, height and depth of output.

4. Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.

5. Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of RESNET50. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

6. Output Layer

Output layer contains the label which is in the form of one-hot encoded.

Tensor Flow:

Tensor flow is an open source software library for numerical computation.

First we define the nodes of the computation graph, then inside a session, the actual computation takes place. Tensor Flow is widely used in Machine Learning.

Keras:

Keras is a high-level neural networks library written in python that works as a wrapper to Tensor Flow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

It is capable of running on top of TensorFlow, 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 Exception deep neural network model.

Pre-trained models

Trained model consists of two parts model Architecture and model Weights. Model weights are large file so we have to download and extract the feature from ImageNet database. Some of the popular pre-trained models are listed below,

- ResNet
- VGG16
- Mobile Net
- InceptionResNetV2
- InceptionV3

Open CV:

Open CV (Open Source Computer Vision) is an open source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, and MATLAB/OCTAVE.

OpenCV 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 opensource BSD license.

AForge.NET, a computer vision library for the Common Language Runtime (.NET Framework and Mono). ROS (Robot Operating System). OpenCV is used as the primary vision package in ROS. VXL, an alternative library written in C++. Integrating Vision Toolkit (IVT), a fast and easyto-use C++ library with an optional interface to OpenCV. CVIPtools, a complete GUI-based computer-vision and image-processing software environment, with C function libraries, a COMbased DLL, along with two utility programs for algorithm development

and batch processing. OpenNN, an open-source neural networks library written in C++ List of free and open source software packages

- OpenCV Functionality
- Image/video I/O, processing, display (core, import, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

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, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open source software and has many contributors.

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.

5 SYSTEM DESIGN

5.1 Introduction

System design is the phase which bridges the gap between the problem domain and the existing system in a manageable way. This phase focuses on the solution domain, which means “how to implement?”

It is the phase where the SRS document is converted into a format that can be implemented and will decide how the system will be operated. In this phase, the complex activity of the system development is divided into several small sub-activities, which coordinate with each other to achieve the main objective of the system development.

5.1.1 Inputs to System Design

System design takes the following inputs:

- Statement of the work.
- Requirement of determination of plan.
- Analysis of the current situation.
- Proposed system requirements including a conceptual data model, modified DFDs, and the Metadata (data about data).

5.1.2 Outputs for System Design

System design results the following outputs:

- Infrastructure and the organizational changes for the proposed system.
- The data schema is often a relational schema.
- Metadata to define the tables/files and the columns/data-items.

- A functional hierarchy diagram or a web page map that graphically describes the program structure.
- Actual or pseudo code for each module in the program.

5.1.3 Types of System Design

Logical Design

Logical design is appropriate to an abstract representation of the data flow, inputs, and the outputs of the system. It describes the sources (inputs), destinations (outputs), databases (data stores), data flows (procedures) all in a format which meets the user requirements.

While preparing the logical design of any system, the system analyst will specify the user needs at the level of detail that virtually determines the information flow into and out of the system and the required data sources. Data flow diagram, and E-R diagram modelling are used.

Physical Design

Physical design is related to the actual input and the output processes of the system. It focuses on how the data is entered into a system, verified, processed, and displayed as output. It produces the working system by defining the design specification that specifies exactly what the candidate system does. It is concerned with the user process design, interface design, and the data design.

It consists of the following steps,

- Specifying the input/output media, designing the database, and specifying the backup procedures.
- Planning the system implementation.

- Devising a test and an implementation plan, and specifying any new hardware and software.
- Updating costs, benefits, conversion dates, and system constraints.

Architectural Design

It is also known as a high level design that focuses on the design of the system architecture. It describes the structure and the behavior of the system. It defines the structure and the relationship between various modules of the system development process.

Detailed Design

It follows the architectural design and focuses on the development of each module.

Conceptual Data Modelling

It is a representation of the organizational data which includes all the major entities and the relationship. System analysts develop a conceptual data model for the current system that supports the scope and the requirements for the proposed system. The main aim of the conceptual data modeling is to capture the much meaning of the data as possible. Most of the organizations today use conceptual data modeling using E-R model which uses a special notation to represent the meaning about data as possible.

