A Project-II Report
on

SIGN LANGUAGE RECOGNITION USING COMPUTER VISION AND NEURAL NETWORKS

Submitted in partial fulfillment of the requirement
For
The award of degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING
2018-2022

Submitted By

Hrushyang Adloori	18311A05C1
Abhiram Reddy Guduru	18311A05D3
Rishith Muthyapu	18311A05F4

Under the Guidance of
Mr. K. Damodhar Rao
Assistant Professor

Department of Computer Science & Engineering
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)
Yamnampet (V), Ghatkesar (M), Hyderabad – 501301, A.P.

AY-2021-2022

Department of computer Science and Engineering

Sreenidhi Institute of Science and Technology

CERTIFICATE

This is to certify that this Project-II report on "**Sign Language Recognition using Computer Vision and Neural Networks**", submitted by **Hrushyang Adloori (18311A05C1), Abhiram Reddy Guduru (18311A05D3), Rishith Muthyapu (18311A05F4)** in the year 2022 in the partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as a part of their **Project-II during Fourth Year Second Semester,** under our guidance. This report has not been submitted to any other Institute or university for the award of any degree.

Internal guide	**Project Coordinator**	**Head of the Department**
Mr. K. Damodhar Rao	Mrs. M. Yellamma	Dr. Aruna Varanasi
Assistant Professor	Assistant Professor	Professor & HOD
Department of CSE	Department of CSE	Department of CSE

Signature of the External Examiner

Date:-

DECLARATION

We **Hrushyang Adloori (18311A05C1), Abhiram Reddy Guduru (18311A05C1), Rishith Muthyapu (18311A05C1)** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR,** studying IV year II semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Project-II work, titled "**SIGN LANGUAGE RECOGNITION USING COMPUTER VISION AND NEURAL NETWORKS**" is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **DR. T. Ch. Siva Reddy,** who most ably runs the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Dr. Aruna Varanasi**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank our Coordinator **Mrs. M. Yellamma** & my internal guide **Mr. K. Damodhar Rao** for Project-II for their technical guidance, constant guidelines, encouragement and support in carrying out my project on time at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

Hrushyang Adloori **18311A05C1**

Abhiram Reddy Guduru **18311A05D3**

Rishith Muthyapu **18311A05F4**

ABSTRACT

Speech impairment is a condition that limits an individual's ability to communicate verbally. Consequently, communication between a verbally handicapped person and a normal person has always been challenging. To solve this problem, sign language, which is one of the most ordered languages, has been adopted. This initiative aims to bridge the gap between differently-abled people, such as the deaf and dumb, and the general public. The use of hand landmarks in conjunction with deep learning aided in the development of a real-time prediction system. The aim is to fabricate alphabets in sign language from real-time hand gestures using Computer Vision. Primarily, the dataset of ASL alphabet is used which is available on Kaggle. The dataset is undergone a meticulous pre-processing for further usage. Media pipe framework developed by google is utilized for pre-processing. The framework extracts hand landmarks from the pictures in the dataset. These coordinate data is further processed into more efficient information through specific techniques. The final processed dataset is used to training a neural net model. This model is trained to classify the 26 alphabets of the English literature which are represented by Sign. This proposed system provides better flexibility and makes the model more trainable for specific instances. Hence is implemented to deploy a real-time ASL alphabet classifier that recognizes the sign alphabets using the constant visual input provided to it.

LIST OF FIGURES

LIST OF TABLES

INDEX

1. INTRODUCTION

Communication is the key to unlocking the doors to the outside world, and only those who lack it can appreciate its significance. In an experiment done by Orfield Laboratories, it was discovered that a normal individual could only stay comfortable in a quiet room for 45 minutes. Imagine spending your entire life in such a peaceful environment. Deaf and mute persons confront similar problems on a daily basis. According to WHO statistics, roughly 466 million people, or about 5% of the world's population, suffer from such impairments, with 35 million of them being children. For those persons, sign language was created so that they could communicate more easily. It is the most organised language, with each and every gesture having a distinct meaning. It also has its own set of grammatical symbols for connecting the words. As a result, they can only converse with others who are already familiar with sign language. Others, on the other hand, may have difficulties communicating. As a result, a technology to recognise sign language is required so that such a person may readily converse with regular people. The fundamental disadvantage of a vision-based sign language recognition system is that the picture collecting process is subject to several environmental concerns, such as camera placement, backdrop conditions, and lightning sensitivity. However, it is more convenient and cost-effective than employing a sensor and tracker to collect data. As a result, preprocessed pictures from the dataset are input into the convolutional neural network, which is subsequently trained and evaluated. It can recognise the indications done in real-time after being tested using the dataset image.

1.1 American Sign Language

ASL is a unified, natural language with linguistic features akin to spoken languages and a grammar distinct from English. Hand and face gestures are used to communicate in ASL. It is the first language of many deaf and hard-of-hearing North Americans, as well as some hearing persons. The American Sign Language (ASL) is a completely separate language from English. It has its own set of rules for pronunciation, word formation, and

word order, as well as all the other essential language characteristics. Distinct languages have certain ways of expressing different functions, such as asking a question instead of making a statement. For example, English speakers can pose a question by raising their voices and modifying the word order, but ASL users can convey a question by lifting their eyebrows, widening their gaze, and gently leaning their body. ASL users, like speakers of other languages, may communicate ideas in a variety of ways. ASL shows regional variances in expression in addition to individual variability. ASL users, like speakers of other languages, may communicate ideas in a variety of ways.

ASL has regional accents and dialects, just as certain English words are spoken differently in different parts of the country; ASL has regional variations in the rhythm of signing, pronunciation, slang, and signs used, just as certain English words are spoken differently in different parts of the country; Age and gender, as well as other sociocultural factors, can influence ASL usage and offer variation, much as they do with spoken languages. Fingerspelling is a kind of American Sign Language (ASL) that is used to sign English words. Each letter of the alphabet is represented by a different hand shape in the fingerspelled alphabet. Fingerspelling is commonly used to represent proper names or the English phrase for anything.

1.1.1 ASL Fingerspelling

The American manual alphabet (ASL) is a collection of 26 signals that may be used to spell out English words. In these signals, the 19 ASL hand forms are employed. The signs for 'p' and 'k,' for example, have the same handshape but are oriented differently. A widespread misunderstanding is that ASL is only based on fingerspelling; while this approach (the Rochester Method) has been utilized, it is not ASL. Fingerspelling is a type of borrowing that involves incorporating words from one language into another. Fingerspelling is used in ASL for proper names and technical terms without an ASL counterpart. Fingerspelling can also be used to highlight a word that would otherwise be signed.

Fingerspelling is a spelling technique in which you spell words with your hands. In sign language, the fingerspelling manual alphabet is used to spell out the names of persons and locations for whom no symbol exists. Fingerspelling may also be used to spell out new phrases on signs or to explain a sign to someone who is unfamiliar with it. Other ASL signals frequently contain fingerspelling signs. The one-handed alphabet is used in American Sign Language (ASL), whereas some other sign languages use the two-handed alphabet.

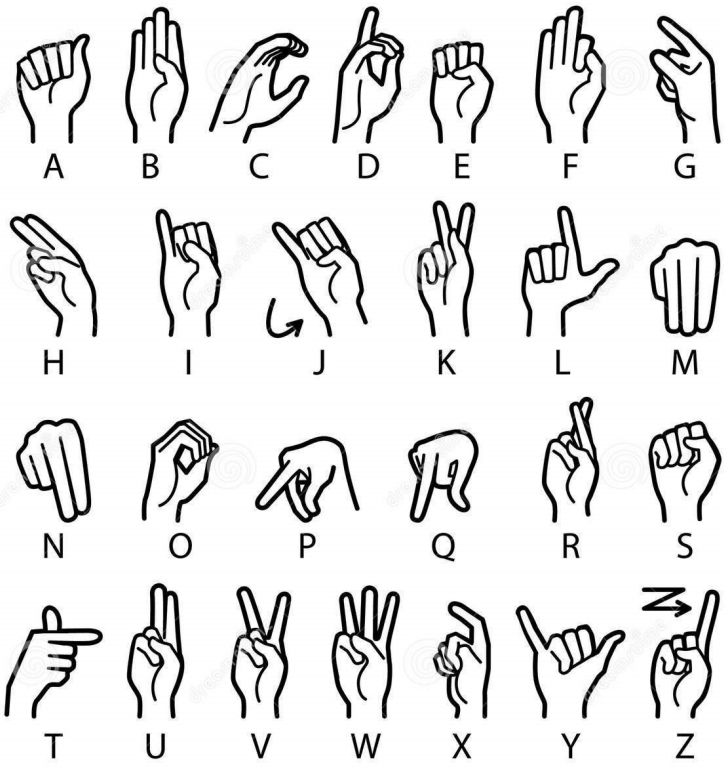


Fig 1.1 ASL – Fingerspelling Alphabet

1.2 Scope

Computer vision is the process of giving machines vision so that they can extract key elements from collected pictures. The subdomains of this large discipline include image processing and machine learning. Machine learning is used with image processing in this

work. To remove the hand from the backdrop, images of the hand are collected and preprocessed. Following preprocessing procedures, images are scaled down and a dataset is created. As a result, the dataset includes photos with no backdrop and that have been scaled down.

As more of a conclusion, this real-time picture recognition might be extremely beneficial to those with disabilities. They can simply send their message via real-time interaction, and the translated text may be presented on-screen or used to make the system voice-enabled. In American Sign Language, Fig. 1 depicts many signs that represent the alphabets of the English language.

1.3 Project Overview

The project revolves around identifying and classifying the ASL alphabet in real-time using live visual feed. Various mathematical and scientific methodologies are effectively implemented to further streamline the process. Data pre-processing is heavier side of the task in this project as it deals with image data. The process was meticulously operated with precious techniques to eliminate edge cases as much as possible. Initially we took 3,000 images of real-time hand poses of each alphabet. That sums up to 78,000 images that consisted in raw dataset. These images are all filtered through multiple processes to condense the dataset into more efficient and cleaner dataset. This data is fed into a neural network architecture that is designed to maximize model's accuracy and reduce the loss to acceptable minimum. All this work intertwined helps us to build an application that recognizes the 26 ASL alphabets individually in real-time.

1.4 Objectives

- To pre-process the images of each alphabet in the dataset using image processing libraries in Python.
- Applying adequate and applicable mathematical coordinate conversion techniques for converting the image data into csv data.
- Designing a neural network architecture with hidden layers and neurons that produce effective comprise between speed and accuracy.
- Combining all the work into usable interface for the users to interact and adapt easily.

1.5 Dataset

The data collection consists of photographs of alphabets in American Sign Language, which are organized into 29 folders that correspond to various classes. The training data set contains 87,000 200x200 pixel images. There are 29 classes in all, with 26 for letters A to Z and three each for SPACE, DELETE, and NOTHING. In real-time and classification applications, these three classes are particularly valuable. The test data collection contains just 29 photographs to encourage the use of real-world test shots.

The image data utilized in this project are only of the alphabet i.e., 78,000 images which are 200x200 pixels. Simplifying the classification task for the model to perform a bit. The other functionalities are dealt in different and more sensible way which is elaborated further in the report.

Below is the link to access the utilized dataset which is available for public usage:
https://www.kaggle.com/datasets/grassknoted/asl-alphabet

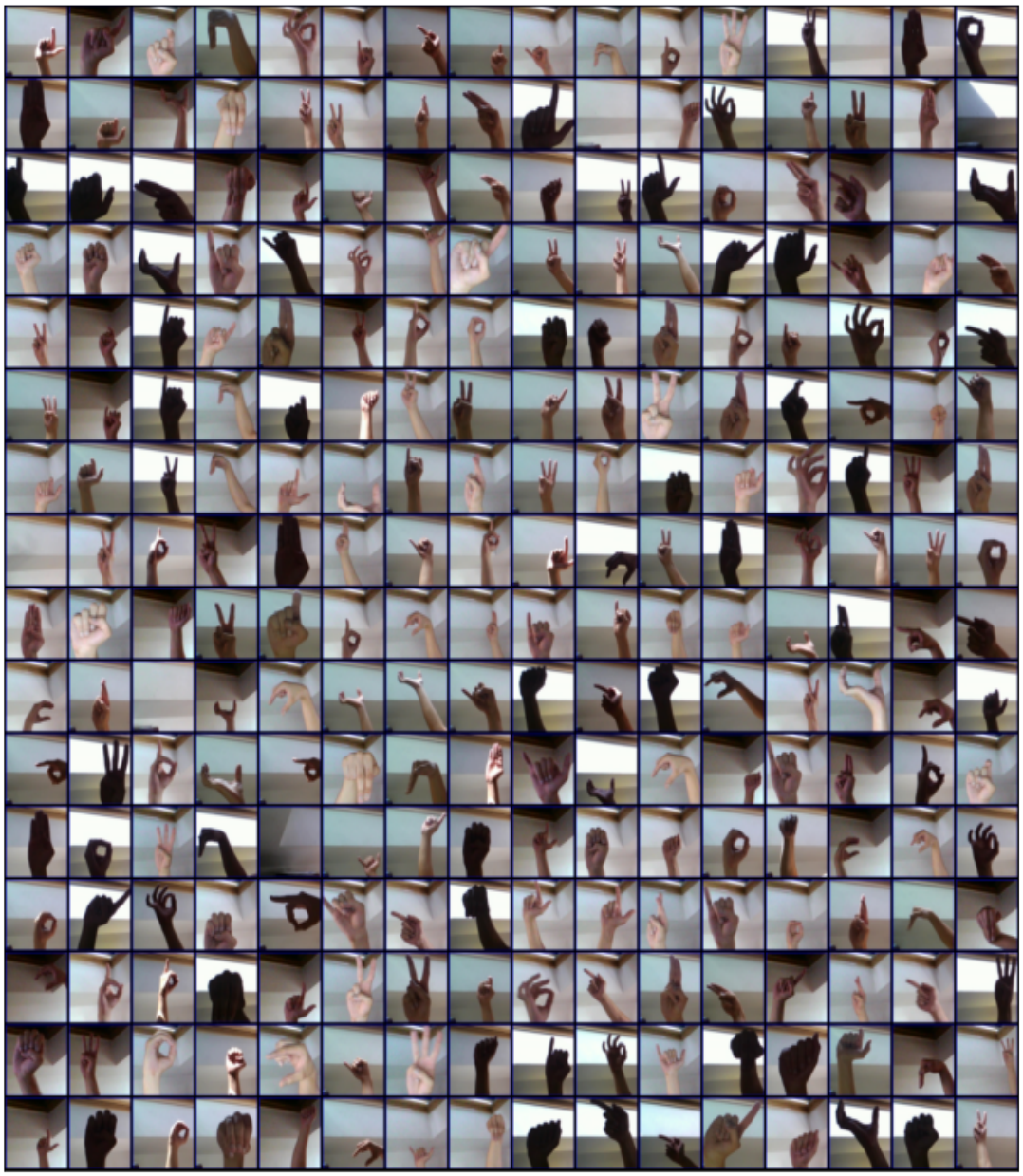


Fig 1.2 Overview of Image Dataset of ASL Alphabet

2. LITERATURE SURVEY

Hearing impairment is a condition in which a person's hearing is impaired and they are unable to hear, whereas mutism is a condition in which a person's speech is impaired and they are unable to speak. Both are merely hearing and/or speech impaired, therefore they can still perform a lot of other things. Communication is the single thing that separates them from ordinary people. If normal people and deaf-mute people can communicate, the deaf-mute people can simply live their lives as regular people. Their sole means of communication is through sign language.

