Sign Language Detection

A Major Project Report submitted
in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

1. KOVVURI SUSHMA (18B01A0574)
2. GADIDESI CHANDRIKA (18B01A0590)
3. PAPPULA KUSUMA LATHA (18B01A05A9)
4. SANKA SRI NAGA VARDHINI VYSHNAVI (18B01A05B7)
5. KASANI RUPA SRI (19B01A0508)

Under the esteemed guidance of
Dr. P. KIRAN SREE
Professor and Head of the Department

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202
2021 – 2022

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)
(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

*This is to certify that the Mini Project-2 entitled "**SIGN LANGUAGE DETECTION**", is being submitted by **K.SUSHMA, G.CHANDRIKA, P.KUSUMA LATHA, S.VYSHNAVI ,K.RUPA SRI** bearing the **Regd. No. 18B01A0574, 18B01A0590, 18B01A05A9, 18B01A05B7, 19B01A0508** in partial fulfilment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by them under my guidance and supervision during the academic year **2021–2022** and it has been found worthy of acceptance according to the requirements of the university.*

Internal Guide **Head of the Department**

External Examiner

ACKNOWLEDGMENT

Behind every achievement there lies an unfathomable sea of graduates to those who activated it, without whom it would ever come into existence. To them we word of gratitude imprinted within us.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju, Chairman of SVES**, for his constant support of each and every progressive work of ours.

We wish to express our sincere thanks to **Dr. G. Srinivasa Rao, Principal of SVECW**, for providing us with all the facilities necessary to carry out this project successfully.

We wish to express our sincere thanks to **Dr. P. Srinivasa Raju, Vice-Principal of SVECW**, for being a source of inspirational constant encouragement.

On the successful completion of our project report, we convey our sincere thanks to **Dr. P. Kiran Sree, Head of the Department of Computer Science and Engineering**, for his valuable pieces of advice in completing this project successfully.

We are really thankful to our honourable guide **Dr. P. Kiran Sree, Head of the Department of Computer Science and Engineering**, who encouraged us a lot in the completion of the project work.

Project Associates

K. Sushma (18B01A0574)
G .Chandrika (18B01A0590)
P. Kusuma Latha(18B01A05A9)
S.S.N.V. Vyshnavi (18B01A05B7)
K. Rupa Sri (19B01A0508)

ABSTRACT

There have been several advancements in technology and a lot of research has been done to help the people who are deaf and dumb. The purpose of this project is to design a convenient system that is used to detect the visual-gestural language used by deaf and hard hearing people for communication purposes. One should learn sign language to interact with them. Most of the existing tools for sign language learning use external sensors which are costly. Because of this, the process of sign language learning is a very difficult task.

Our project aims at extending a step forward in this field by using Deep Learning and python. In this approach, a dataset is collected and the useful information extracted is input into supervised learning techniques. Computer recognition of sign language deals from sign gesture acquisition and continues till text generation followed by conversion of this text into speech. First the images are collected for learning using webcam and OpenCV. Then the images are labeled for sign language detection using labelImg. Next, we build a CNN model and then we train the model with the training dataset and finally the language is detected using OpenCV and webcam.

So ideally the result is a real-time object detection device that can detect different sign language poses based on American Sign Language System.

Keywords: deep learning, sign language recognition, hand gesture recognition, gesture analysis, CNN, OpenCV.

CONTENTS

LIST OF FIGURES

INTRODUCTION

1.INTRODUCTION

Sign Language is the means of communication among the deaf and mute community. Sign Language emerges and evolves naturally within the hearing-impaired community. Sign Language communication involves manual and non-manual signals where manual signs involve fingers, hands, arms and non-manual signs involve face, head, eyes and body. Sign Language is a well-structured language with a phonology, morphology, syntax and grammar. Sign language is a complete natural language that uses different ways of expression for communication in everyday life. Sign Language recognition systems transfer communication from human-human to human-computer interaction. The aim of the sign language recognition system is to present an efficient and accurate mechanism to transcribe text or speech, thus the "dialog communication" between the deaf and hearing person will be smooth.

American Sign Language (ASL) is a complete, natural language that has the same linguistic properties as spoken languages, with grammar that differs from English. ASL is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing, and is used by many hearing people as well.

There are two approaches for this problem:

Sensor Based Approach: This approach collects the data of gestures performed by using different sensors. The data is then analyzed and conclusions are drawn in accordance with the recognition model. In case of hand gesture recognition different types of sensors were used and placed on the hand, when the hand performs any gesture, the data is recorded and is then further analyzed. But this approach damages the natural motion of the hand because of use of external hardware. The major disadvantage is that complex gestures cannot be performed using this method.