Entity Relationship Model

It is a technique used in the database design that helps to describe the relationship between various entities of an organization.

Terms used in E-R model

- **ENTITY** − It specifies the distinct real world items in an application. For example, item, vendor, teachers, course, students etc.
- **RELATIONSHIP** – Relationship is the meaningful dependencies between entities. For example, teacher teaches courses, vendor supplies items, then it is said to be that supplies and course are relationship.
- **ATTRIBUTES** − It specifies the properties of the relationships. For example, student name, vendor code. Symbols used in E-R model and their respective meanings are given below in the following table

Three types of relationships could exist between two sets of data. They are one-to-one, one-tomany, and many-to-many.

5.1.4 File Organization

It describes how the records are stored within a file. There are four file organization methods.

- **Serial** − Records are stored in a chronological order (in an order as they are input or occur). Examples are recording of telephone charges, ATM transactions and Telephone queues.
- **Sequential** – the records are stored in an order based on a key field which contains a value that uniquely identifies that record. For example, phone directories.
- **Direct (relative)** – Here every record is stored based on a physical address or at a location on the device. Address is calculated from the value which is stored in the record's key field. Hashing algorithm or randomizing routine does the conversion.
- **Indexed** − Records are processed both sequentially and non-sequentially using the indexes.

5.1.5 File Access

A user can access a file using either Random Access or Sequential Access. File Access methods allow the computer programs read or write the records in a file.

Sequential Access

Every record on the file is processed starting with the first record until End of File (EOF) is reached. It is efficient when a large number of the records on the file are needed to be accessed at any given time. Data stored on a tape (sequential access) can be accessed only sequentially.

Direct (Random) Access

Records are located by knowing their physical locations or the addresses on the device rather than their positions relative to the other records. Data stored on a CD device (direct-access) can be accessed either sequentially or randomly.

5.1.6 Advantages

- It can reduce the system downtime, speed up maintenance tasks, cut costs.
- It provides the clear description of the formal flow of present system and helps to understand the type of the input data and how the output can be produced.
- It provides an effective and an efficient way of communication between the technical and the nontechnical users about the system.
- It facilitates the training of a new user so that he can easily understand the flow of the system.
- It help the users to solve the problems such as troubleshooting and helps the manager to take better final decisions of the organization system.

- It provides a better control to the internal or external working of the system.

5.1.7 Types of Documentations

When it comes to the System Design, there are following four main documentations.

- Program documentation
- System documentation
- Operations documentation
- User documentation

5.1.8 Program Documentation

- It describes the inputs, outputs, and processing logic for all the program modules.
- The program documentation process starts in the system analysis phase and continues during the implementation.
- This documentation guides the programmers, who construct the modules that are well supported by the internal and the external comments and the descriptions that can be understood and are maintained easily.

5.1.9 User Documentation

It includes the instructions and the information to the users who will interact with the system. For example, tutorials, help guides and user manuals. User documentation is valuable for the training users and for the reference purpose. It must be clear, understandable, and should be readily accessible to the users at all levels. The analysts, system owners, users and the programmers, all put combined efforts to develop a user's guide.

A user documentation should include,

- A system overview that clearly describes all the major system features, capabilities, and the limitations.
- Description of the source document content, preparation, processing, and the samples.
- Overview of the menu and data entry screen options, contents, and the processing instructions.
- Examples of the reports that are produced regularly or available at the user's request, including samples.
- Security and audit trail information.
- Explanation of the responsibility for the specific input, an output, or the processing requirements.
- Procedures for the requesting changes and the reporting problems.
- Examples of the exceptions and the error situations.
- Frequently asked questions (FAQs).
- Explanation of how to get help and procedures for updating the user manual.

5.2 Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system.