Sign language, a visual-gestural language, is used by deaf and hard-of-hearing people to communicate. Three-dimensional locations and hand gestures (along with other body parts) are used to convey messages. It has a completely different vocabulary and grammar than spoken or written languages. In order to express meaning, spoken languages rely on oratory talents to make sounds that are related to certain words and grammatical combinations. The oratory components are subsequently absorbed and processed suitably by the auditory faculties. Unlike verbal communication, sign language makes use of visual abilities. To construct comprehensive messages, spoken language employs rules; similarly, sign language is guided by a complicated grammar.

Despite the fact that sign language is critical to deaf-mute people's capacity to communicate with each other and with themselves, it receives little widespread attention. We, as normal people, tend to ignore the importance of sign language until we have family and friends who are deaf-mute. One method of communicating with deaf-mute people is to hire a sign language interpreter. However, using a sign language interpreter might be expensive. For the deaf-mute and normal individuals to converse regularly, a low-cost solution is necessary.

A sign language recognition system is a system that converts sign language into text or voice in a simple, efficient, and accurate manner. To detect the alphabet flow and translate sign language words and phrases, computerized digital image processing and a number of classification algorithms were applied. Hand motions, head posture, and body parts can all be used to express information in sign language. Gesture modelling, gesture analysis, gesture recognition, and gesture-based application systems are the four basic components of a gesture recognition system.

2.1 Existing System

Most of the previous techniques implemented were done using image processing heavily. They included filter masking, color co-occurrence method, gaussian blur. The continuation of this image processing techniques was done by applying various machine learning and deep learning algorithms with these processed images as input. The variety in hand gestures had taken deep toll on such implementation when they ran in real-time. Due to complexity and 3-Dimentional nature of ASL hand poses, the image processing techniques could only perform on some of the alphabet. Hence leaving a huge gap of improvement for the remaining alphabet. This limited the research in image-based analysis and moved on to gesture based. Where the inputs were more complicated and collected through usage of various sensors detecting movement of each and every finger precisely. However, this isn't feasible in the vast application since the setup is rather more complex than a simple web cam feed.

2.2 Related Work

Visual Interpretation of Hand Gestures for Human-Computer Interaction (HCI):

For sign language recognition, a variety of vision- and sensor-based approaches have been utilized. The visual interpretation of hand movements for Human-Computer Interaction was explored by Pavlovic et al. The advantages and disadvantages, as well as significant changes in gesture interpretation methodologies depending on whether a 3D model of the human hand or an image appearance model of the human hand is utilized, are highlighted in a 1997 article. 3D hand models offered a manner of more complex modelling of hand motions at the time this study was conducted, but also posed computational challenges that had not been addressed given the real-time requirements of HCI. They also spoke about existing gestural systems and other possible uses for vision-based gesture recognition.

Towards Interpreting Robotic System for Finger spelling Recognition in Real-Time. HRI Companion (2018):

The study, reported by Kalidolda et al in their article, intended to construct a sign language translating robotic system. The project had two main goals: real-time fingerspelling recognition and performance testing "in the field," as well as getting input on the NAO robotic interpretation system from deaf-mute people. Microsoft Kinect SDK for human identification and tracking, and Leap Motion SDK for hands detection and tracking, are among the software and hardware components of the robotic system. A stationary humanoid NAO robot that serves as a social interface. NAO may also act as a robotic instructor for youngsters. Pepper, a humanoid stationary robot that serves as a social interface, for teaching apps, a computer display and an Android tablet are utilized.

Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images (2021):

In this research, image classification and a machine learning method are utilized to recognize sign language. Convolutional neural networks are used in machine learning. The image collection for this study was made up of static sign language motions captured by an RGB camera. The dataset consisted of static sign movements used to represent alphabets and integers, which were effectively identified. The suggested neural network in this study incorporates neurons from several levels that are independently linked to each other. If a picture has 256 × 256 x 3 pixels with values ranging from 0 to 255, there is a lot of data to analyze. Because all pixels do not carry relevant information, a convolutional neural network is utilized instead. Alternatively, when the picture is supplied to the next layer, convolutional neural networks convolve around the input pixels, reducing the image's size.

Nearest Neighbor Classification of Indian Sign Language Gestures using Kinect Camera (2016):

PCA is used by Divya Deora and Nikesh Bajaj to recognize sign symbols (Principal Component analysis). The article also suggests using neural networks for recognition. The data was collected with a 3 mega pixel camera; hence the quality was low. They collected 15 photos for each sign and saved them in their database. One of the reasons for their unsatisfactory results was the tiny dataset. They segmented the RGB and divided it into components, as well as doing a rudimentary border pixel analysis. They claim that while combining the fingertip method with PCA produced a good result, neural networks can produce a superior outcome.

Sign Language Recognition using Convolutional Neural Networks (2015):

Lionel et al used a Convolutional Neural Network to recognize sign language (CNN). Two phases were completed to automate the process of sign language recognition: feature extraction and action categorization. The first phase is carried out with a CNN, while the second step is carried out with an Artificial Neural Network. The dataset contains a total of 20 distinctive Italian gestures signed by 27 persons in various environments. Microsoft Kinect is used to record the videos. A total of 6600 photos were utilized for development, with 4600 being used for training and the remainder being used for validation. The photographs were cropped in post-production to remove the upper section of the hand and body. Only one side of each motion has to be learned by the model. For feature extraction and classification, max pooling is utilized. It is made up of two CNNs, the output of which is fed into an ANN. The output of the two CNNs is combined by the ANN. A CPU was utilized to supplement the data, while a GPU was used to train the model.

2.3 Proposed System

The goal of the research is to use a live visual stream to detect and categories the ASL alphabet in real time. To speed up the procedure even further, a variety of mathematical and scientific approaches are used. Because this project works with picture data, data pre-processing is the most difficult part of the job. The approach was methodically carried out using priceless methodologies in order to remove as many edge situations as feasible. We started by taking 3,000 photos of real-time hand positions for each letter of the alphabet. In all, 78,000 photos were included in the raw dataset. These photos have all been filtered using a variety of methods in order to make the dataset more efficient and cleaner. The preprocessing is done by extraction of hand landmarks. These hand landmarks are the coordinate points on a palm. They are hard knuckles of the hand and the wrist bottom. These are extracted using Mediapipe framework developed for hand tracking. The coordinates are then normalized with respect to one the wrist landmark so that all data is uniform and scalable to fit different perspectives of same hand pose. This information is incorporated into a neural network architecture that aims to optimize model accuracy while lowering loss to an acceptable level. All of this effort comes together to let us create an application that identifies each of the 26 ASL alphabets in real time.

3. SYSTEM ANALYSIS

This analysis helps the analyst to make right decisions and suggest a course of action to achieve better results. In this analysis the system functions will be broken down into pieces to analyze the scenario, and to analyze the aim of the project. This step also involves analyzing what needs to be done and by assigning the users to define the requirements.

3.1 Functional Requirements

Functional requirements in software describes the way the system should work. A function is a combination of input, behavior and output. These requirements which manage the outline or execution. Functional requirements are “framework must do <requirement>” whereas on the other hand non-functional requirements are “framework should be <requirements>”. Functional requirements manage framework outline and help in the application design of a framework whereas non-functional requirements manage framework engineering and drive the specialized engineering of a framework.

3.2 Performance Requirements:

These requirements are one of the important ones among the non functional requirements. They are to be listed down clearly by questioning the users and knowing their requirements in fine detail. It gives a description of the expected functionality of the system by the users. Some of them include throughput, speed etc. These play a vital role in both the design phase and the test phase of the product.

3.3 Software Requirements

- Operating System - Windows 10, linux, MacOS
- Anaconda Distribution or GoogleColab

- Python 3.0+
- Libraries- openCV, mediapipe, scipy, numpy, sklearn, tensorflow

To run a software, it should satisfy minimum system requirements. For accomplishment of this project, we used ANACONDA Distribution. In Anaconda Distribution, either Jupyter Notebooks or Spyder can be used. If for some reason the Jupyter Notebook is not responsive then in such cases Google Collaboratory can be used which is a web hosted VM.

Jupyter Notebook:

Jupyter notebook or formerly known as 'Python notebooks comes with Anaconda Distribution or even can be installed Individually. Jupyter notebooks is a web based interactive computational interface which supports various languages such as Python, Ruby, R, Julia etc. Jupyter notebooks can be converted to various standard output formats such as HTML, pdf, Python files etc.

OpenCV:

In order to create Computer Vision applications which are real time applications we use the openCV library. OpenCV library has at least 2500 algorithms included for computer vision and machine learning techniques. These techniques may distinguish plant images, monitor their behaviour etc.

Scipy:

Scipy has a number of sub-packages that aid with the most prevalent problems in Scientific computation. It is an open-source library and easy to understand. The users can manipulate and visualize the data using python commands.

Numpy:

Numpy is widely used in Data Science and performs basic mathematical calculations. It contains array data type and has sorting techniques, indexing etc. With the extension of numpy, scipy is built.

Mediapipe:

Hands by MediaPipe is a high-resolution hand and finger tracking system. From a single image, machine learning (ML) is utilised to derive 21 3D landmarks of a hand. While previous state-of-the-art systems rely on sophisticated desktop environments for inference, our technology achieves real-time performance on a mobile phone and is even scalable to many hands.

Sklearn:

Simple and effective tools for analysing predictive data It is open to everyone and may be utilised in a number of contexts. This was made using NumPy, SciPy, and Matplotlib. Open source, BSD licence, and commercially useful.

Tensorflow:

TensorFlow is an open-source machine learning framework that automates the whole process. It has a broad, flexible ecosystem of tools, libraries, and community resources that enable academics to improve the state-of-the-art in machine learning and developers to swiftly build and deploy machine learning applications.

3.4 Hardware Requirements

- Processor - Intel core i5
- Speed - 1.6 GHz
- Disk Space - 400 GB - (for faster processing of data)
- RAM - at least 8 GB
- GPU Acceleration (Optional)

4. FEASIBILITY STUDY

In this phase, the project's feasibility is assessed, and a business proposal is presented, along with a very generic project design and some cost estimates. A feasibility assessment of the proposed system is to be carried out during system analysis. This is to guarantee that the planned system will not cause the organisation any problems. A basic grasp of the system's primary needs is required for feasibility analysis.

The feasibility analysis takes into account three major factors.

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

4.1 ECONOMICAL FEASIBILITY

This research is being carried out to determine the system's economic impact on the organisation. The amount of money the corporation has to invest in the system's research and development is restricted. It is necessary to justify the spending. As a result, the created system came in under budget, which was made possible by the fact that the majority of the technologies employed were publicly accessible. The customised items were the only ones that needed to be acquired.

4.2 TECHNICAL FEASIBILITY

This research is being carried out to determine the system's technological feasibility, or technical needs. Any system that is created should not place a large burden on the available technological resources. As a result, there will be a lot of pressure on the existing technological resources. As a result, the client will be subjected to severe demands. Because very minor or no changes are necessary to deploy this system, the designed system must have a low requirement.

4.3 SOCIAL FEASIBILITY

The goal of the study is to establish the system's level of acceptability among users. This refers to the process of training a user how to utilise technology successfully. The user should not fear the system, but rather accept it as a need. The methods utilised to educate and familiarise users with the system are solely accountable for the level of user acceptance. Because he is the system's final user, his self-esteem must be reinforced in order for him to provide constructive criticism, which is encouraged.

5. SYSTEM DESIGN

System design process involves identification and defining of components, modules, interfaces, data of a system and finally the architecture. The main goal of the system design is to satisfy the requirements specified and design the system in such a way. In a sentence, it can be described as application of systems theory to product development. Most widely used methods for systems design being object-oriented analysis and design.

5.1 Data Flow Diagram

Data flow diagram depicts the flow of a data stream in a system and various information sources and their flows. Generally, a data flow diagram is used to represent how the data is being processed or its flow, how data is stored, controlled and manipulated within many states/streams. The particular design depicts the flow of data from storage to processed output analytics involving various processing stages.

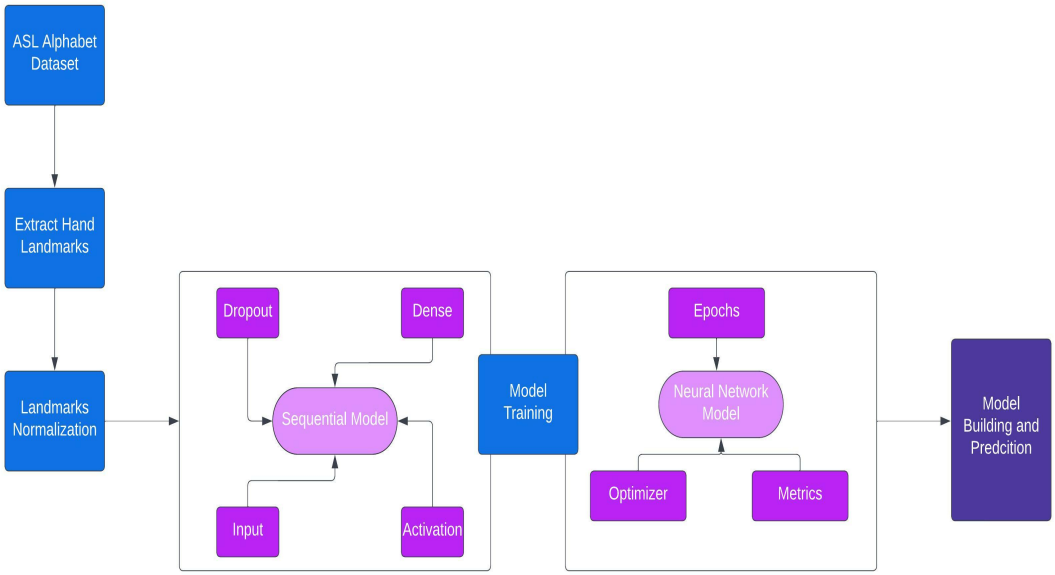


Fig 5.1 Data Flow Diagram

Steps for the Data Flow Diagram:

- In the first step the images are extracted from the ASL alphabet dataset which consists of 3,000 per alphabet.
- The hand landmarks are extracted from each image in the dataset.
- Then these coordinates are processed using mathematical techniques which is normalization.
- These normalized coordinate data is stored as csv data file which can read using csv library and prepared for feeding into the model.
- The neural network model is designed i.e., all layers are added according to extract maximum accuracy and also without overfitting.
- Then the model is trained with parameters like epochs, optimizer & metrics as parameters.
- The model is trained with different parameters to optimize on the model accuracy and model loss.
- This model is converted as tensorflow lite object so that it used in our application without retraining.
- The tensorflow lite module saved, is used to classify the ASL alphabet in real-time.