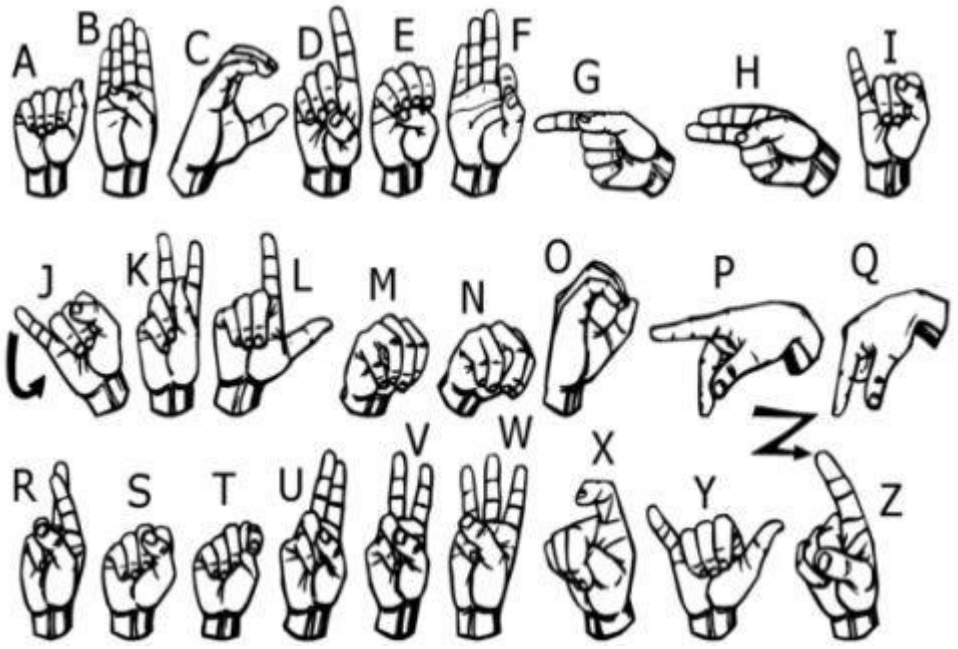
Vision Based Approach: This approach takes images from the camera as data of gestures. The vision-based method mainly concentrates on capturing images of gestures and extracting the main feature and recognizing it. This approach is more convenient to use as it does not need the user to wear any gadgets.

The goal of this project was to build a neural network able to classify which gesture of the Sign Language is being signed, given an image of a signing hand. This project is a first step towards building a possible sign language translator, which can take communications in sign language and translate them into written language. Such a translator would greatly lower the barrier for many deaf and mute individuals to be able to better communicate with others in day-to-day interactions.

Minimizing the verbal exchange gap among D&M and non-D&M people turns into a want to make certain effective conversation among all. Sign language translation is among one of the most growing lines of research and it enables the maximum natural manner of communication for those with hearing impairments. A hand gesture recognition system offers an opportunity for deaf people to talk with vocal humans without the need of an interpreter. The system is built for the automated conversion of ASL into textual content and speech.

This goal is further motivated by the isolation that is felt within the deaf community. Loneliness and depression exist at 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 in integrating a society.

In our project we primarily focus on producing a model which can recognize Fingerspelling based hand gestures in order to form a complete word by combining each gesture. The gestures we aim to train are as given in the image below. Therefore, to enable dynamic communication, we present a sign language recognition system that uses Convolutional Neural Networks (CNN) in real time to translate an image of a user's signs into text.



Objectives

- Collection of images for dataset
- Building a CNN model and training the model with a training dataset.
- Integrating the model with the GUI.
- Real Time translation of sign language to text in live.
- Finally, the text that is predicted will be converted into speech.

SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

2.1 Existing System

The increasing growth of machine learning, computer techniques divided into traditional methods and machine learning methods. This section describes the related works of conversion of Sign language to text and how machine learning methods are better than traditional methods.

The existing method in this project is the sensor-based method. the sensor-based approach collects the data of gestures performed by using different sensors. The data is then analyzed and conclusions are drawn in accordance with the recognition model. In case of hand gesture recognition different types of sensors were used and placed on the hand, when the hand performs any gesture, the data is recorded and is then further analyzed. But this approach damages the natural motion of the hand because of use of external hardware. The major disadvantage is that complex gestures cannot be performed using this method.

Disadvantages:

- Need to carry sensors every time with hand
- High cost
- Low Accuracy
- Complex gestures cannot be recognized
- No speech conversion is present.

2.2 Proposed System

In our project we primarily focus on producing a CNN model which can recognize Sign Language gesture and can convert into text. The predicted text is again converted to speech so that the visually challenging people can also able to understand the gesture. Both the text and audio will be shown in our Application. Here using convolution neural networks which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus our system produces the text and then converted into voice using google APIs.

Advantages :

- Accuracy is good
- Low complexity
- Highly efficient
- Complex gestures can also be detected
- Speech conversion is present

2.3 Feasibility Study

A feasibility study should be conducted after the project has been pitched but before any work has actually started. The study is part of the project planning process. In fact, it's often done in conjunction with a SWOT Analysis or project risk assessment, depending on the specific project.

Generally, the feasibility study is used for determining the resource cost, benefits and whether the proposed system is feasible with respect to the organization. The proposed system feasibility could be as follows:

- Technical Feasibility
- Economic Feasibility
- Behavioural Feasibility
- Operational Feasibility
- Social Feasibility

Technical Feasibility:

Technical feasibility is the formal process of assessing whether it is technically possible to manufacture a product or service. Before launching a new offering or taking up a client project, it is essential to plan and prepare for every step of the operation. Technical feasibility helps determine the efficacy of the proposed plan by analysing the process, including tools, technology, material, labour and logistics.

Technical feasibility deals with the existing technology, software and hardware requirements for the proposed system. The proposed system "Sign Language Detection" using Deep Learning. Thus, the project is considered technically feasible for the development. The work for the project can be done with current equipment, existing software technology and available personnel. Hence the proposed system is technically feasible.

- To understand about the development of the product with the available technology.
- To have an idea that it is done with the trending technologies and whether it fulfilled all the features of the project.

Economic Feasibility:

An evaluation of economic feasibility must include reliable estimates of the economic benefits and costs of the project. If the benefits generated by our project exceed project costs, then the project is considered to be economically feasible.

This method is most frequently used for evaluating the effectiveness of a Project. It is also called as benefit analysis. The project is developed on existing software technology. Since the required hardware and software for developing the system is already available in the organization, it doesn't cost much developing the proposed system.