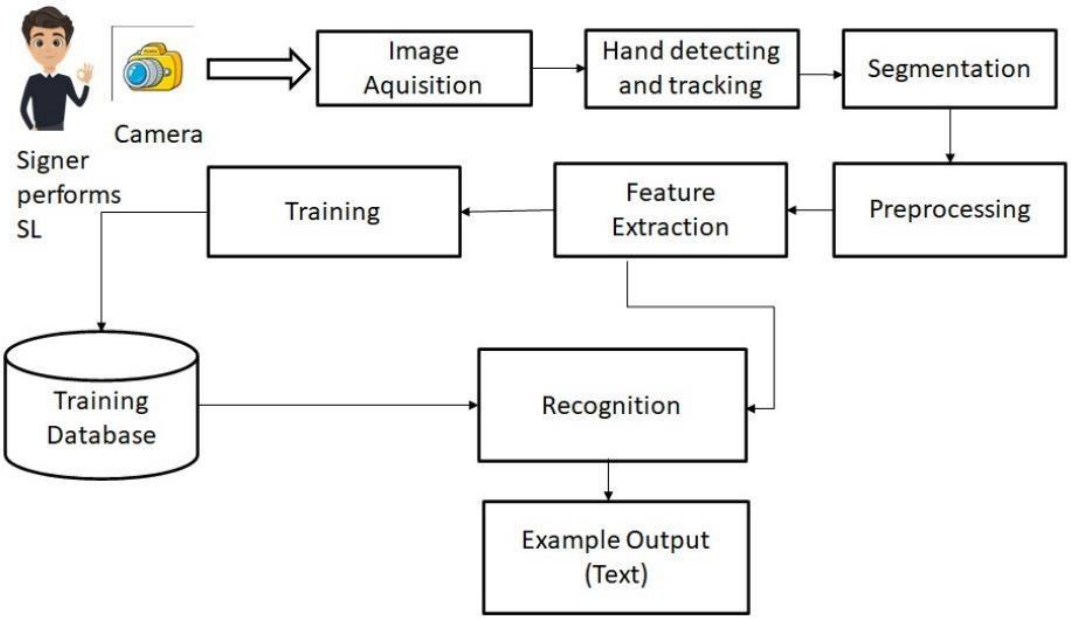

Figure 5-1 System Architecture

5.3 UML Diagrams

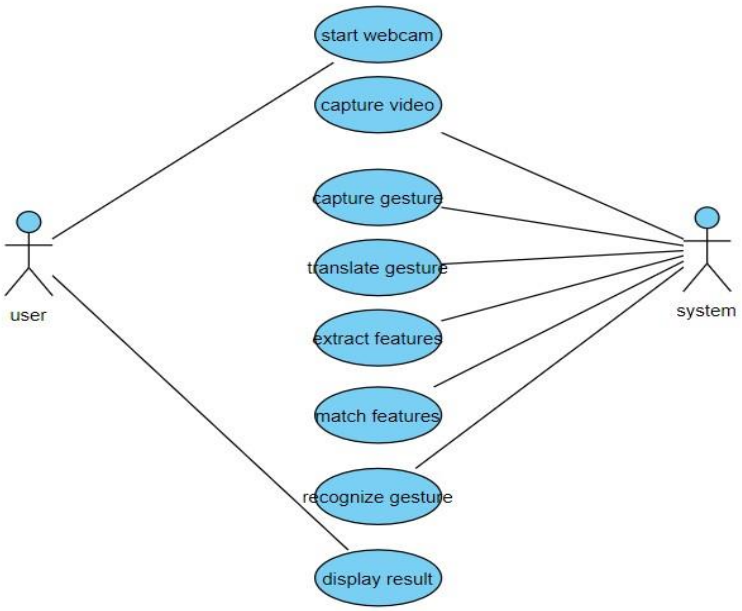

Figure 5-2 Use Case diagram by using user and system as actors

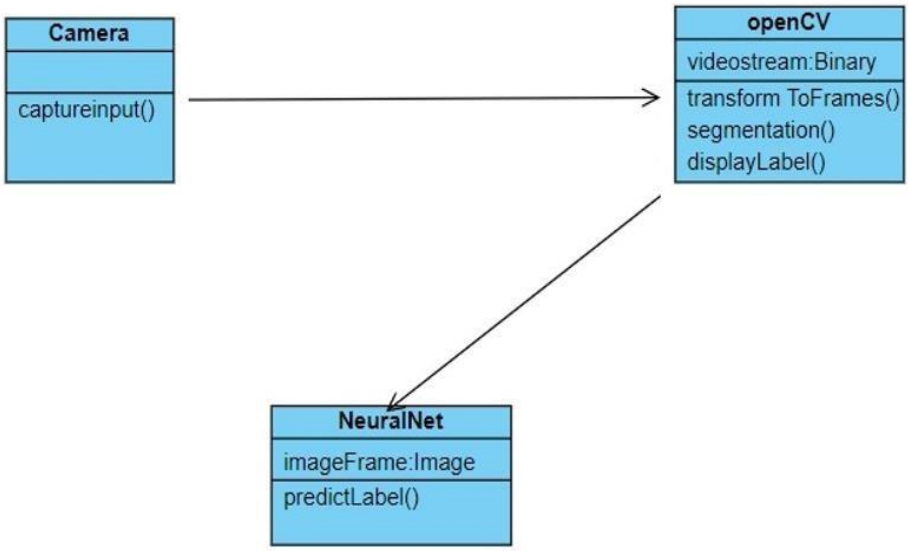

Figure 5-3 Class diagram by using camera, openCV and NeuralNet Classes

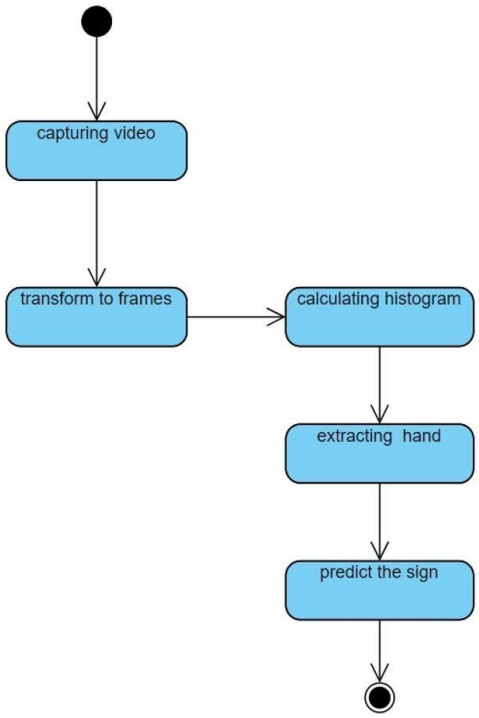

Figure 5-4 State diagram for sign language detection

6 SYSTEM IMPLEMENTATION

6.1 INTRODUCTION:

Systems implementation is a set of procedures performed to complete the design contained in the approved systems design document and to test, install, and begin to use the new or revised Information System. Transitioning the system support responsibilities involves the changing from the system development to the system support and the maintenance mode of operation, with an ownership of the new system moving from the Project Team to the Performing Organization. A key difference between the System Implementation and all other phases of the lifecycle is that all the project activities up to this point have been performed in secure, protected, and safe environments, where the project issues that arise have a little or no impact on the day-today business operations. Once the system begins live, however, this is no longer the case. This phase consists of these following processes:

- Prepare the System Implementation, where all the steps needed in advance of deploying the applications are performed, including the preparation of both the production environment and the Consumer communities.
- Deploy the system, where the full deployment plan, initially developed during the system design and evolved throughout the subsequent lifecycle phases, are executed and validated.
- Transition to the performing organization, where the responsibility and the ownership of the application are transitioned from the project team to the unit in the performing organization that will provide the system support and the maintenance.

System Implementation serves as it is own measurement of success; indeed, a smooth System Implementation reaches the climax and validates the entire system development effort. However, even before the final turnover, the Project Manager can utilize the measurement criteria below to assess how successfully the implementation is proceeding. More than one “No” answer indicates that a serious risk to the success of this phase has occurred and to the entire project.

6.2 Prepare for System Implementation

The purpose of prepare for system implementation is to take all the possible steps to ensure that the upcoming system deployment and the transition occurs flawlessly, efficiently, and smootly

In the implementation of any new system, it is necessary to ensure that the consumer community is the best positioned to utilize the system once the deployment efforts have been validated. Therefore, all the necessary training activities must be scheduled and must be coordinated. Since this training is often the first exposure to the system for many individuals, it should be conducted as professionally and competently as possible. A positive training experience is the great first step towards the customer acceptance of the system. During the system implementation it is essential that everyone involved have to be absolutely synchronized with the deployment plan and with each other. Often the performance of the deployment efforts impacts many of the performing organization’s normal business operations. Examples of these impacts will include:

- Consumers may experience the period of time in which the systems that they depended on to perform their jobs are temporarily unavailable to

them. They may be asked to maintain the detailed manual records or the logs of the business functions that they perform to be entered into the new system once it is operational.