5.2 UML DIAGRAMS

The acronym for "Unified Modeling Language" is UML. UML is a standardised general-purpose modelling language in the field of object-oriented software engineering. The standard was developed by the Object Management Group, which is in charge of it. The goal is for UML to become the industry standard for creating object-oriented software models. A meta-model and a notation are the two main components of UML in its current form. In the future, UML may be extended to include or be linked to some form of method or process. The Unified Modeling Language (UML) is a standard for defining,

visualising, building, and documenting software system artefacts, as well as business modelling and non-software systems.

The Unified Modeling Language (UML) is a set of tried-and-true engineering practises for modelling large, complex systems. The UML is an important part of the software development process and object-oriented software development. The UML primarily uses graphical notations to explain the design of software projects.

The following are the primary aims of the UML design:

- Users should be able to construct and exchange meaningful models using a ready-to-use, expressive visual modelling language.
- To expand the essential principles, provide tools for extendibility and specialization.
- Be unconstrained by programming languages or development processes.
- Establish a rigorous foundation for comprehending the modelling language.
- Encourage the market for OO tools to expand.
- Higher-level development ideas like collaborations, frameworks, patterns, and components should be supported.
- Best practises should be included.

5.2.1 Use Case Diagram

In the Unified Modeling Language, a use case diagram is a type of behavioural diagram that is described by and produced from a Use-case analysis (UML). Its purpose is to provide a visual representation of a system's functionality in terms of actors, goals (represented as use cases), and any links between use cases. The main goal of a use case diagram is to show which system functions are executed for each actor. It is possible to display the roles of the system's actors.

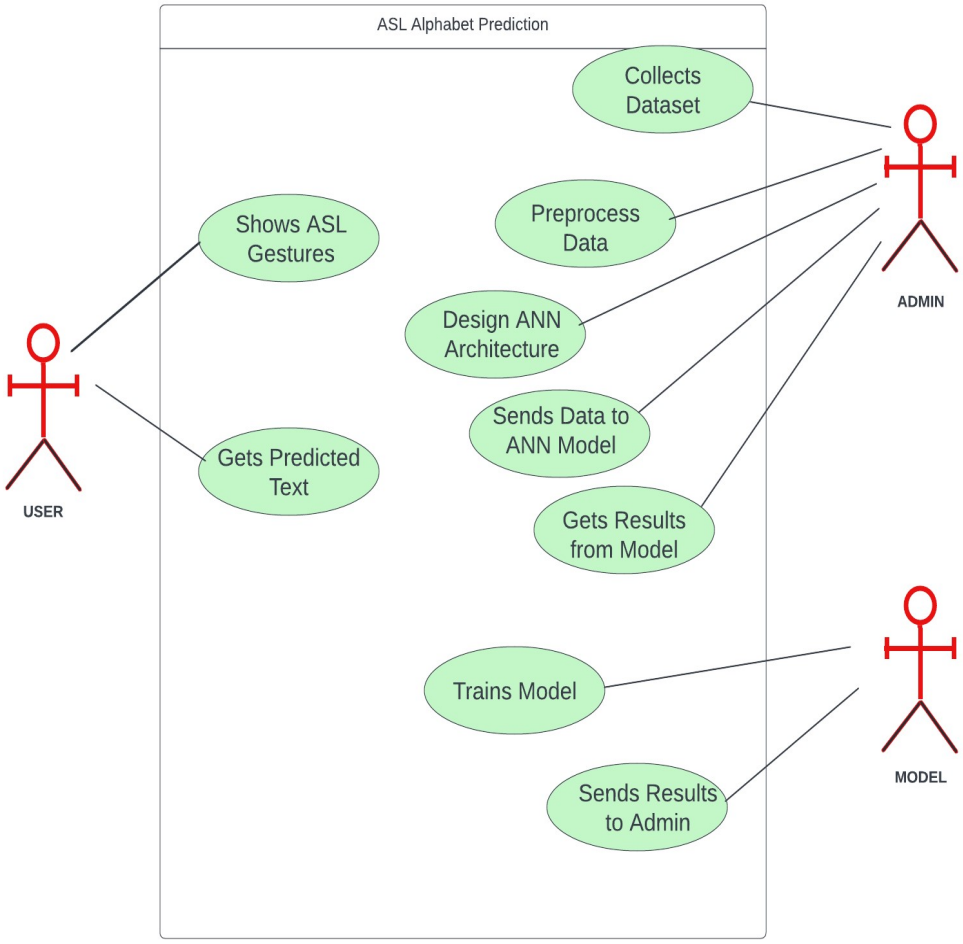


Fig 5.2 Use Case Diagram

5.2.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) displays the structure of a system by presenting the system's classes, characteristics, actions (or methods), and links between the classes. It specifies the type of data that is stored in which class.

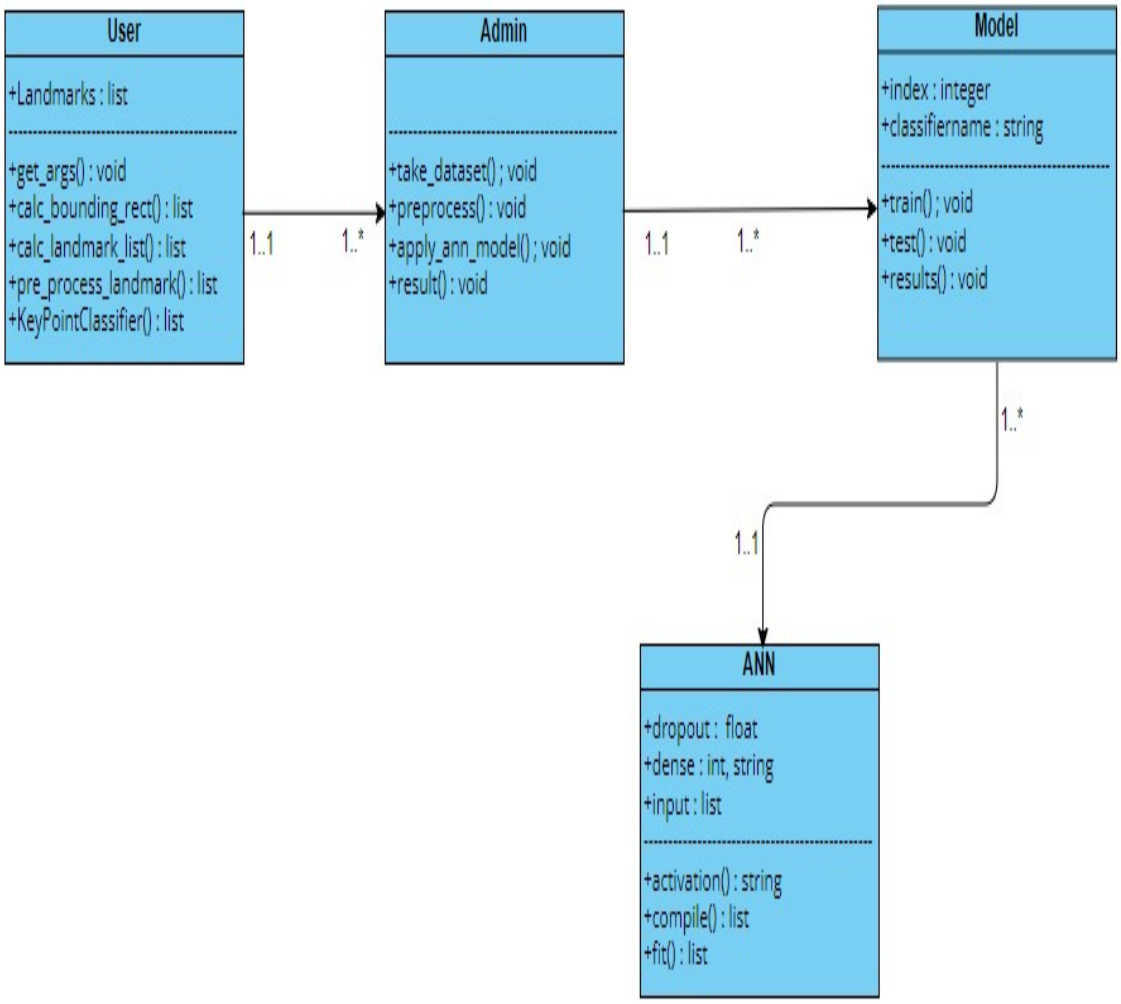


Fig 5.3 Class Diagram

5.2.3 Sequence Diagram

Diagrams of Sequences Represent the items involved in the interaction on a horizontal plane and the time on a vertical plane. A Use Case is a form of behavioural classifier that reflects the statement of a certain behaviour. Each use case defines a single task that can include or exclude modifications that the subject can do in collaboration with one or more

actors. Use cases describe the subject's offered behaviour without reference to the subject's internal structure. These interactions, which involve the actor and the subject, may result in changes in the subject's state and communication with its environment. Exceptional behaviour and error handling are examples of possible deviations of a use case's fundamental behaviour.

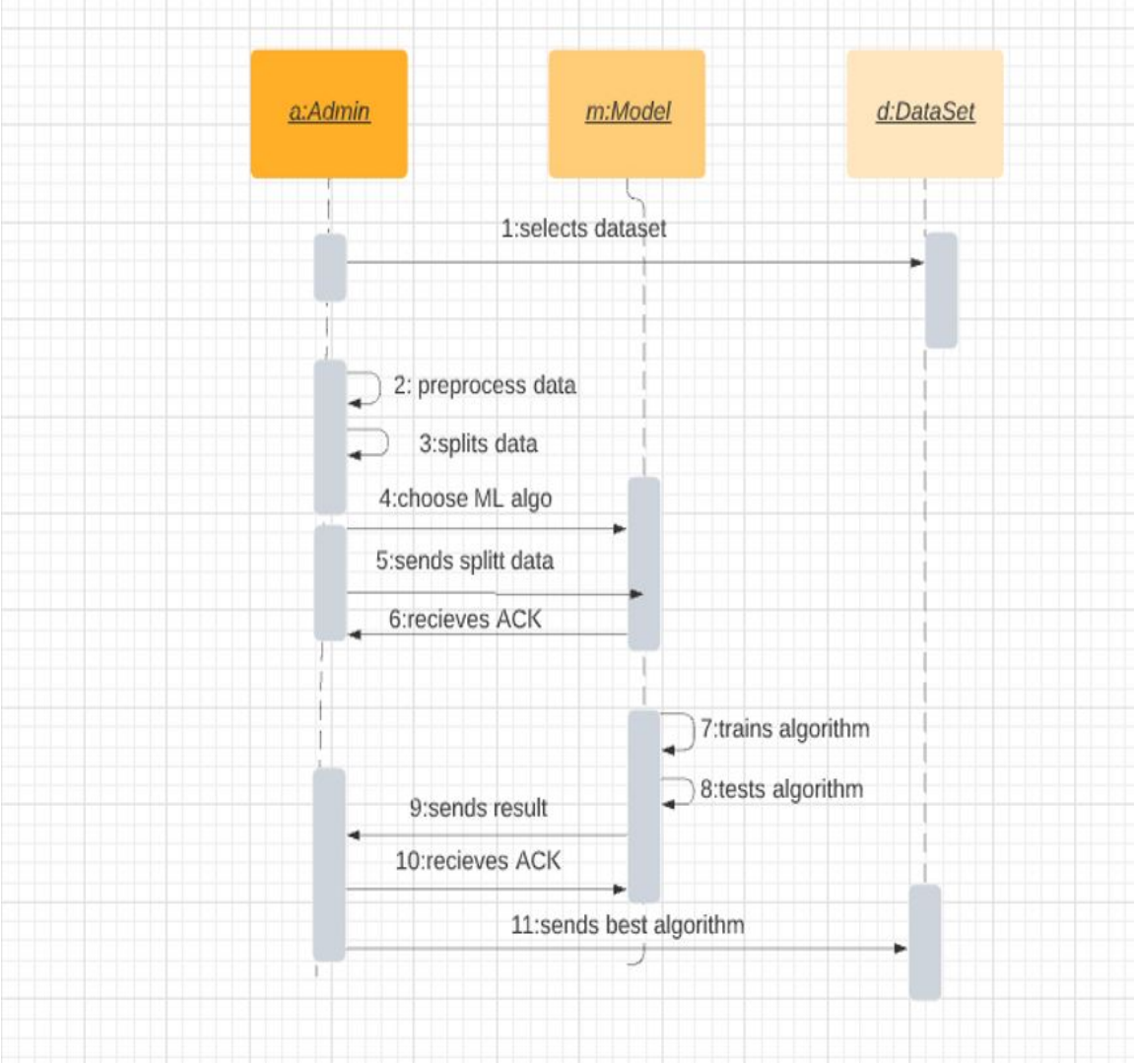


Fig 5.4 Sequence Diagram

5.2.4 Activity Diagram

Activity diagrams graphically illustrate workflows that consist of sequential activities and actions with options for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams may be used to illustrate the business and operational step-by-step processes of system components. The whole control flow is depicted in an activity diagram.

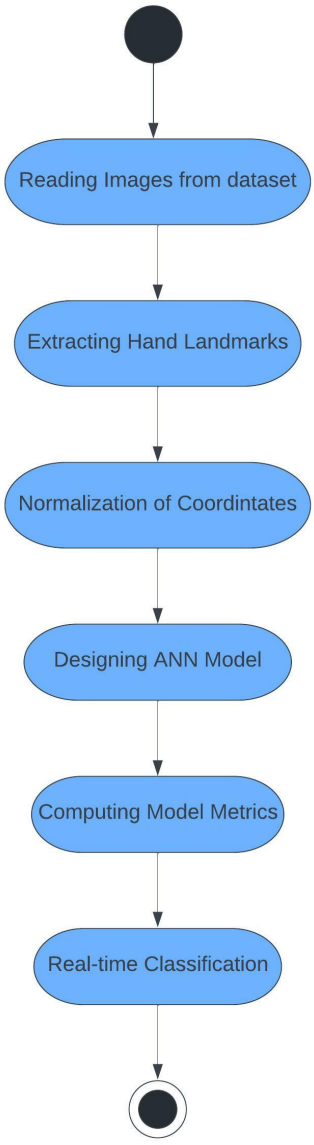


Fig 5.5 Activity Diagram

5.2.5 Component Diagram

In terms of nature and behaviour, component diagrams differ. Component diagrams are used to represent a system's physical components. The issue now is, what are these bodily characteristics? Physical aspects are the items that live in a node, such as executables, libraries, files, documents, and so on. Component diagrams are used to show how a system's components are organised and related. These diagrams are also used to create systems that can be executed.