Behavioural Feasibility

It evaluates and estimates the user attitude or behaviour towards the development of new system. It helps in determining if the system requires special effort to educate, retrain, transfer, and changes in employee's job status on new ways of conducting business. This project has much behavioural feasibility because users are provided with a better facility.

Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

Social Feasibility

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

SYSTEM REQUIREMENTS SPECIFICATION

3. SYSTEM REQUIREMENTS SPECIFICATION (SRS)

A SRS is a document which sets out what the client expects and what is expected of the software system which is being developed. It is a mutual agreement and insurance policy between client and developer and is a vital part of the Software Development Lifecycle.

It is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interaction that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do well as what it is not expected to do.

Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

Requirements analysis can be a long and tiring process during which many delicate psychological skills are involved. Large systems may confront analysts with hundreds or thousands of system requirements. New systems change the environment and relationships between people, so it is important to identify all the stakeholders, take into account all their needs to ensure they understand the implications of the new system. Analysis can employ several techniques to elicit the requirements from the customer.

3.1 Software Requirements

The software requirements are descriptions of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from the client's point of view.

Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.Python can be treated in a procedural way, an object-oriented way or a functional way.

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc

Keras:

Keras runs on top of open-source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA. Keras applications module is used to provide pre-trained model for deep neural networks

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

3.2 Hardware requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

1. Processor: Intel core i5
2. RAM: 8GB
3. Hard disk: 516GB

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Introduction

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

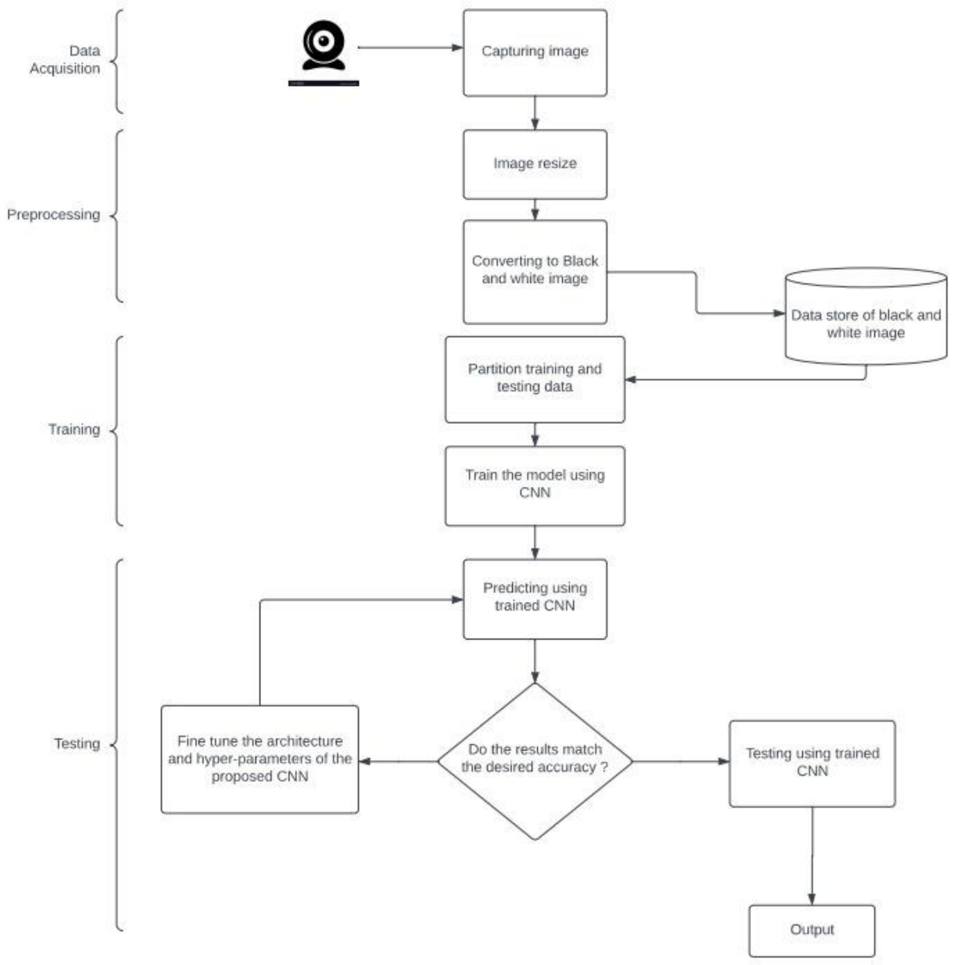
System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is similar to a blueprint or plan for the solution, and is used later during implementation, testing and maintenance.

The design activity is often divided into two separate phase-system designs and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided.

A design methodology is a systematic approach to creating a design by application of a set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it’s partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for property designing the part.

System Architecture


Data Acquisition
Preprocessing
Training
Testing
Capturing image
Image resize
Converting to Black and white image
Data store of black and white image
Partition training and testing data
Train the model using CNN
Predicting using trained CNN
Do the results match the desired accuracy ?
Fine tune the architecture and hyper-parameters of the proposed CNN
Testing using trained CNN
Output

4.2 UML Diagrams

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

The goal is for UML to become a common language for creating models of object oriented computer software.

In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Importance of UML Diagram

UML diagrams can be used as a way to visualize a project before it takes place or as documentation for a project afterward. But the overall goal of UML diagrams is to allow teams to visualize how a project is or will be working, and they can be used in any files, not just software engineering. The diagrams will allow teams to visualize together how a system or process will work or did work. It can provide new ideas for how the team needs to collaborate to achieve the goal of the workflow process.