- Technical services personnel may be required to assume the significant implementation responsibilities while at the same time having to continue the current levels of the service on other critical business systems.
- Technical support personnel may experience the unusually high volumes of the support requests due to the possible disruption of the day-to-day processing.

Because of these and the other impacts, the communication of the planned deployment activities to all the parties involved in the project are critical. A smooth deployment requires a strong planning, leadership and communications. By this point in the project lifecycle, the team will have to spend the countless hours devising and refining the steps to be followed. During this preparation process the project manager must verify that all the conditions that must be met prior to initiate the deployment activities have been met, and that the final 'green light' is on for the team to proceed.

The final process within the System Development Lifecycle (SDLC) is to transition the ownership of the system support responsibilities to the Performing Organization. In order to this, there should be an efficient and an effective transition, the Project Manager should make sure that all the involved parties are aware of the transition plan, the timing of the various transition activities, and their role in its execution. Due to the number of project participants in this phase of the SDLC, many of the necessary conditions and the activities may be beyond the direct control of the Project Manager. Consequently, all the Project Team members with the roles in the implementation efforts must understand the plan,

acknowledge their responsibilities, recognize the extent to which other implementation efforts are dependent upon them, and confirm their commitment.

6.3 Deploy System Process

The purpose of the Deploy System process is to perform all the activities required to successfully install the new system and make it available to the consumers.

Deploying the system is the culmination of all prior efforts, where all of the meetings, deliverable reviews, planning sessions, development, prototypes and the testing pay off in the delivery of the final system. It is also that the point in the project that often requires the most coordination, due to the breadth and the variety of the activities that must be performed. Depending upon the complexity of the system being implemented, it may affect the technical, operational, and the cultural aspects of the organization. A representative sample of the high-level activities might include the installation of a new hardware, increased network capabilities, deployment and the configuration of the new system software, the training and awareness campaign, activation of new job titles and its responsibilities, and a completely new operational support structure aimed at providing the consumer-oriented assistance during the hours that the new system is available for use (to name a few). Whatever the realm of the activities related to the new system, their impacts should be addressed in the Organizational Change Management Plan, while the specific deployment activities should all be encompassed in the Project Implementation and the transition Plan (both created during the Project Planning phase of the Project Management Lifecycle.)

All consumer training should be performed prior to the physically migrating the system to the production environment. This will enable the consumers to begin

to familiarize themselves with the system, and will help to establish their expectations regarding what the system will do and will not do.

The sequencing of the deployment activities is just as important as it was with the previous testing activities. This sequence of those events should be encompassed in the Deployment and Transition Plan section of the Technical Specification, and will address and will prioritize any necessary training activities, set-up activities which are needed to prepare the production environment such as desktop, Local Area Network, servers, data center, etc., and data the conversion and the validation activities. This deployment plan will also define the steps for the physically migrating of the system and any associated utilities to the production, and for validating the accuracy and the completeness of this migration after these steps have been performed. During deployment, the project team members may often be required to work for the extra hours to meet the aggressive timeframes, or additional staff may be brought in temporarily to assist with the large data entry efforts. Proper planning and the sequencing of the deployment can help to minimize these situations, and reduce the chance of any missteps that could result in having to restart the deployment process, or lengthen the implementation schedule. As the system is enabled, and the Project Team validates that the application is performing to the expectations, there may be times when the certain system functions seems to be suspected. One of the challenges most frequently faced by the project teams is to determine the root cause of the potential issues. Discrepancies that exist within the data could be due to the defects in the application's business logic, or could be the result of the data that was improperly migrated into the system. Similarly, the inability of a consumer to access specific features of the system could be caused by an

improperly configured hardware, or incorrectly established security privileges. To minimize the confusion and to reduce the opportunity for such issues to the surface, every attempt should be made to immediately validate every step of the deployment as it is performed.

6.4 Transition to Performing Organization

The purpose of the Transition to Performing Organization process is to successfully prepare the Performing Organization to assume the responsibility for maintaining and for supporting the new application.