Purpose of Component Diagrams

A component diagram is a form of diagram in the UML language. The aim is also separate from all of the other diagrams shown thus far. It does not explain how the system works; rather, it explains the components that allow it to do so.

As a result, component diagrams are used to visualise the physical components of a system from that point of view. These components include libraries, packages, files, and so on.

A component diagram can be used to represent a static implementation perspective of a system. Static implementation refers to the grouping of components at a specific point in time.

Even though a single component diagram cannot describe the entire system, a succession of diagrams is used instead.

The component diagram's purpose may be described as follows:

- Visualize the system's components.
- Forward and reverse engineering are used to create executables.
- Describe the components arrangement and connections.

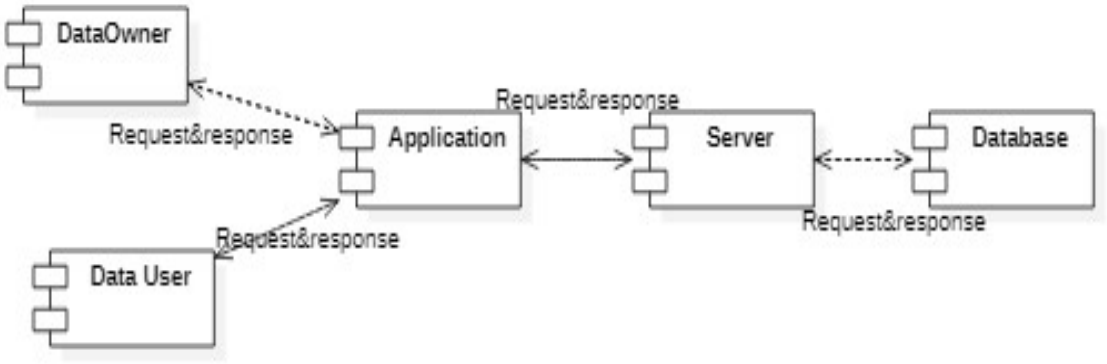


Fig 5.6 Component Diagram

5.2.6 Deployment Diagram

There may be more steps involved, depending on what specific requirements you have, but below are some of the main steps. It visualizes the physical hardware on which the software will be deployed. It involves nodes and relationships.

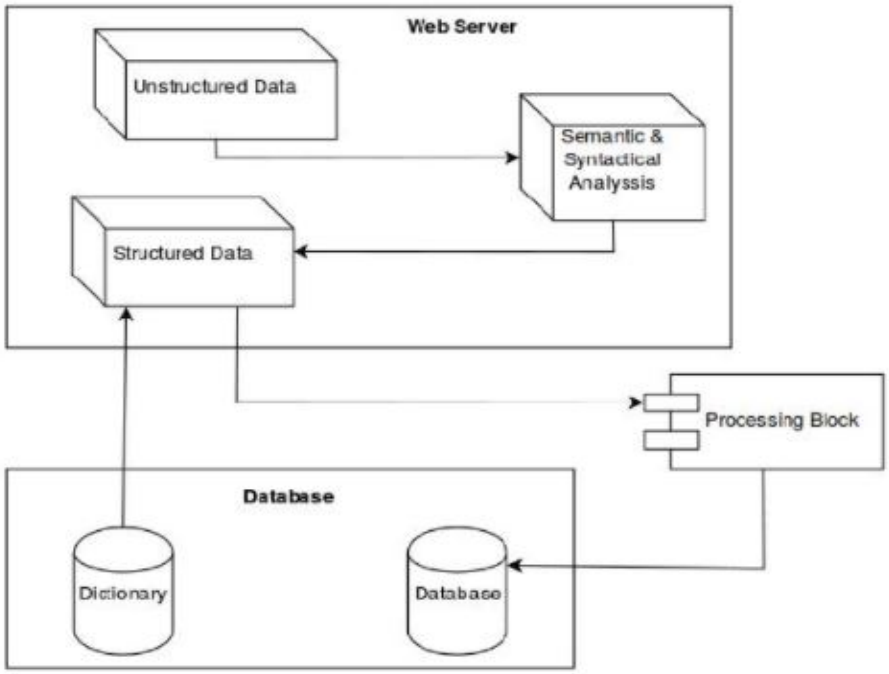


Fig 5.7 Deployment Diagram

5.3 Input Design and Output Design

5.3.1 Input Design

The input design is the link between the information system and the user. It comprises developing data preparation requirements and processes, as well as the actions necessary to convert transaction data into a processable format. This can be done by looking at the computer to see if it can read data from a written or printed document, or by having users manually enter the data into the system. The purpose of input design is to reduce the amount of input required, control errors, reduce delays, eliminate unnecessary phases, and simplify the process. The input is designed to provide security and convenience while still retaining privacy. Input Design took into mind the following factors:

- What information should be provided as input?
- How should the data be organised or coded?
- The dialogue will serve as a guide for the operational people when it comes to delivering input.
- Methods for creating input validations, as well as what to do if an error occurs.

5.3.2 Objectives

1. Make a suggestion. Design refers to the process of converting a user-oriented description of an input into a computer-based system. This design is essential for preventing data input mistakes and directing management in the proper direction for obtaining reliable data from the computerised system.

2. It's done by creating user-friendly data entry screens that can handle massive volumes of information. The goal of input design is to make data entry as simple as possible while avoiding errors. The data entry panel is set up so that you may do any data modifications. You may also look at your records using it.

3. The info will be checked when it has been input. Information can be entered on screens. When necessary, appropriate notifications are sent, ensuring that the user is never caught off guard. As a result, the purpose of input design is to create an input layout that is simple to understand.

5.3.3 Output Design

A high-quality output meets the demands of the end user and displays information clearly. The outcomes of any system's processing are communicated to users and other systems via outputs. In output design, it is chosen how the information will be displaced for immediate usage as well as the hard copy output. It is the most important and direct source of information for the user. Through efficient and intelligent output design, the system's interface with the user is improved.

1. Computer output should be developed in a methodical, well-thought-out manner, with each output component constructed in such a way that users would find the system easy to use and effective. While analysing and designing computer output, they should identify the specific output that is necessary to meet the criterion.
2. Decide on how you'll deliver the data.
3. Create a paper, report, or other format that contains the system's information.

An information system's output form should achieve one or more of the following goals.

- Communicate information regarding past activity, present condition, or future estimates.
- Important events, opportunities, difficulties, or cautions are signalled.
- An action is triggered.
- An action is confirmed.

6. IMPLIMENTATION AND RESULTS

6.1 Language / Technology Used:

Python language is used to write the code. Python provides a wide variety of libraries for scientific and computational usage. In a vision-based sign language recognition system, we applied an appearance-based method. It allows for the use of photos as input. They deduce straight from the photographs or videos. Unlike the 3-D-based technique, which relied on 3-D knowledge on major body parts, this approach does not require any spatial representation of the body. Libraries such as opencv, os, numpy, skimage, math, scipy, pandas etc are used for image processing techniques such as filtering and segmentation. The main functionalities in the project are

1. **Data Augmentation:** Increase the data actificially by modifying the orientation (angle, degree, rotate etc) of images in the collected data.
2. **Image Conversion:** Applying mediapipe framework to images to exrtract hand landmarks.
3. **Neural Network Architecture:** Designing appropriate ANN model to train with the coordinate dataset.
4. **Real-time Recognition:** The trained model is used to classify the 26 ASL alphabet posed by human hand in real-time.

The libraries or packages used for the above functionalities are:

Data Augmentation:

- from keras preprocessing.image we import Image Data Generator
- opencv library is used to capture the images
- os libraries are used to read the labels, rename the images and store them

- PIL (python image library) is used to manipulate the images

Image Conversion:

- os libraries are used to read the labels, rename the images and store them
- mediapipe library is used to extract hand landmarks from the images read.
- opencv library is used for image processing
- os library are used to retrieve images and perform operations on them
- matplotlib and seaborn are data visualization libraries.
- numpy is a library used for numerical calculations for image pixel data manipulation
- pandas is a data management library

Neural Network Architecture:

- numpy is a library used for numerical calculations for image pixel data manipulation
- tensorflow library is used for constructing the ANN model and to train it
- matplotlib and seaborn are data visualization libraries
- sklearn is a library which is used for machine learning and statistical modelling of data

Real-Time Recognition:

- mediapipe library is used to extract hand landmarks from the images read
- utils library is used to for real-time feed analysis and information
- collections is used for importing the data structures like queues and counter

- opencv is used to initiate the visual feed to real-time recognition and classification

6.2 Methods / Algorithms used

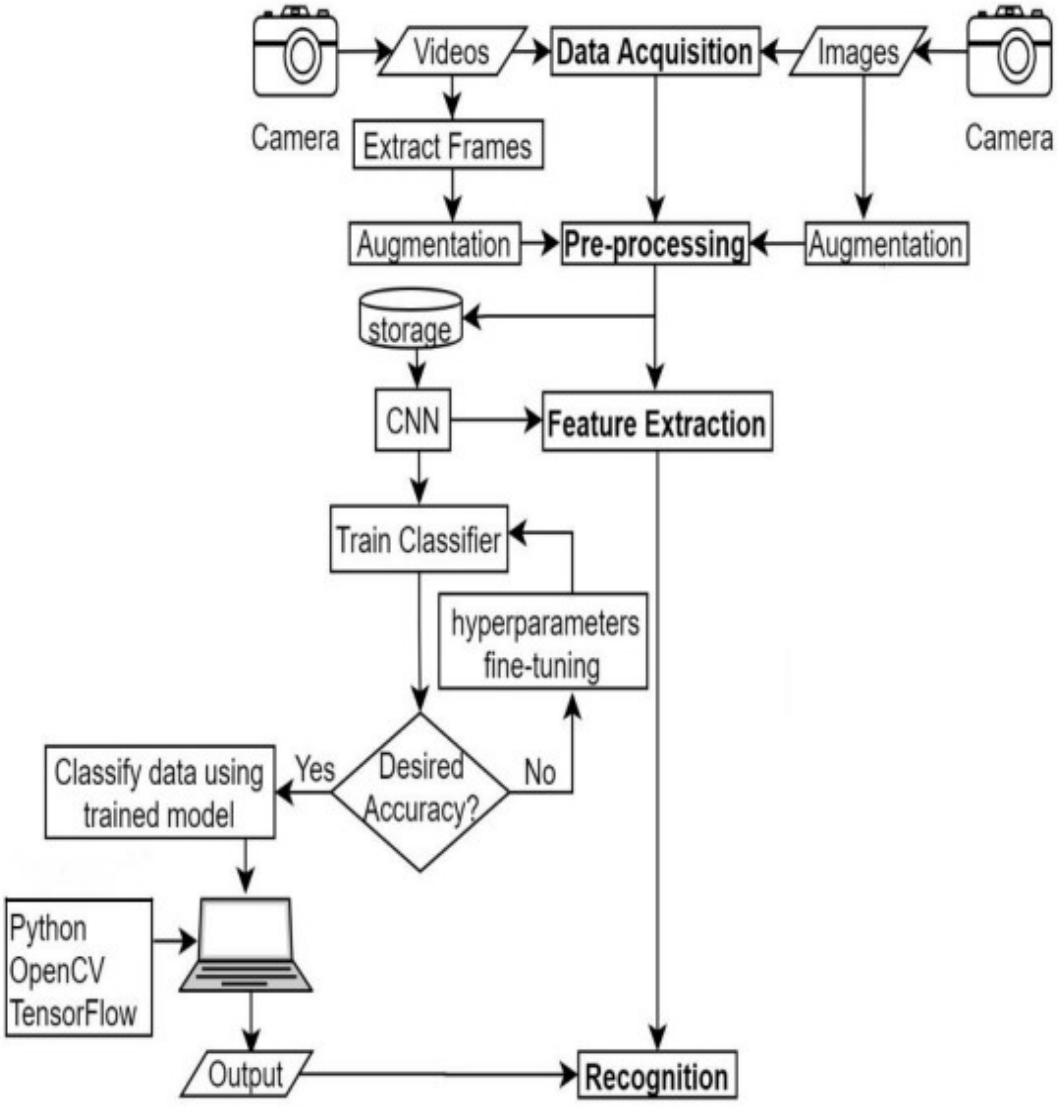
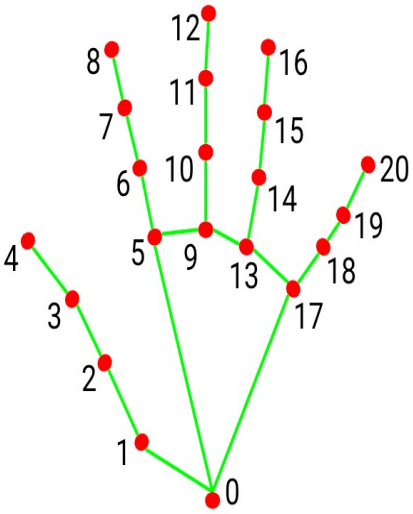


Fig 6.1 Project Architecture

6.2.1 Mediapipe

Recognition of hand shape and motion can assist enhance user experience across a wide range of technological areas and platforms. It may be used to understand sign language and regulate hand movements, as well as to overlay digital content and information on top of the physical world in augmented reality. Because hands regularly occlude themselves or one other (for example, finger/palm occlusions and hand shaking), and because they lack high contrast patterns, robust real-time hand perception is a tough computer vision challenge.

Hands is a high-resolution hand and finger tracking device by MediaPipe. From a single image, machine learning (ML) is utilised to derive 21 3D landmarks of a hand. While previous state-of-the-art systems rely on sophisticated desktop environments for inference, our technology achieves real-time performance on a mobile phone and is even scalable to many hands. We expect that by making these hand sensing capabilities available to the rest of the research and development community, new applications and research paths will emerge, sparking new applications and research.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fig 6.2 Hand Landmarks Tracked by Mediapipe Module

Following palm identification over the whole image, our next hand landmark model use regression to precisely localise 21 3D hand-knuckle coordinates inside the detected hand regions, i.e. direct coordinate prediction. The model produces a consistent internal hand posture representation even with partially visible hands and self-occlusions.