UML can be used to develop diagrams and provide users(programmers) with ready-to-use, expressive modelling examples. Some UML tools generate program language code from UML.

UML can be used for modelling a system independent of a platform language. UML helps to organize, plan and visualize a program. It is used in a variety of purposes and its readability and re-usability make it an ideal choice for programmer.

USE CASE DIAGRAM:

A Use case diagram is a graphical depiction of a user’s possible interactions with a system. A Use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

1. Functionalities to be represented as use case.
2. Actors.
3. Relationships among the use cases and actors.

Use case diagram symbols and notation:

- **Use cases:** A use case describes a function that a system performs to achieve the user’s goal. Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** An actor represents a role of a user that interacts with the system that you are modelling. Stick figures that represent the people actually employing the use cases.
- **Associations:** A Line between actors and use cases.

In our example the preliminaries are,

ACTORS: Admin , User , Server
USE CASES: Preparing the subset , Build and train the model , starts the application , opens the webcam , capture the gesture , Display the gesture in text , text to speech , Hears the speech

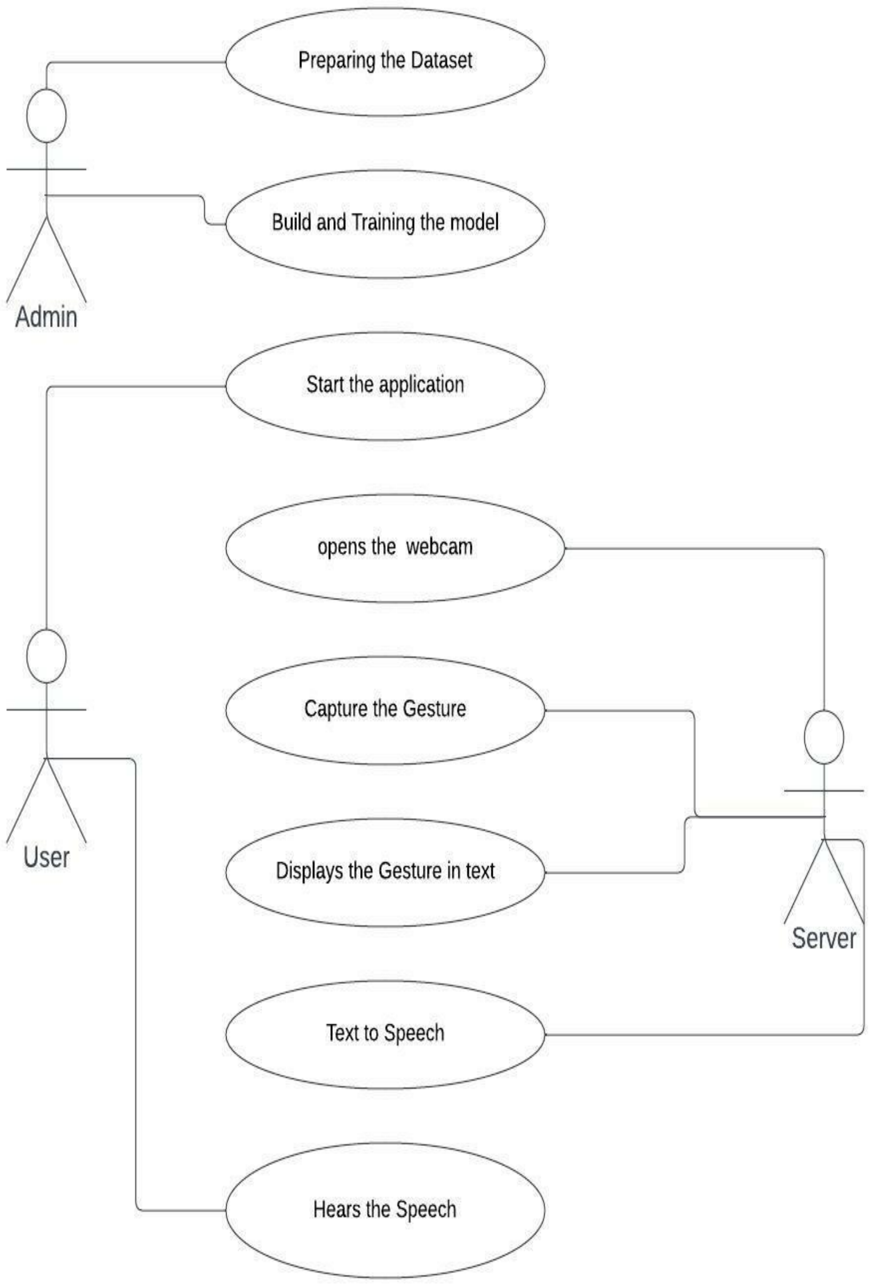


Fig: 4.2.1 – Use Case Diagram

CLASS DIAGRAM:

A class diagram in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modelling, the classes of the conceptual design are often split into subclasses.

A class diagram resembles a flowchart in which classes are portrayed as boxes, each boxes that contain three compartments.

1. The top compartment contains the name of the class; It is printed in bold and centred, and the first letter is capitalized.
2. The middle compartment contains the attributes of the class; They are left-aligned and the first letter is lowercase.
3. The bottom compartment contains the operations the class can execute; They are also left-aligned and the first letter is lowercase.

There are three types of modifiers that are used to decide the visibility of attributes and operations.

1. **'+'** is used for public visibility
2. **'#'** is used for protected visibility
3. **'-'** is used for private visibility

CLASSES: User, System.