In most of the organizations, the team of the individuals may be are responsible for the long-term support and the maintenance of the system is different from the team which is initially responsible for the designing and developing the application. Often, the two teams include a comparable set of the technical skills. The responsibilities associated with the supporting of an operational system, however, they are different from those associated with a new development. In order to affect this shift of the responsibilities, the Project Team must have to provide the responsibility for the system support in the Performing Organization with a certain combination of the technical documentation, training, and hands-on assistance to enable them to provide an acceptable level of operational support to the consumers. This system transition is one element (albeit a major one) of the overall project implementation and the transition plan, developed as part of the PM Lifecycle. The Project Manager should review the transition plan to confirm that all those defined actions have been successfully completed.

7 TESTING

7.1 Introduction

7.1.1 Software testing

Software testing is the process of finding defects in the software so that these can be debugged and the defect-free software can meet the customer needs and expectations. Software testing is one of the important phases in software development life cycle. A quality software can be achieved through testing. Effective testing reduces the maintenance cost and provides reliable outcomes. The intention of software testing process is to produce a defect-free system.

7.1.2 Types of testing

Functional:

The ultimate goal of functional testing is to ensure that software works according to specifications and in line with user expectations. While the objective sounds simple, the task involves many functional testing types, some of which might be preferred or prioritized over others depending on the nature of the application and organization

Build Acceptance Testing or Build Verification Testing

Build Verification Testing (BVT), also known as Smoke Tests or Build Acceptance Testing (BAT), is a type of software testing that is aimed at ensuring the most important functions work correctly when you push new code. The results of this testing are used to decide if a build is stable enough to proceed with further testing. Typically the BVT process is automated.

Sanity Testing

A type of regression testing, QA professionals perform sanity testing on new versions of stable builds to validate either new functionality or bug fixes. While similar to smoke testing in that both provide a gate check that a build is ready for more testing, sanity testing is unscripted and specifically targets the area that has undergone a code change.

Sanity testing example: A web page for a telehealth provider returns a 404 error for its mental health page. The developers fix the issue, then commit the build for testing. The QA professional performs a sanity check to determine whether the basic functionality and navigation for that specific page work as intended.

Smoke Testing

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing.

Regression Testing

Just because functional tests pass once doesn’t mean they’ll always pass. When developers commit new code or change a feature, you run regression tests to make sure the software still functions as expected. Regression testing helps

maintain a stable product while changes are made to it. Regression tests are often automated.

Regression testing example: A clothing retailer adds the ability to pay with customer rewards points on their mobile app. Testers might perform regression tests on other existing functionality, such as the ability to pay with credit cards and gift cards, to make sure all forms of payment work correctly.

Re-Testing

Retesting is a type of testing performed to check the test cases that were unsuccessful in the final execution are successfully pass after the defects fixed. Usually, tester assigns the bug when they find it while testing the product or its component. The bug allocated to a developer, and he fixes it. After fixing, the bug is assigned to a tester for its verification. This testing is known as retesting.

Alpha Testing

Another subset of acceptance testing, alpha testing uses internal team members to evaluate the product. These team members should be knowledgeable of the project but not directly involved in its development or testing. Where some builds might still be somewhat unstable, alpha testing provides an immediate subset of testers to root out major bugs before the software is seen by external users.

Alpha testing example: In this functional testing example, a casino games provider releases a new version of its app that includes video poker. The organization compiles a cross-functional group of internal users that test whether

the app functions correctly on their devices and how the user experience can improve.

Beta Testing

After the internal team tests the product and fixes bugs, beta testing occurs with a select group of end users. Beta testing serves as a soft launch, enabling you to get feedback from real users who have no prior knowledge of the app. Beta testing enables you to gather feedback from unbiased users who may interact with the product differently than you intended, perhaps identifying critical unknown bugs before release to a wide user base.

Beta testing example: A restaurant chain releases a new mobile order and pickup system. Before the company releases the functionality to all of its mobile app users, it tests the app with a small number of dedicated customers and provides them with rewards for participating.