To obtain ground truth data, researchers manually labelled 30K real-life photographs with 21 3D locations, as seen below (we take Z-value from image depth map, if it exists per corresponding coordinate). In order to better cover the available hand positions and offer further supervision on the nature of hand geometry, we also construct a high-quality synthetic hand model over various backgrounds and map it to the relevant 3D coordinates.

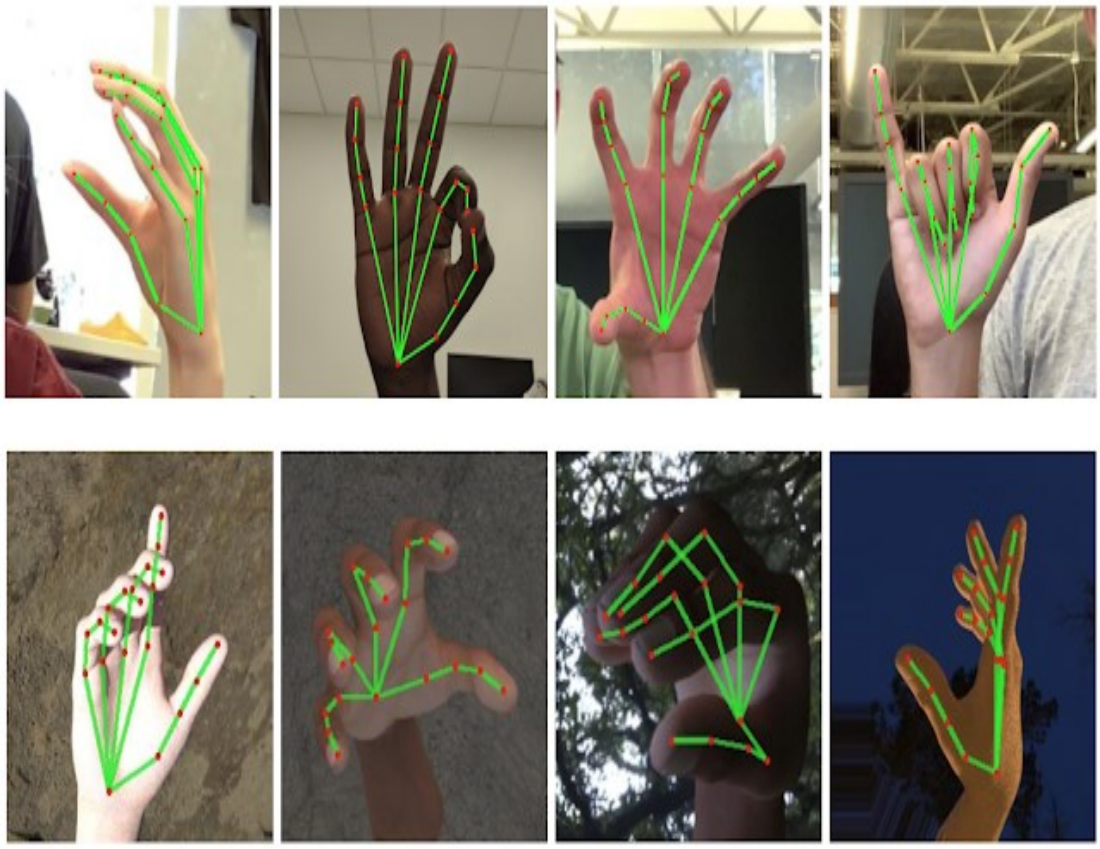


Fig 6.3 Sample Real-Time Hand Tracking Images

6.2.2 Coordinate Normalization

Images that have been normalised are mean-centered and have a unit variance. The maximum distance between normalised picture coordinates is one unit. As a result, the image coordinates produced by the projection matrix are closer to the ground truth normalised coordinates. Residue is a performance metric that we use.

To normalise coordinates, we use pixel values for x and y, which represent the centre of the bounding box on the x- and y-axes, respectively. Then we divide the value of x by the image's width and the value of y by the image's height. The bounding box's width and height are represented by width and height.

```
def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]

        temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
        temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y

    # Convert to a one-dimensional list
    temp_landmark_list = list(
        itertools.chain.from_iterable(temp_landmark_list))

    # Normalization
    max_value = max(list(map(abs, temp_landmark_list)))

    def normalize_(n):
        return n / max_value

    temp_landmark_list = list(map(normalize_, temp_landmark_list))

    return temp_landmark_list
```

Fig 6.4 Code for Normalized Coordinates

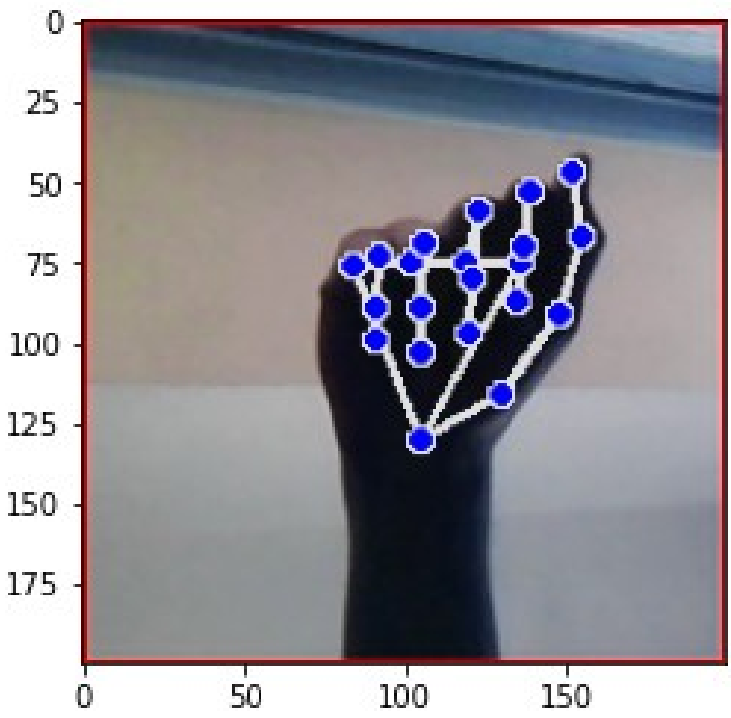


Fig 6.5 21 Hand Landmarks Tracked

Pixel Coordinates (For Fig 6.5):

[[91, 116], [114, 100], [128, 74], [130, 52], [126, 35], [109, 54], [117, 41], [118, 61], [117, 78], [93, 56], [103, 46], [106, 69], [106, 87], [79, 60], [91, 58], [93, 80], [93, 94], [65, 67], [78, 64], [80, 80], [80, 91]]

Normalized Pixel Coordinates (For Fig 6.5):

[0.0, 0.0, 0.2839506172839506, -0.19753086419753085, 0.4567901234567901, -0.5185185185185185, 0.48148148148148145, -0.7901234567901234, 0.43209876543209874, -1.0, 0.2222222222222222, -0.7654320987654321, 0.32098765432098764, -0.9259259259259259, 0.3333333333333333, -0.6790123456790124, 0.32098765432098764, -0.4691358024691358, 0.024691358024691357, -0.7407407407407407, 0.14814814814814814, -0.8641975308641975, 0.18518518518518517, -0.5802469135802469,

0.18518518518518517, -0.35802469135802467, -0.14814814814814, -0.691358024691358, 0.0, -0.7160493827160493, 0.024691358024691357, -0.4444444444444444, 0.024691358024691357, -0.2716049382716049, -0.32098765432098764, -0.6049382716049383, -0.16049382716049382, -0.6419753086419753, -0.13580246913580246, -0.4444444444444444, -0.13580246913580246, -0.30864197530864196]

The pixel coordinates of the 21 hand landmarks obtained are undergone processing through the pre_process_landmark function. The coordinates are iterated through the being input to the normalization equation. The normalized coordinates range from -1 to 1, and are normalized with respect to the wrist hand landmark coordinate.

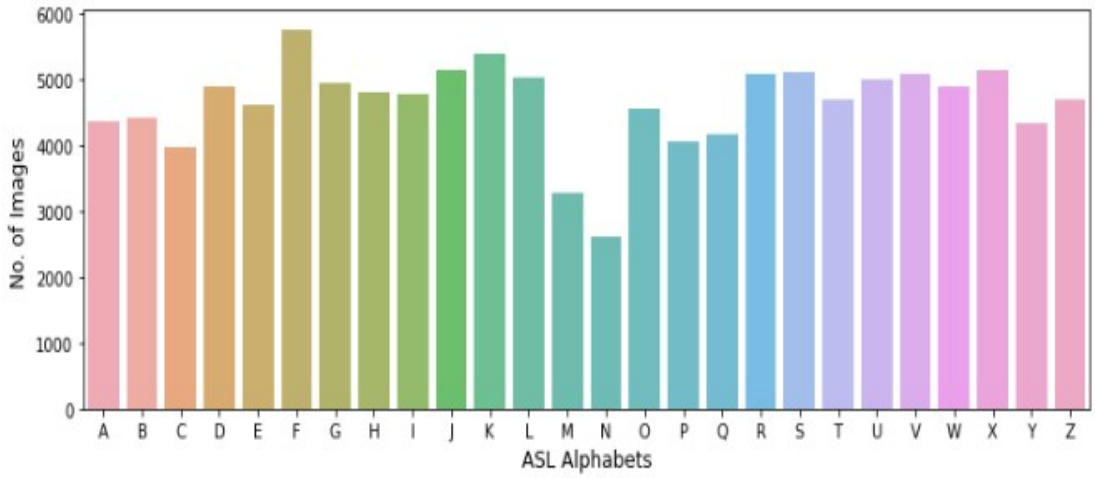


Fig 6.6 Data Distribution

6.2.3 Neural Network Architecture

It is one of the most significant aspects of our network, since it defines the architecture, we require. Below is a simplified explanation of CNN, as well as a diagram of our model architecture summary:

- Begins with an image as an input.
- It creates a feature map using a variety of filters.

- Increases non-linearity by using a RELU function.
- Adds a pooling layer to each subsequent map.
- Combining all of the pooled photos into a single long vector.
- Feeds the vector into a fully linked artificial neural network.
- Dropout can also be employed to reduce overfitting.
- The voting of the classes is provided by the final fully linked layer.
- Trains for multiple epochs using forward propagation and backpropagation.

Model building

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((21 * 2, )),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(45, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(15, activation='relu'),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])
```

```
model.summary()  # tf.keras.utils.plot_model(model, show_shapes=True)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout_9 (Dropout)         (None, 42)                0

 dense_12 (Dense)            (None, 45)                1935

 dropout_10 (Dropout)        (None, 45)                0

 dense_13 (Dense)            (None, 30)                1380

 dropout_11 (Dropout)        (None, 30)                0

 dense_14 (Dense)            (None, 15)                465

 dense_15 (Dense)            (None, 26)                416

=================================================================
Total params: 4,196
Trainable params: 4,196
Non-trainable params: 0
_________________________________________________________________
```

Fig 6.7 Neural Network Model Architecture

After defining the model, the following step is to compile it. Model compilation is done with Tensorflow. The process of compiling is the setting of parameters for model training

and predictions. In the background, the CPU/GPU or distributed memory can be utilised. We must define a loss function that will be used to calculate weights for the various layers. The optimiser changes the learning rate and cycles through different weight sets. The loss function in this situation will be Binary Cross Entropy. We will utilise ADAM, which is an effective stochastic gradient descent technique, as an optimizer.

```
# Model compilation
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Fig 6.8 Model Compilation

Model training is the process of fitting the model. The model is now ready to run through the data and train itself when it has been compiled. The Keras fit() method may be utilised in the model training process. The following are the two primary parameters that are used prior to model training:

Epochs: A single iteration across the whole dataset.

Batch Size: At each batch size, the weights are updated. Epochs are made up of batches of data that are evenly dispersed.

Model training

```
abc = model.fit(
    X_train,
    y_train,
    epochs=10000,
    batch_size=128,
    validation_data=(X_test, y_test),
)
```

Fig 6.9 Model Fitting

In this procedure, a GPU or a CPU is employed. Depending on the epochs, batch size, and most crucially, the quantity of the data, training might take a long time. The evaluate() method may also be used to test the model on the training dataset. The data may be separated into training and testing sets, with testing X and Y being used to evaluate the model.

This will generate a prediction and gather scores for each input and output combination, Including the average loss and any measures we have placed, such as accuracy. The evaluate() method will return a list of two values. The first will be the dataset's model loss, and the second will be the dataset's model accuracy. We're simply concerned with the report's accuracy, therefore we'll overlook the significance of the loss.