The methods in System are takeDataset() , opencam() , captureGesture() , generateText(), generateSpeech().

The methods in User are uploadDataset() , buildModel() , trainModel() , viewResults()

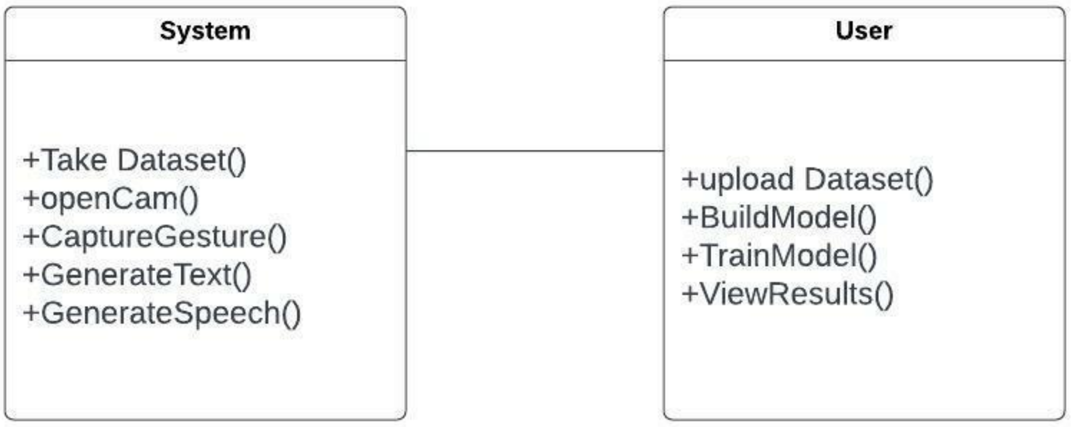


Fig: 4.2.2 – Class Diagram

SEQUENCE DIAGRAM:

A Sequence diagram is an interaction diagram that emphasizes the time ordering of messages. It consists of a set of objects and their Relationships, including the messages that may be dispatched among them. Sequence diagrams address the dynamic view of the system. Sequence diagrams are two dimensional in nature.

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the runtime. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Notations of the sequence diagram

- **Lifeline:** An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram.
- **Actor:** A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. An actor may or may not represent a physical entity, but it purely depicts the role of an entity.
- **Activation:** It is represented bya thin rectangle on the lifeline.It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.
- **Messages:** The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

Following are types of messages enlisted below:

- **Create message:** It describes a communication ,particularly between the lifelines of an interaction describing that target has been instantiated.
- **Call message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.
- **Return message:** It defines a particular communication between the lifelines of interaction that represents the flow of information from the receiver of the corresponding caller message.

- **Self message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.