Installation Testing

Installation testing is check that software application is successfully installed & it is working as expected after installation. This is testing phase prior to end users will firstly interact with the actual application. Installation testing is also called as “Implementation Testing”. This is most important as well as most interesting step in the Software testing life cycle.

Usability Testing

Usability testing refers to evaluating a product or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and takes notes. The goal is to identify

any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product.

Reliability Testing

Reliability Testing is a software testing process that checks whether the software can perform a failure-free operation for a specified time period in a particular environment. The purpose of Reliability testing is to assure that the software product is bug free and reliable enough for its expected purpose.

Reliability means “yielding the same,” in other terms, the word “reliable” means something is dependable and that it will give the same outcome every time. The same is true for Reliability testing.

Security Testing

All apps and websites require robust security to maintain consumer trust and protect both data and intellectual property. There are numerous automated scans and assessments that every company should perform as part of security testing, but active testing elements should bring the human side into testing as well.

Applause provides a global team of white-hat hackers to penetration test your digital products. These penetration testers attempt to break into your protected system to identify vulnerabilities. Applause provides these insights securely within our platform and offers a breadth of knowledge and experience to give peace of mind that a release meets high security standards.

Non Functional:

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a

software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software. Non-functional testing is equally important as functional testing and affects client satisfaction.

Performance Testing

Performance testing checks how well software components work. These tests find issues in software design and architecture performance. This is typically done by: Measuring response times, Identifying bottlenecks and locating failure points. Performance tests ensure software quality. They validate that it's fast, scalable, stable, and reliable.

Unit Level Testing

Unit means a program unit, module, component, procedure, subroutine of a system developed by the programmer. The aim of unit testing is to find bugs by isolating an individual module using test stub and test drivers and by executing test cases on it. During module testing, test stub and test drivers are designed for proper testing in a test environment. The unit testing is performed to detect both structural and functional errors in the module. Therefore, test cases are designed using whitebox and black-box testing strategies for unit testing. Most of the module errors are captured through white-box testing.

Integration Level Testing

Integration testing is another level of testing, which is performed after unit testing of modules. It is carried out keeping in view the design issues of the

system into subsystems. The main goal of integration testing is to find interface errors between modules. There are various approaches in which the modules are combined together for integration testing.

- Big-bang approach
- Top-down approach
- Bottom-up approach
- Sandwich approach

User Acceptance Testing (U.A.T)

Acceptance testing is a kind of system testing, which is performed before the system is released into the market. It is performed with the customer to ensure that the system is acceptable for delivery. Once all system testing have been exercised, the system is now tested from the customer's point of view. Acceptance testing is conducted because there is a difference between the actual user and the simulated users considered by the development organization. The user involvement is important during acceptance testing of the software as it is developed for the endusers. Acceptance testing is performed at two levels, i.e., Alpha testing and Beta Testing.

8 CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

- In this report, a functional real time vision based American Sign Language recognition for D&M people have been developed for ASL alphabets.
- We achieved final accuracy of 95.7% on our dataset.
- We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other. This way we are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.
- We also implemented a feature which makes sign language training sessions or tutorials for the people who don't know the gestures. The user gives input in the form of words the system will show the gestures in the form of slideshow. So people who are using this application or software will be knowing the basics of the gestures which makes the job easy for the user to understand the gestures of the deaf & dumb people.

8.2 Future Scope

We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms. We are also thinking of improving the preprocessing to predict gestures in low light conditions with a higher accuracy.

As it is mentioned in the conclusion this software is very flexible for the future updates and it can be extended for the future purposes or updates anytime. So, we are planning to improve this application to achieve higher accuracy in the even in a case of complex backgrounds by trying out various background applications like background subtraction algorithms. This subtraction algorithms

will help to improve the system to recognize signs made by the user in a more accurate and efficient way. We are also thinking of improving the preprocessing to predict the gestures in low light conditions with a higher accuracy.

By improving the preprocessing accuracy, the low light problem will be resolved. There will be constant updates in the software where it improves the performance of the application and it becomes more easy to use by the user. We are trying to add a one more feature where the user can record his own gestures and save it in the database where the user can use it anytime in the real world and communicate just by using his own gestures.

We are planning to implement a feature which takes video as an input which has signs and converts into corresponding letters, words and sentences in the future implementation.