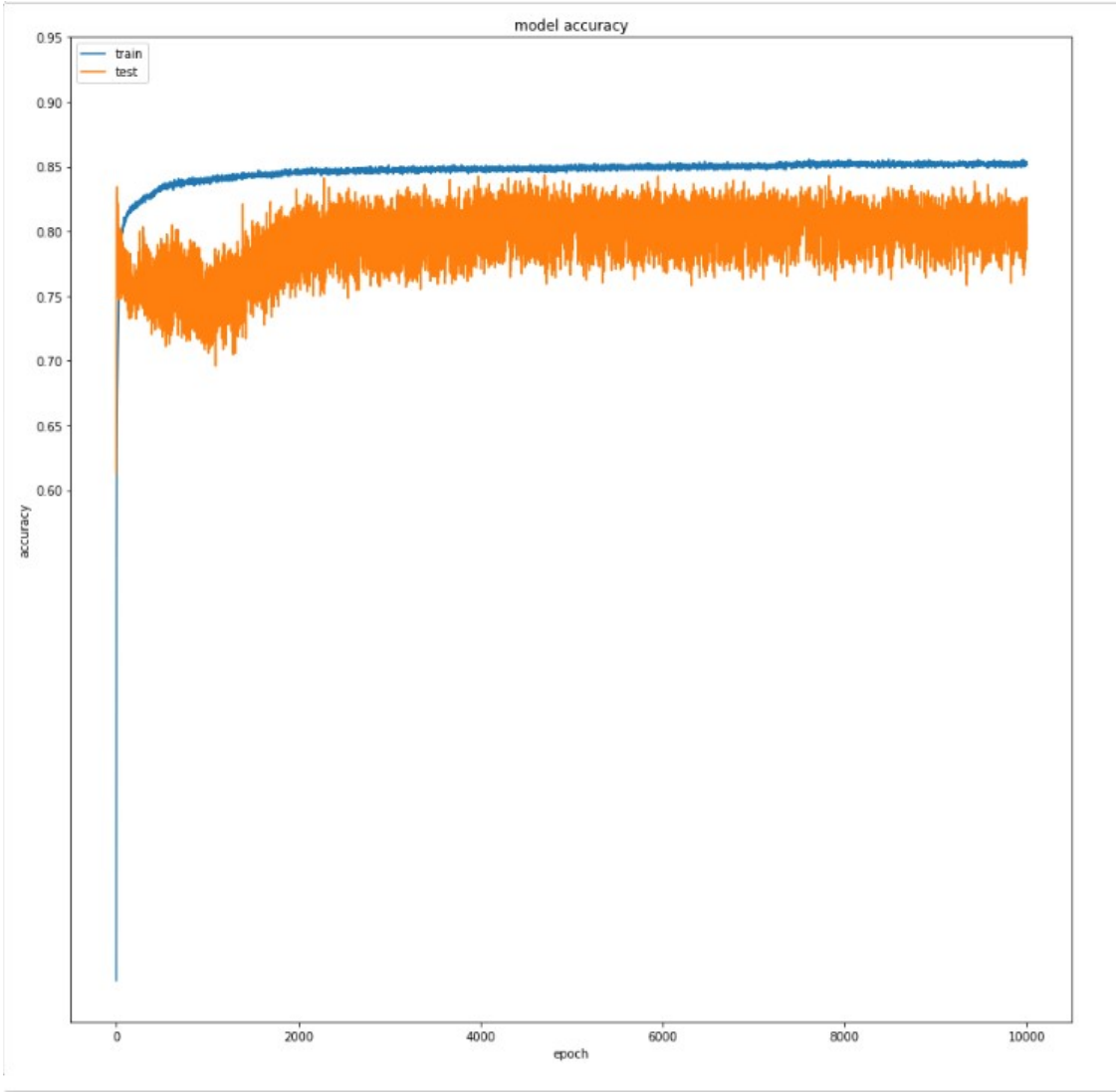


Fig 6.10 Plateaued Accuracy Curve

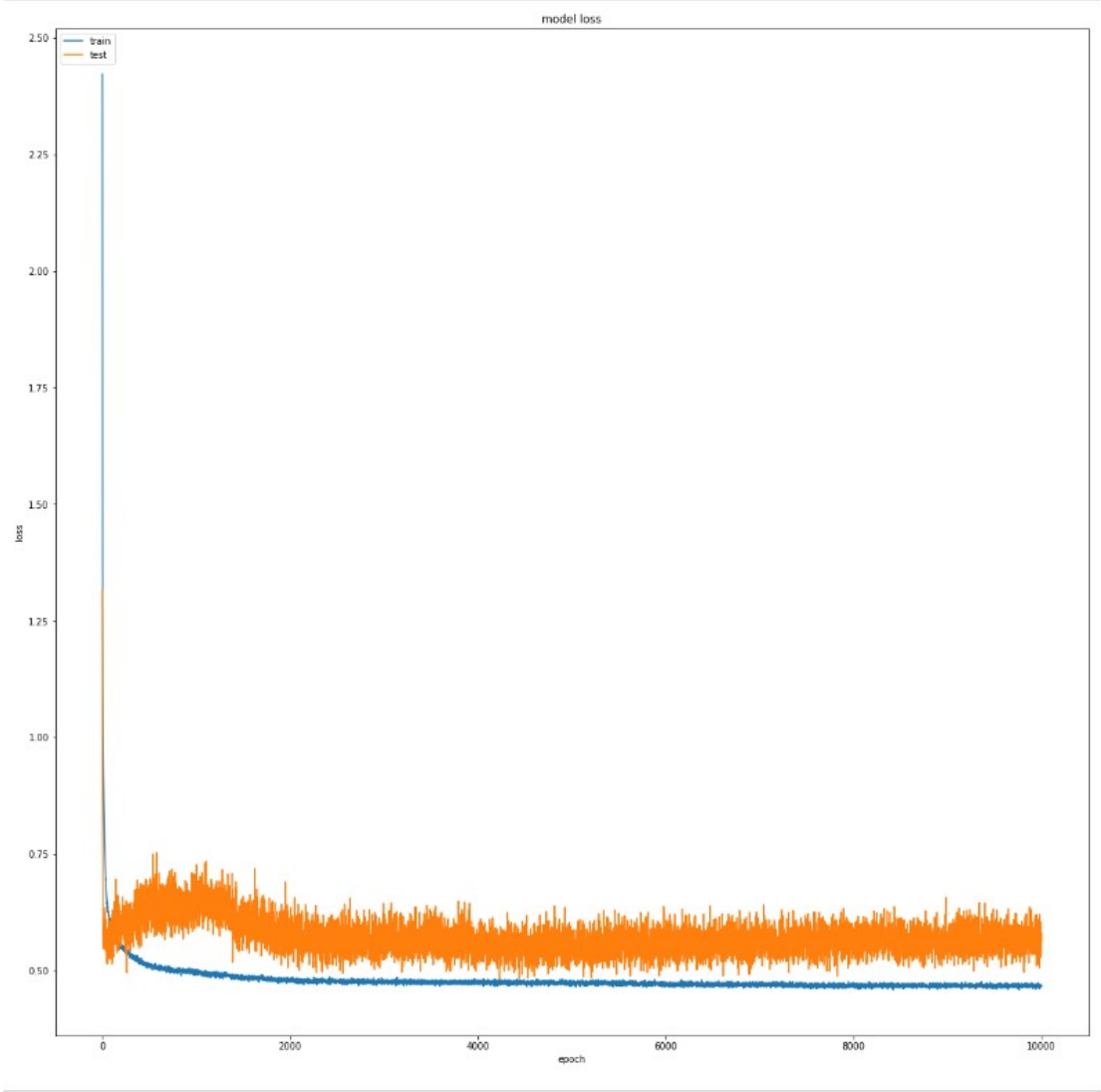


Fig 6.11 Plateaued Loss Curve

When it comes to dealing with categorization issues, the confusion matrix is a useful tool. It may be used to handle problems involving multiclass classification as well as binary classification. A binary classification confusion matrix is shown in Table 6.1.

	Predicted_Positive	**Predicted_Negative**
Actual_Positive	True Positive (TP)	False Positive (FP)
Actual_Negative	False Negative (FN)	True Negative (NP)

Table 6.1 Confusion Matrix for Binary_Classifiction

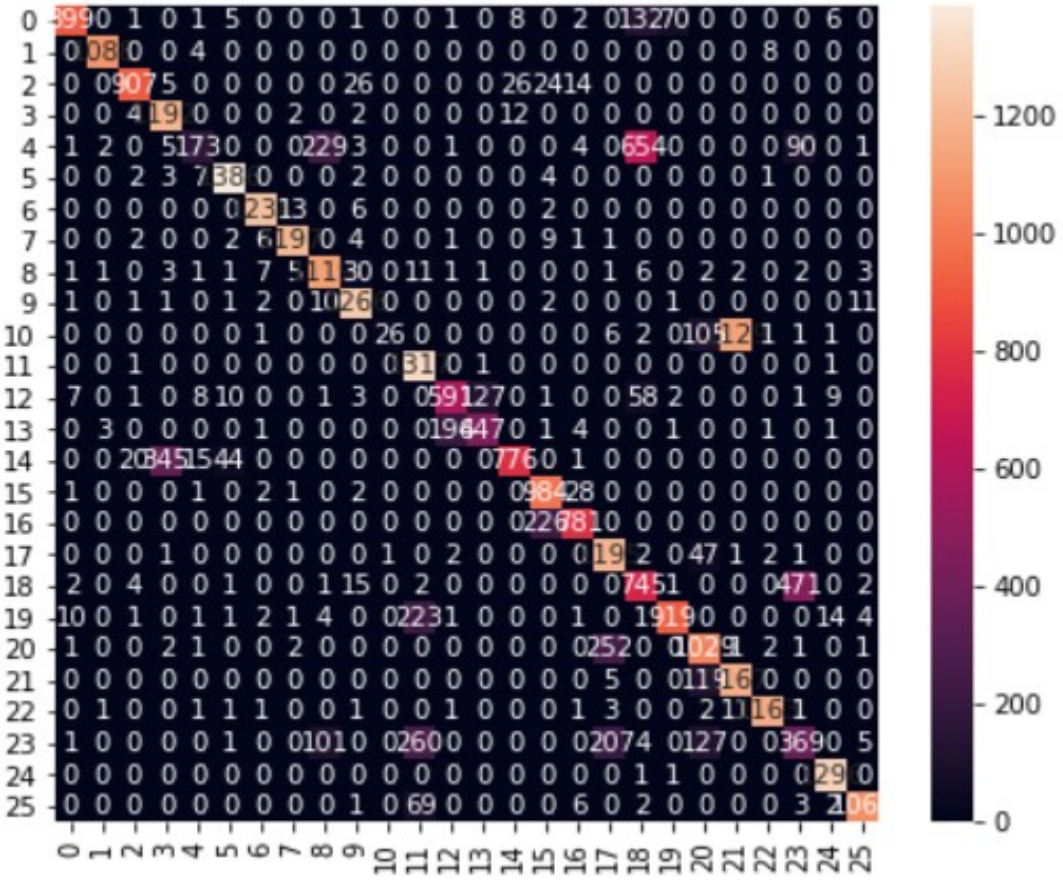


Fig 6.12 Confusion Matrix ASL Alphabet Classification

The brighter diagonal represents the accuracy of the model. In this case, it represents the True Positives that the model occurred to classify. We can see the model suffers a bit in distinguishing between “M” and “N”. This happened due to complexity in between the alphabet gesture and also for the fact that they are quite to similar upon 2-Dimensional perception. Futher, this is can resolved by improving the quantity and quality of data availble for them.

6.2.4 Classification Report

A classification report is one of the performance evaluation indicators for a classification-based machine learning model. It displays your model's accuracy, recall, F1 score, and support. It enables us to have a better understanding of the overall performance of our trained model. To understand the categorization report of a machine learning model, you must be familiar with all of the metrics displayed. I've discussed all of the indicators below for a clear comprehension of the categorization report of your machine learning model:

	Precision	Recall	F1-score	Support
0	0.97	0.80	0.88	1126
1	0.99	0.99	0.99	1100
2	0.96	0.91	0.93	1002
3	0.77	0.98	0.86	1212
4	0.81	0.15	0.25	1163
5	0.95	0.99	0.97	1405
6	0.98	0.98	0.98	1255
7	0.98	0.98	0.98	1223
8	0.76	0.93	0.84	1197
9	0.93	0.98	0.95	1296
10	0.96	0.02	0.04	1272
11	0.70	1.00	0.82	1320
12	0.74	0.72	0.73	819
13	0.78	0.68	0.73	655
14	0.94	0.65	0.77	1201
15	0.79	0.97	0.87	1019
16	0.93	0.78	0.84	1007
17	0.72	0.95	0.82	1252
18	0.46	0.60	0.52	1244
19	0.92	0.78	0.84	1183
20	0.72	0.80	0.76	1292
21	0.50	0.90	0.65	1291
22	0.99	0.98	0.98	1187
23	0.39	0.34	0.37	1075
24	0.97	1.00	0.99	1298
25	0.98	0.93	0.95	1146
Accuracy			0.80	30240
Macro avg	0.83	0.80	0.78	30240
Weighted avg	0.83	0.80	0.78	30240

Table 6.2 Classification Report

6.2.5 Real-Time Recognition Application

With the trained model saved as tensor lite object, it is ready to be implemented to recognize and classify the ASL alphabet in real-time. The model is capable of recognizing both hands and sign made by both hands. But for usability and interactivity, the recognition capability to assigned to only one hand. In this case, since majority people are right hand biased, the model is assigned to recognize gestures of ASL only by the right hand.

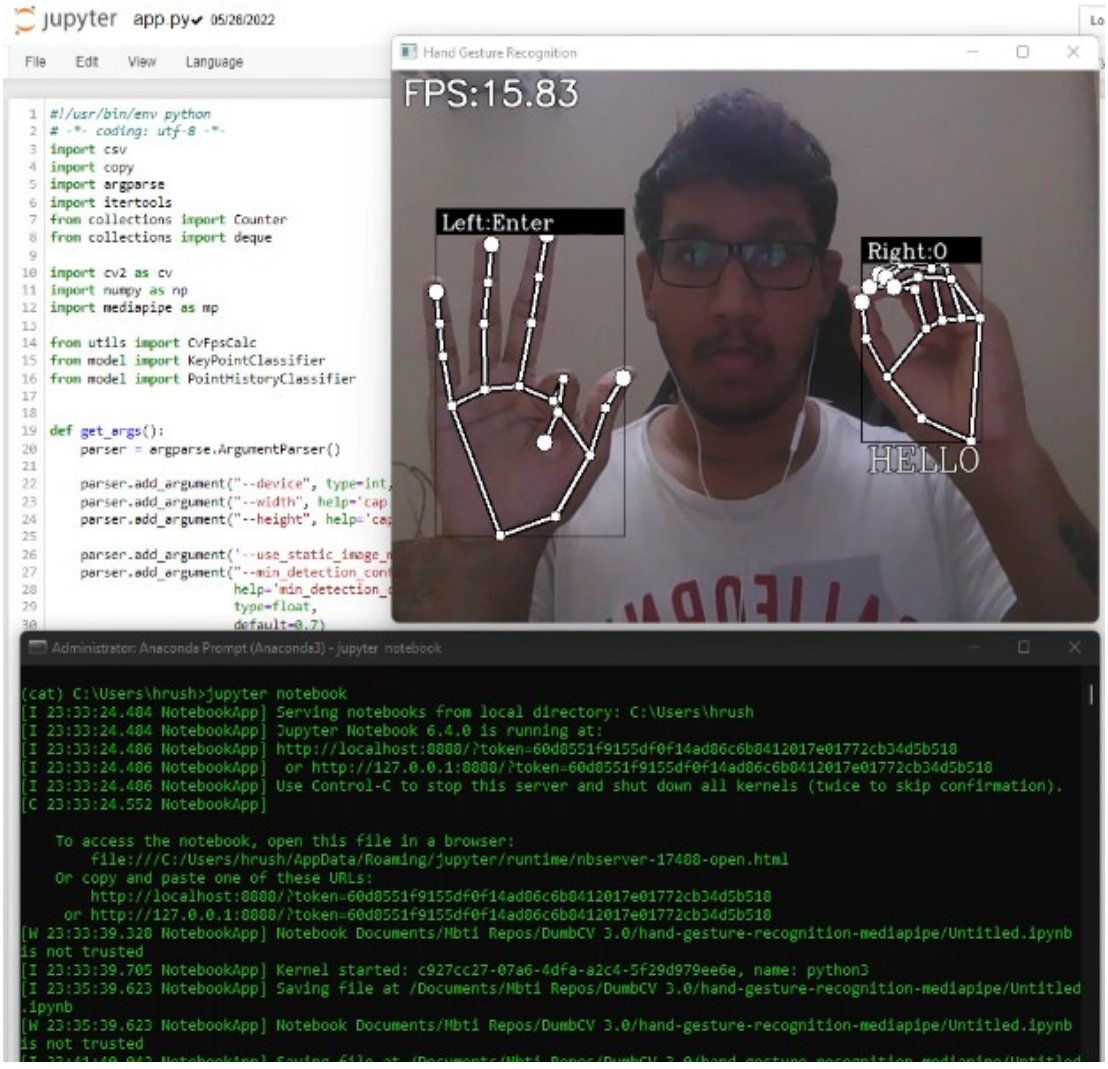


Fig 6.13 Real-Time Detection of ASL

To take advantage of the other hand, it hard-coded to recognize certain gestures. The gestures help the user to spell the words without using any keyboard interface. This enables the user to effective fingerspell using ASL even though they're slow at it. The functions assigned to left hand are as follows:

- Enter
- Space
- Erase
- Delete

Enter: The current alphabet recognized is stored and written on canvas. The gesture is replicated of that of "typing". From initial raised finger i.e., open palm, the index finger is dropped or closed to activate the Enter function.