OBJECTS: Admin , Server , User

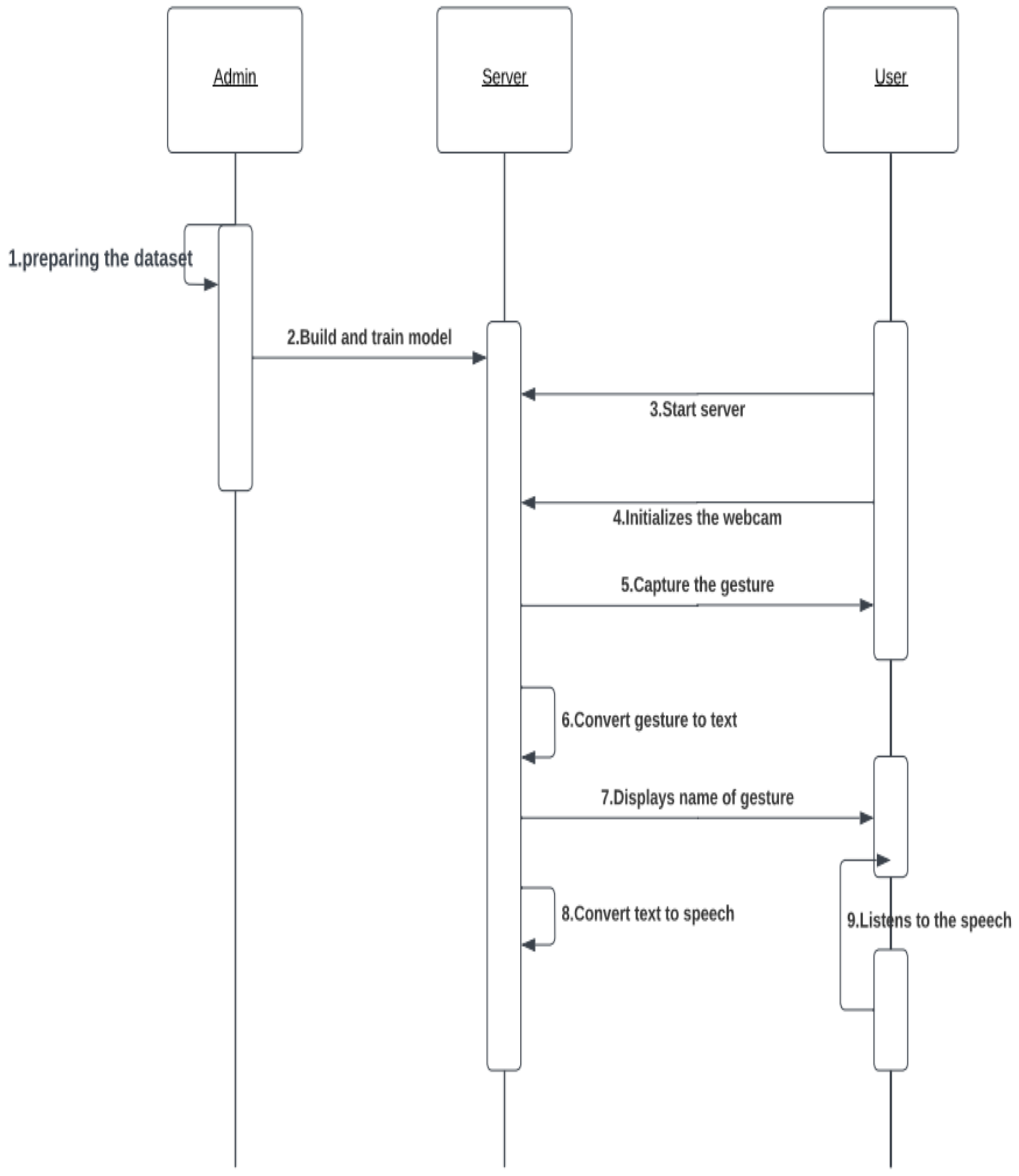


Fig: 4.2.3 – Sequence Diagram

ACTIVITY DIAGRAM:

An activity diagram is a special kind of state chart diagram that shows the flow from activity to activity within a system. Activity Diagrams address the dynamic view of a system. They are especially important in modelling the function of a System and emphasize the flow of control among objects

• It describes the flow of activity in the system.

• Before drawing an activity diagram, you should list activity of user and connection between both activities.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The basic purpose of the activity diagram is capture the dynamic behaviour of the system. It is also called object-oriented flowchart.

This UML diagram focuses on the execution and flow of the behaviour of a system instead of implementation. Activity diagram consists of activities that are made up of actions that apply to behavioural modelling technology.

Activity diagram is used to model business processes and workflows. These diagrams are used in software modelling as well as business modelling. most commonly activity diagram is used to model the workflow in a graphical way, which is easily understandable. Capture the dynamic behaviour of a system. Generate high-level flowcharts to represent the workflow of any application. It uses the action nodes, control nodes and object nodes.

Activity diagram notations

Activity diagrams symbols can be generated by using the following notations

- **Initial states:** The starting stage before an activity takes place is depicted as the initial state.
- **Final states:** The state which the system reaches when a specific process ends is known as final state.
- **Decision Box:** It is a diamond shape box which represents a decision with alternate paths. It represents the flow of control.

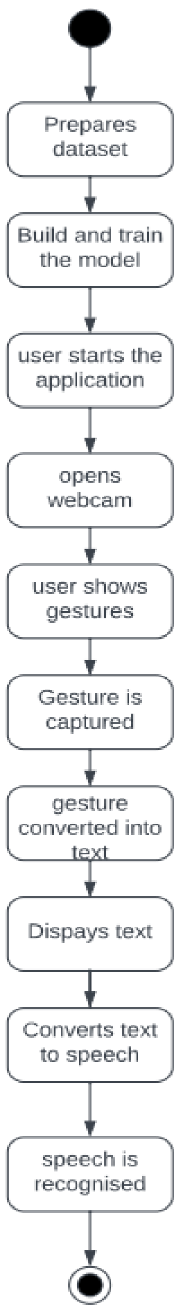


Fig: 4.2.4 – Activity Diagram

COLLABORATION DIAGRAM:

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the UML. These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams.

It is the organization of the objects that sends messages. A collaboration diagram is very similar to a sequence diagram. Collaboration diagram shows the objects and their association with other objects. Unlike a sequence diagram, a collaboration diagram shoes the relationship among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways.

Notations of a collaboration diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time.

The four major components of a collaboration diagram are:

1. **Objects:** An object is represented by an object symbol showing the name of the object and its class underlined, separated by a column.
2. **Object name:** class name
3. **Actors:** Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction. If you have several actor instances in the same diagram, try keeping them in the periphery of the diagram.
4. **Links:** Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram. A link is a relationship among objects across which messages can be sent. In collaboration diagram, a link is shown as a solid line between two objects.
5. **Messages:** A message is a communication between objects that conveys information with the expectation that activity will ensue. In collaboration diagram, a message is shown as a labelled arrow placed near a link.
   - The message is directed from sender to receiver.
   - The receiver must understand the message.

- The association must be navigable in that direction.

OBJECTS: User, System.

When user selects category and chooses any one of the options then the system asks to allow the camera. Then a Yoga posture along with its name is displayed to help the user perform. A countdown timer starts when the user starts performing correct posture.