9 BIBILOGRAPHY

9.1 Web links

https://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf
https://ieeexplore.ieee.org/document/7507939
https://www.sciencedirect.com/science/article/pii/S1877050917320720
https://ieeexplore.ieee.org/document/7916786
https://www.iosrjen.org/Papers/vol3_issue2%20(part-2)/H03224551.pdf

9.2 References

[1] Pavlovic, V, Sharma, R., &Huang T. (1997). Visual Interpretation of Hand Gestures for Human-Computer Interaction (HCI): A Review. IEEE TOPAMI, VOL. 19, NO. 7.

[2] Jiyong, M.W., Gao, Jiangqin, &Wu Chunli, Wang (2000). A Continuous Chinese sign language recognition system in Automatic Face and Gesture Recognition. Fourth IEEE International Conference on 2000.

[3] Vogler, C.M., &Dimitris (2004). Handshapes and Movements: Multiple-Channel American Sign Language Recognition Gesture-Based Communication in Human-Computer Interaction. Ed. A.V.Camurri, Gualtiero. Vol. 2915: Springer Berlin Heidelberg. 247-258.

[4] Gunasekaran, K., &Manikandan, R. (2013). Sign Language to Speech Translation System Using PIC Microcontroller. International Journal of Engineering and Technology (IJET), Vol 5 No 2.

[5] Kalidolda, N., &Sandygulova, A. (2018). Towards Interpreting Robotic System for Fingerspelling Recognition in Real-Time. HRI Companion: 2018 ACM/IEEE International Conference on Human-Robot Interaction Companion.

[6] Liang, R., & Ouhyoung, M. (1998). Real-time continuous gesture recognition system for sign language. Proc Third. IEEE International Conf: on Automatic Face and Gesture Recognition, pp. 558-567.

[7] LeCun, Yann, Lon, Bottou, Yoshua, Bengio, & Haffner, Patrick (1998). Gradient-based learning applied to document recognition', Proceedings of the

IEEE 86, vol. 11, pages 22782324. [8] Simonyan, Karen, & Zisserman, Andrew (2015). Very Deep.... Recognition. Conference paper ICLR.

[9] “Deafness and hearing loss,” World Health Organization. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafnessand-hearing-loss.

[10] “American Sign Language,” National Institute of Deafness and Other Communication Disorders, 14-Dec-2020. [Online]. Available: https://www.nidcd.nih.gov/health/american-signlanguage.

[11] J. Wu, L. Sun and R. Jafari, "A Wearable System for Recognizing American Sign Language in Real-Time Using IMU and Surface EMG Sensors," in IEEE Journal of Biomedical and Health Informatics, vol. 20, no. 5, pp. 1281-1290, Sept. 2016.

[12] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and

Computer Vision," 2018 IEEE International Conference on Big Data (Big Data),

Seattle, WA, USA, pp. 4896-4899, 2018.

[13] Sruthi, C. J., and A. Lijiya. "Signet: A deep learning based indian sign language recognition system." 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0596-0600. IEEE, 2019.

[14] Rao, G. Anantha, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry. "Deep convolutional neural networks for sign language recognition."

[15] Poppe, R.: A survey on vision-based human action recognition. Image and vision computing 28(6), 976–990 (2010)

[16] Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13). pp. 1139–1147 (2013)

[17] Van Herreweghe, M.: Prelinguaal dove jongeren en nederlands: een syntactischonderzoek.

Universiteit Gent. Faculteit Letteren en Wijsbegeerte (1996) 8 L. Pigou, S. Dieleman, P.

Kindermans, B. Schrauwen

[18] Verschaeren, R.: Automatische herkenning van gebaren met de microsoft kinect(2012) [19] Zaki, M.M., Shaheen, S.I.: Sign language recognition using a combination of new vision based features. Pattern Recognition Letters 32(4), 572–577 (2011)

[20] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional neural networks. arXiv preprint arXiv:1311.2901 (2013)

9.3 Books

[1] Dr. Ulhas D. Shiurkar , Prof. Anita Nikalje Development of Sign Language Recognition Techniques

[2] George Awad A Framework for Sign Language Recognition

[3] Deshmukh Mona Phrase and Sign Language Recognition and Translation