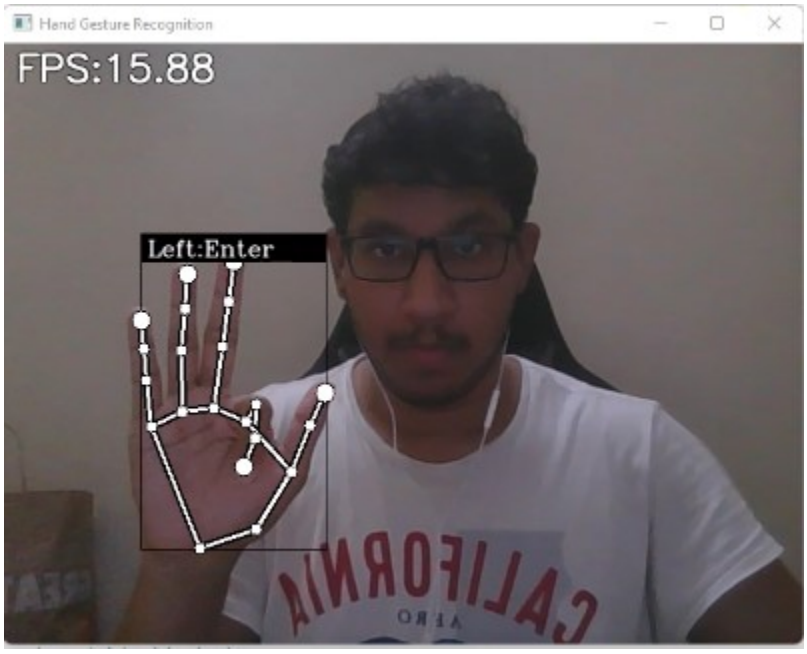


Fig 6.14 Enter Functionality

Space: A space is given and registered on canvas. The gesture associated with this is the dropping of the whole open palm to a perpendicular to the screen. This activates the Space function.

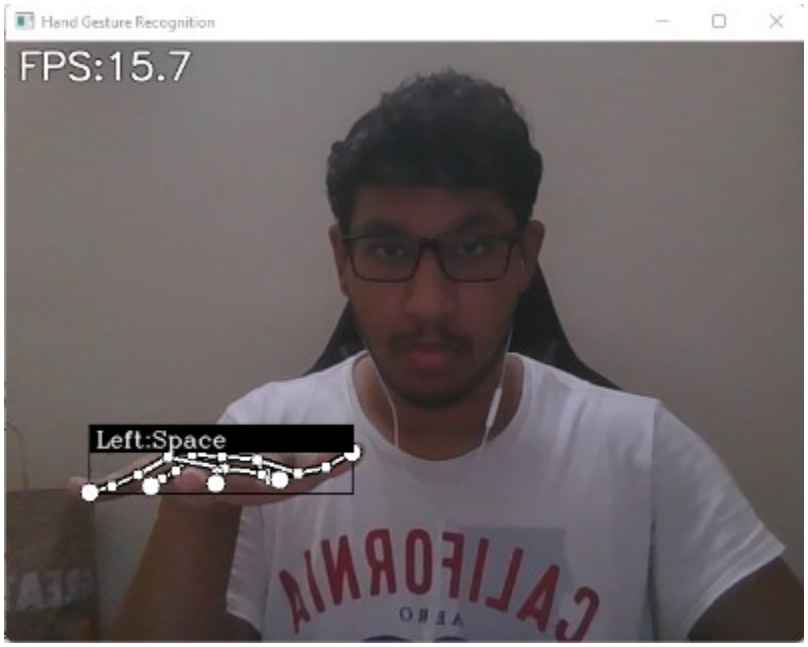


Fig 6.15 Space Functionality

Erase: The most recently written alphabet is erased from the sentence, leaving the rest of the sentence or word as it is. The gesture associated with this is the folding of thumb finger into the palm. This activates the Erase function.

Fig 6.16 Erase Functionality

Delete: The whole word or sentence written is cleared completely, leaving the canvas empty. The gesture associated with this is flipping of the palm, showing the back of the hand to the screen. This activates the Delete function.

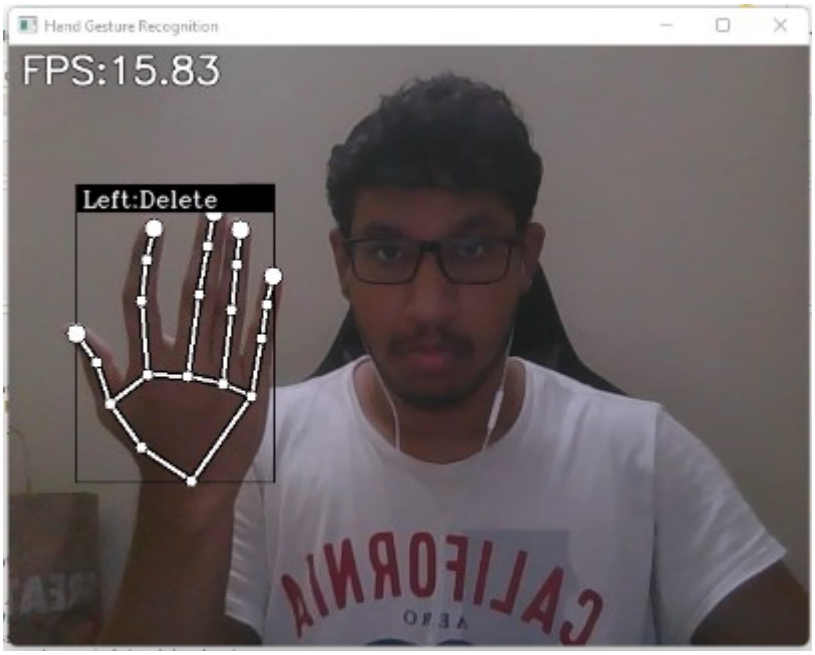


Fig 6.17 Delete Functionality

7. TESTING

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page

The primary goal of testing is to find flaws. Testing is the practice of attempting to find all possible flaws or weaknesses in a work product. It is the process of testing software to ensure that it satisfies its requirements and meets user expectations, and that it does not fail in an undesirable way.

7.1 Types of Testing

There are many types of testing methods are available in that mainly used testing methods are as follows.

7.1.1 Unit Testing

Unit testing entails creating test cases to ensure that the program's underlying logic is working properly and that programme inputs result in valid outputs. Validation should be performed on all decision branches and internal code flow. It is the testing of the application's individual software parts. It is done after an individual unit has been completed and before it is integrated. This is an intrusive structural test that depends on prior knowledge of the structure. Unit tests are used to verify a single business process, application, or system configuration at the component level. Unit tests guarantee that each individual route of a business process follows the published specifications and has clearly defined inputs and outputs.

7.1.2 Integration Testing

Integration tests are used to see if two or more software components can work together as a single application. Testing is event-driven, with a focus on the fundamental consequence of screens or fields. Integration tests indicate that, while the components were individually satisfying, the combination of components is proper and consistent, as demonstrated by successful unit testing. Integration testing is a type of testing that focuses on uncovering issues that occur from the integration of components.

7.1.3 Functional Testing

Functional tests demonstrate that the functions being tested are available in accordance with the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems / Procedures: interfacing systems or procedures must be invoked.

Functional tests are organized and prepared around requirements, important functions, or unique test cases. Furthermore, comprehensive coverage of Business process flows, data fields, established procedures, and subsequent processes must be considered for testing. Additional tests are found before functional testing is completed, and the effective value of present tests is assessed.

7.1.4 System Testing

System testing guarantees that the complete integrated software system complies with the specifications. It checks a setup to verify that the results are known and predictable. The configuration-oriented system integration test is a form of system testing. Process descriptions and flows are used in system testing, with an emphasis on pre-driven process connections and integration points.

7.1.5 White Box Testing

White Box Testing is a type of software testing in which the software tester is familiar with the software's inner workings, structure, and language, or at the very least its purpose. It serves a function. It's used to evaluate regions that aren't accessible with a black box level.

7.1.6 Black Box Testing

Testing software without knowing the inner workings, structure, or language of the module being tested is known as black box testing. Black box tests, like most other types of tests, require a definite source document, such as a specification or requirements document. It's a type of testing in which the programme being tested is treated as if it were a black box. It is impossible to "look" into it. The test accepts inputs and responds to outputs without taking into account how the software functions.

7.2 Test cases:

Tested	Test name	Inputs	Expected output	Actual Output	status
1	giving test image or video	image or video	input taken	successfully taken	success

Tested	Test name	Inputs	Expected output	Actual Output	status
2	loading model	invoking ANN model	model loaded successfully	Model Loaded	success

Tested	Test name	Inputs	Expected output	Actual Output	status
3	object detection	frames divided by model	palm detected	successfully detected	success

Tested	Test name	Inputs	Expected output	Actual Output	status
4	displaying output	detected ASL alphabet in image	image with ASL alphabet in box	successfully displayed	success

Fig 7.1 Test Cases

8. CONCLUSION

From classifying primarily static signs and alphabets, the Sign Language Recognition System has progressed to a system that can detect dynamic gestures in continuous sequences of pictures. Researchers are currently focusing their efforts on developing a vast repertoire for sign language recognition systems. Many researchers are employing a tiny vocabulary and a self-made database to construct their Sign Language Recognition System. For some of the countries engaged in building Sign Language Recognition System, a large database built for the system is still unavailable. It has always been difficult to communicate between a deaf-mute and a regular person. Our project's purpose is to lower the barrier between them. We have contributed to the field of American Sign Language recognition as part of our work. We created an ANN-based human hand gesture recognition system in this project. Our approach distinguishes itself by eliminating the requirement to create a model for each move based on hand traits such as fingers and curves. A functional real-time vision-based sign language recognition system for deaf and dumb persons has been created in this research. On our dataset, we attained a final accuracy of 83%. After incorporating two layers of algorithms, we were able to enhance our prediction. We also double-checked our results for similar-looking gestures that were more prone to misclassification. We can recognize practically all symbols this way if they are shown correctly, there is no background noise, and the illumination is acceptable.

9. FUTURE SCOPE

We hope to expand our datasets with other alphabets and refine the model so that it can recognize more alphabetical characteristics while maintaining high accuracy. By experimenting with various background removal methods, we hope to obtain improved accuracy even in the situation of complicated backgrounds. We're also considering upgrading the pre-processing to better detect gestures in low-light situations. We'd like to improve the system even further by include voice recognition so that blind individuals may benefit as well. Future scope includes but is not limited to:

- The expansion of our model to include additional sign languages, such as Indian Sign Language, since it presently only supports American Sign Language.
- Improving the model's ability to detect common words and phrases.
- Improve the neural network's ability to detect symbols using two hands.
- Using linear classifiers to improve the findings.
- Additional to the present static finger spelling, provide dynamic hand movements.
- Using convolutional neural networks to recognize motions taken by sensors like Kinect while also accounting for depth data.

10. BIBLIOGRAPHY

[1] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scienti_c Computing Conference (SciPy) (Jun 2010), oral Presentation

[2] Chai, X., Li, G., Lin, Y., Xu, Z., Tang, Y., Chen, X., Zhou, M.: Sign Language Recognition and Translation with Kinect (2013), http://vipl.ict.ac.cn/sites/default/_les/papers/_les/2013 FG xjchai Sign Language Recognition and Translation with Kinect.pdf

[3] Cire_san, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classi_cation. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 3642{3649. IEEE (2012)

[4] Cooper, H., Ong, E.J., Pugeault, N., Bowden, R.: Sign language recognition using sub-units. The Journal of Machine Learning Research 13(1), 2205{2231 (2012)

[5] Escalera, S., Bar, X., Gonzlez, J., Bautista, M.A., Madadi, M., Reyes, M., Ponce, V., Escalante, H.J., Shotton, J., Guyon, I.: Chalearn looking at people challenge 2014: Dataset and results. In: ECCV workshop (2014)

[6] Glorot, X., Bordes, A., Bengio, Y.: Deep Sparse Recti_er Networks. Proceedings of the 14th International Con- ference on Arti_cial Intelligence and Statistics 15, 315{323 (2011), http://eprints.pascal-network.org/archive/00008596/

[7] Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082 (2013)

[8] Chenyang Zhang et al, "Multi-modality American Sign Language Recognition", in Image Processing (ICIP), 2016 IEEE International Conf. ,2016. doi: 0.1109/ICIP.2016.7532886

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

APPENDIX A: IMAGE DATA AUGMENTATION

Image Data augmentation techniques are used to increase the training dataset artificially by modifying the versions of images in the collected data. We get better performance if our dataset is large enough. Instead of collecting manually to increase the dataset, image augmentation expands the dataset by different image augmentation techniques. We have different data augmentation techniques based on the color, position and have many predefined functions. Some of the image data augmentation techniques are:

Rotation of images:

A person can capture the photo from any angle so, image data augmentation gives all possible orientation of the images. Though there is change in orientation or the angle the information inside the image is the same.

Shifting of images:

This image augmentation technique shifts the images by changing the object's position in different ways. If the initial position of image is a,b then the new positions are A,B. This gives the model a variety of images and increases the performance of the model.

Flipping of images:

Flipping of images is similar to rotation of images. But here flipping can be done either in a horizontal or vertical manner.

Noising of Image:

In this technique noise is added to the image and we get more data to train the model. There are different types of noise as well. Generally we use filters to remove the noise whereas here we are adding the noise to make the model robust.