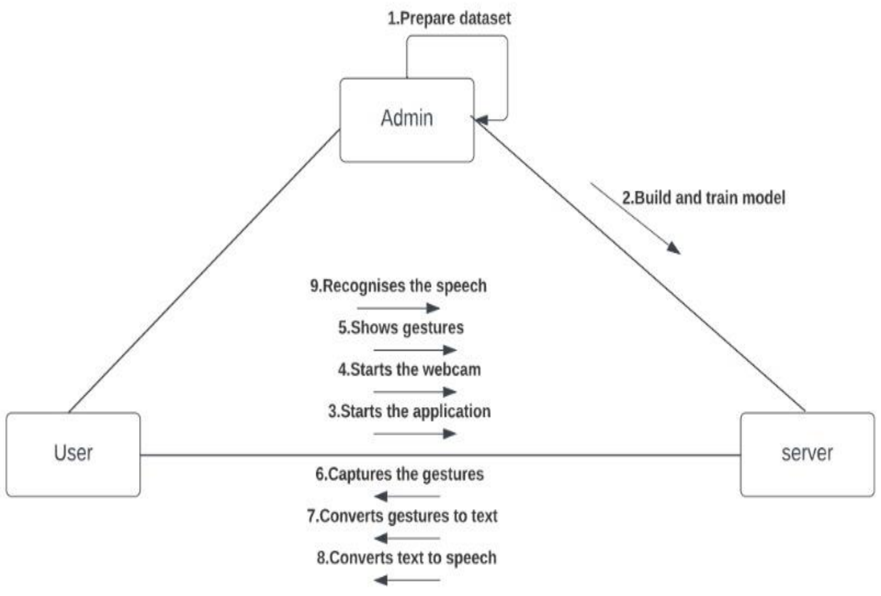


Fig: 4.2.5 – Collaboration Diagram

SYSTEM IMPLEMENTATION

5. SYSTEM IMPLEMENTATION

5.1 Introduction

American sign language is a predominant sign language Since the only disability Deaf and Dumb (hereby referred to as D&M) people have is communication related and since they cannot use spoken languages, 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, behaviour and visuals. D&M people make use of their hands to express different gestures to express their ideas with other people. Gestures are the non-verbally exchanged messages and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language. A sign language is a language which uses gestures instead of sound to convey meaning combining hand-shapes, orientation and movement of the hands, arms or body, facial expressions and lip-patterns. Contrary to popular belief, sign language is not international. These vary from region to region.

5.2 Project Modules

- **Building and training the model**

  This model consists of preparing the dataset and pre-processing of dataset. Then the dataset is divided into training dataset and testing dataset, And then building a CNN model. After building the model the model is trained with the training dataset.

- **Live Detection module**

  In this module, the user can directly turn on their camera and start showing the gestures, these gestures are then given to the model and the model translates the gesture into text. The user can see the translated sign to text in the screen.

- **Text to Speech conversion module:**

  The user can also able to hear the translated text into speech. Both the text and audio will be shown to the user. i.e., which ever is present in the screen the user will be able to hear it.

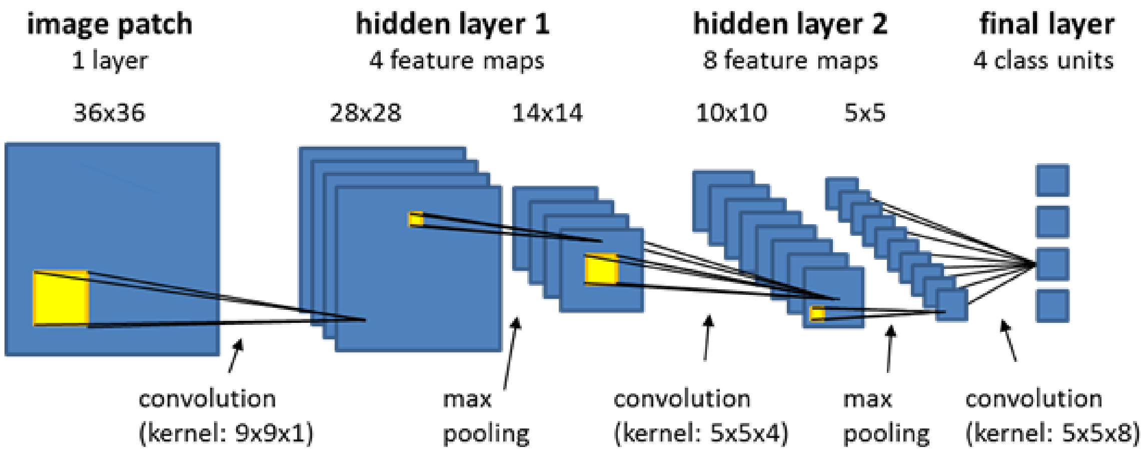
5.3 Algorithm

Convolutional Neural Networks :

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 overlaps to cover the entire visual area.

Convolutional Layer :

In convolution layer we take a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process we will create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour.



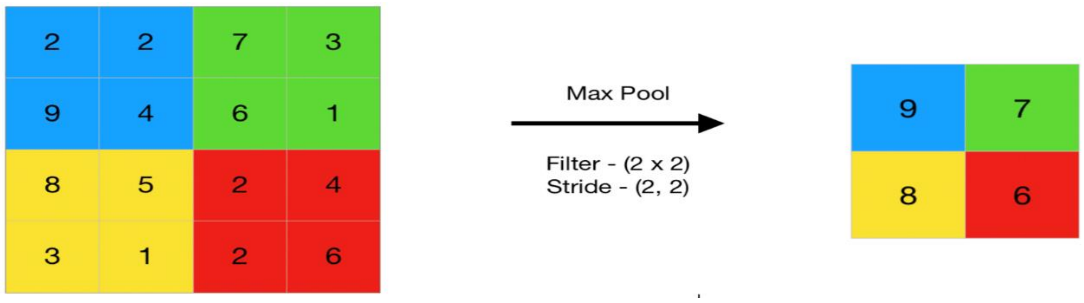
Relu Layer:

Rectified linear unit is used to scale the parameters to non negativevalues.We get pixel values as negative values too . Inthis layer we make them as 0’s. The purpose of applying the rectifier function is to increase the non-linearity in our images. The reason we want to do that is that images are naturally non-linear. The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation. What the rectifier function does to an image like this is remove all the black elements from it, keeping only those carrying a positive value (the grey and white colors).The essential difference between the non-rectified version of the image and the rectified one is the progression of colors. After we rectify the image, you will find the colors changing more abruptly. The gradual change is no longer there. That indicates that the linearity has been disposed of.