Blurring of image:

We might have images which are blurred. All images may not be of the same quality. Some might have very good quality; few might be moderate and remaining might be bad. In order to make the model robust we blur the original image. All the above types of image data augmentation are due to the position of the image. Similarly we have different types based on the color of the image.

Brightness of image:

This kind of augmentation changes the values of image pixels. By changing the image properties we get many images and the dataset also increases.

Contrast of image:

Contrast refers to the level of brightness exhibited in an image. This technique enhances the image features and makes use of the maximum of colors that are available in a device.

The above is some of the image data augmentation techniques. In conclusion, to have a model with high performance and accuracy within a small dataset. Image augmentation plays a crucial role and generates a wide range of data artificially.

APPENDIX-B: CONVOLUTIONAL NEURAL NETWORKS

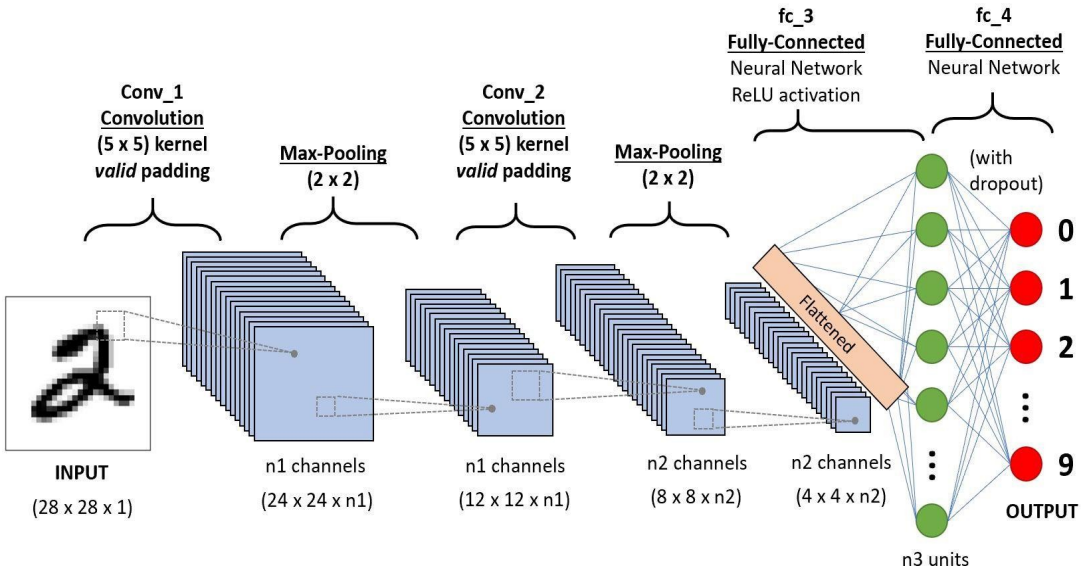


Fig B.1 Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Convolution Layer — The Kernel:

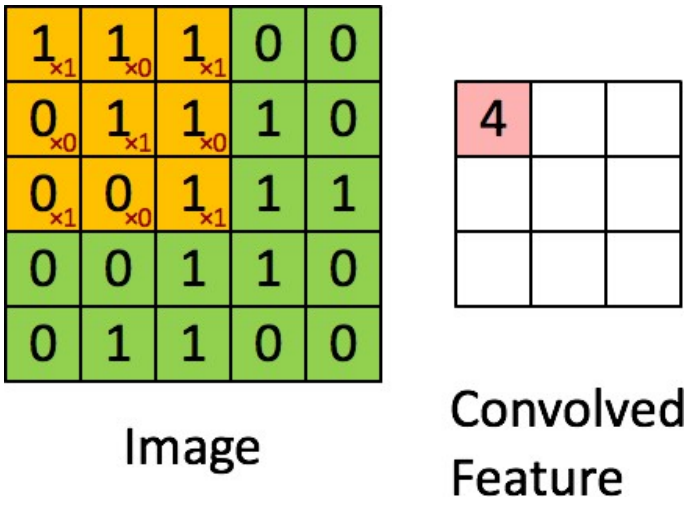


Fig B.2 Convolved Feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB).In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.

Pooling Layer:

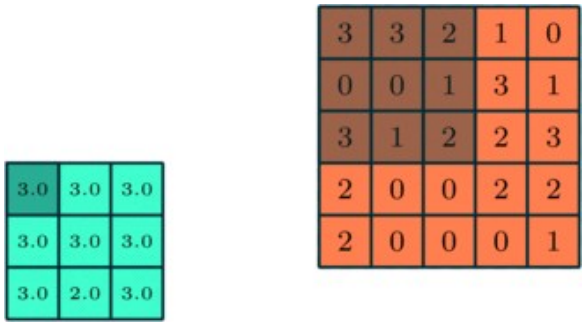
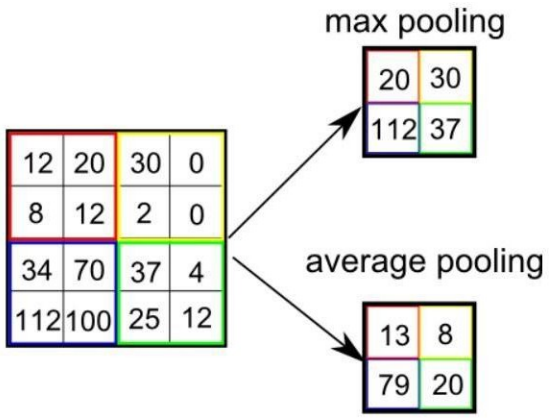


Fig B.3 Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant,



thus maintaining the process of effectively training of the model.

Fig B.4 Pooling Methods

Classification-Fully Connected layer:

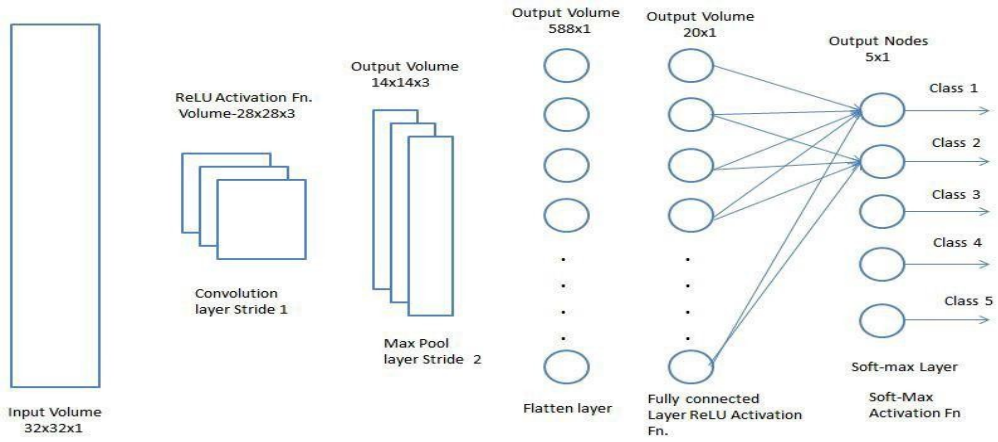


Fig B.5 Fully Connected Layer

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully- Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed- forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in
images and classify them using the SoftMax Classification technique.

Dense layer:

Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

Flatten Layer:

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

KERAS:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should

be possible via a clear path that builds upon what you've already learned.

- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Sign Language Recognition using Computer Vision and Neural Networks

ORIGINALITY REPORT

17%	11%	7%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Koushik Roy, Md. Akiful Hoque Akif. "Real Time Hand Gesture Based User Friendly Human Computer Interaction System", 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET), 2022 Publication	2%
2	google.github.io Internet Source	1%
3	www.nidcd.nih.gov Internet Source	1%
4	Submitted to Kingston University Student Paper	1%
5	Submitted to Nexford University Student Paper	1%
6	Submitted to Texas A&M University - Commerce Student Paper	1%

International Conference on Advances in Computer Engineering & Communication Technology

ICACET-2021

22nd & 23rd October 2021

ICACET-2021-AU-025

Organized by

Sponsored by

Associated with IOP Publishing

Certificate

Paper ID	*22*
Portrayal	*Presenter*
Title	*Prediction of Personality Traits based on Myers-Briggs Type Indicator - Catboost Model Approach*
Author	*Hrushyang A*
Institution	*Sreenidhi Institute of Science and Technology, Hyderabad, India*

Co-Coordinator
Dr. Ch V Raghavendran

Coordinator
Dr. T K Rama Krishna Rao

Department of IT, CSE & ECE
Aditya College of Engineering & Technology
(Permanently Affiliated to JNTUK, Kakinada, Approved by AICTE, New Delhi)
Recognized by UGC Under Sections 2(f) and 12(B) of UGC Act 1956
Surampalem, Andhra Pradesh, India.

International Conference on Advances in Computer Engineering & Communication Technology

ICACET-2021

22nd & 23rd October 2021

ICACET-2021-AU-027

Organized by

Sponsored by

Associated with IOP Publishing

Certificate

Paper ID	*22*
Portrayal	*Presenter*
Title	*Prediction of Personality Traits based on Myers-Briggs Type Indicator - Catboost Model Approach*
Author	*Abhiram Reddy G*
Institution	*Sreenidhi Institute of Science and Technology, Hyderabad, India*

Department of IT, CSE & ECE

Aditya College of Engineering & Technology

(Permanently Affiliated to JNTUK, Kakinada, Approved by AICTE, New Delhi)

Recognized by UGC Under Sections 2(f) and 12(B) of UGC Act 1956

Surampalem, Andhra Pradesh, India.

Co-Coordinator

Dr. Ch V Raghavendran

Coordinator

Dr. T K Rama Krishna Rao

RAJARAMBAPU INSTITUTE OF TECHNOLOGY

IEEE BOMBAY SECTION

2022 International Conference for Advancement in Technology (ICONAT)

21st – 22nd January 2022

Certificate

This is to certify that Dr./Prof./Mr./Ms. Rishith Muthyapu *has presented paper entitled* Machine Learning Models for Predicting and Clustering Customer Churn Based on Boosting Algorithms and Gaussian Mixture Model *in 2022 International Conference for Advancement in Technology (ICONAT) during 21st & 22nd January 2022.*

General Chair	Publication chair	Organizing Chair
Dr. Sandeep A. Thorat	Dr. Amol C. Adamuthe	Dr. Sachin K. Patil

Sreenidhi Institute of Science and Technology

Department of Computer Science and Engineering

PROJECT - II

Batch No: C-16		**Title Of The Project-II**
Roll No	**Name**	
18311A05C1	Hrushyang Adloori	Sign Language Prediction using Computer Vision and Neural Networks
18311A05D3	Abhiram Reddy Guduru	
18311A05F4	Rishith Muthyapu	

ABSTRACT

Speech impairment is a condition that limits an individual's ability to communicate verbally. Consequently, communication between a verbally handicapped person and a normal person has always been challenging. To solve this problem, sign language, which is one of the most ordered languages, has been adopted. This initiative aims to bridge the gap between differently-abled people, such as the deaf and dumb, and the general public. The use of hand landmarks in conjunction with deep learning aided in the development of a real-time prediction system. The aim is to fabricate alphabets in sign language from real-time hand gestures using Computer Vision. Primarily, the dataset of ASL alphabet is used which is available on Kaggle. The dataset is undergone a meticulous pre-processing for further usage. Media pipe framework developed by google is utilized for pre-processing. The framework extracts hand landmarks from the pictures in the dataset. These coordinate data is further processed into more efficient information through specific techniques. The final processed dataset is used to training a neural net model. This model is trained to classify the 26 alphabets of the English literature which are represented by Sign. This proposed system provides better flexibility and makes the model more trainable for specific instances. Hence is implemented to deploy a real-time ASL alphabet classifier that recognizes the sign alphabets using the constant visual input provided to it.

	Internal guide	**Project Coordinator**	**Head of the Department**
Hrushyang Adloori Abhiram Reddy Guduru Rishith Muthyapu	Mr. K. Damodhar Rao Assistant Professor Department of CSE	Mrs. M. Yellamma Assistant Professor Department of CSE	Dr. Aruna Varanasi Professor & HOD Department of CSE

Batch No: C-16		Title Of The Project-II
Roll No	**Name**	Sign Language Prediction using Computer Vision and Neural Networks
18311A05C1	Hrushyang Adloori	
18311A05D3	Abhiram Reddy Guduru	
18311A05F4	Rishith Muthyapu	

Table 1: Project/Internship correlation with appropriate POs/PSOs (Please specify level of Correlation, H/M/L against POs/PSOs)

H	**High**	**M**	**Moderate**	**L**	**Low**

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Project Correlation with POs/PSOs

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
M	L	H	H	L	M	M	H	M	L	M	M	H	M	M

	Internal guide	Project Coordinator	Head of the Department
Hrushyang Adloori	Mr. K. Damodhar Rao	Mrs. M. Yellamma	Dr. Aruna Varanasi
Abhiram Reddy Guduru	Assistant Professor	Assistant Professor	Professor & HOD
Rishith Muthyapu	Department of CSE	Department of CSE	Department of CSE

Roll No	Name	Title Of The Project-II
18311A05C1	Hrushyang Adloori	Sign Language Prediction using Computer Vision and Neural Networks
18311A05D3	Abhiram Reddy Guduru	
18311A05F4	Rishith Muthyapu	

Table 2: Nature of the Project/Internship work (Please tick √ Appropriate for your project)

Batch No.	Title Of The Project-II	Nature of Project-II			
		Product	Application	Research	Others (please specify)
C-16	Sign Language Prediction using Computer Vision		✓	✓	

	Internal guide	Project Coordinator	Head of the Department
Hrushyang Adloori	Mr. K. Damodhar Rao	Mrs. M. Yellamma	Dr. Aruna Varanasi
Abhiram Reddy Guduru	Assistant Professor	Assistant Professor	Professor & HOD
Rishith Muthyapu	Department of CSE	Department of CSE	Department of CSE

Table 3: Domain of the Project/ Internship work (Please tick √ Appropriate for your project)

Batch No.	Title Of The Project-II	Domain of the Project-II				
		ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND DEEP LEARNING	COMPUTER NETWORKS, INFORMATION SECURITY, CYBER SECURITY	DATA WAREHOUSING, DATA MINING, BIG DATA ANALYTICS	CLOUD COMPUTING , INTERNET OF THINGS	SOFTWARE ENGINEERING, IMAGE PROCESSING
C-16	Sign Language Prediction using Computer Vision	✓				

	Internal guide	Project Coordinator	Head of the Department
Hrushyang Adloori	Mr. K. Damodhar Rao	Mrs. M. Yellamma	Dr. Aruna Varanasi
Abhiram Reddy Guduru	Assistant Professor	Assistant Professor	Professor & HOD
Rishith Muthyapu	Department of CSE	Department of CSE	Department of CSE