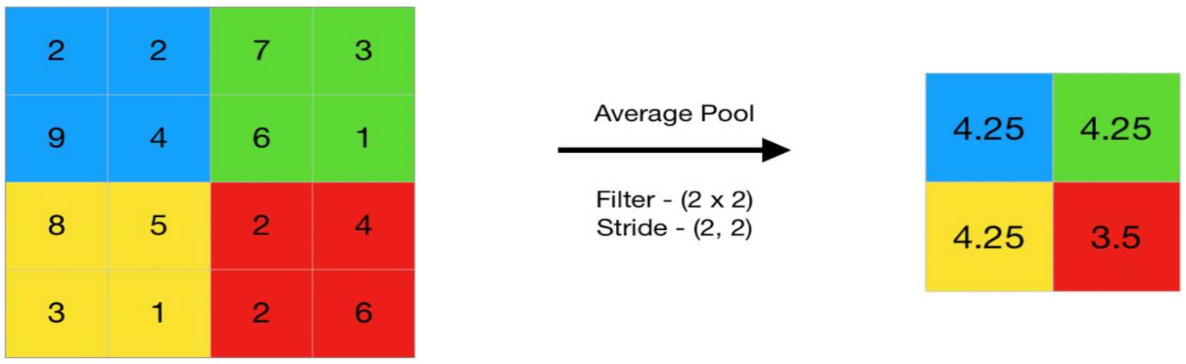
Pooling Layer:

We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of "overfitting" from coming up. There are two types of pooling:

a. Max Pooling: In max pooling we take a window size [for example window of size 2*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.



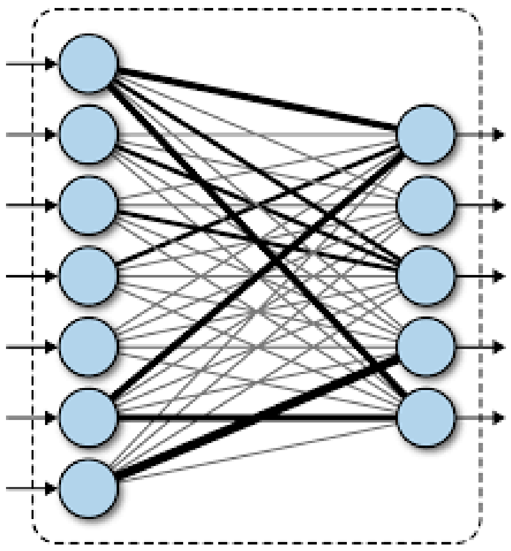
b. Average Pooling: In average pooling, we take advantage of of all Values in a window. Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.



Fully Connected Layer:

The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network. It has neurons linked to each other ,and activates if it identifies patterns and sends signals to output layer .the outputlayer gives output class based on weight values, For now, all you need to know is that the loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness. That requires certain things to be altered in our network. These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features and has to be reviewed multiple times for the sake of optimization.This full connection process practically works as follows:

- The neuron in the fully-connected layer detects a certain feature; say, a hand.
- It preserves its value
- It communicates this value to the classes trained images.



Final output layers

After getting values from fully connected layer, we will connect them to the final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

Code snippets of the implemented functionalities:

- The below function is used to display the predefined gestures at the right most window

```
def openimg():
        cv2.namedWindow("Image", cv2.WINDOW_NORMAL)
        image = cv2.imread('template.png')
        cv2.imshow("Image",image)
        cv2.setWindowProperty("Image",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FULLSC
REEN)
        cv2.resizeWindow("Image",298,430)
         cv2.moveWindow("Image", 1052,214)
```

- The predictor function is used to predict the corresponding gesture from the list of alphabets

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

    if result[0][0] == 1:
          return 'A'
    elif result[0][1] == 1:
          return 'B'
    elif result[0][2] == 1:
          return 'C'
    elif result[0][3] == 1:
          return 'D'
    elif result[0][4] == 1:
          return 'E'
    elif result[0][5] == 1:
          return 'F'
    elif result[0][6] == 1:
          return 'G'
    elif result[0][7] == 1:
            return 'H'
    elif result[0][8] == 1:
            return 'I'
    elif result[0][9] == 1:
            return 'J'
    elif result[0][10] == 1:
            return 'K'
    elif result[0][11] == 1:
            return 'L'
    elif result[0][12] == 1:
            return 'M'
    elif result[0][13] == 1:
            return 'N'
    elif result[0][14] == 1:
            return 'O'
    elif result[0][15] == 1:
            return 'P'
    elif result[0][16] == 1:
            return 'Q'
    elif result[0][17] == 1:
            return 'R'
    elif result[0][18] == 1:
            return 'S'
    elif result[0][19] == 1:
            return 'T'
    elif result[0][20] == 1:
            return 'U'
    elif result[0][21] == 1:
            return 'V'
    elif result[0][22] == 1:
            return 'W'
    elif result[0][23] == 1:
            return 'X'
    elif result[0][24] == 1:
            return 'Y'
    elif result[0][25] == 1:
            return 'Z'
```

- Whenever we run the application, it displays the gif of all gestures and repeats until the next action is performed

```
class Dashboard(QtWidgets.QMainWindow):
    def __init__(self):
        super(Dashboard, self).__init__()
        self.count = 0
        self.setWindowFlags(QtCore.Qt.WindowMinimizeButtonHint)
        cv2.destroyAllWindows()
        self.setWindowIcon(QtGui.QIcon('icons/windowLogo.png'))
        self.title = 'Sign language Recognition'
        uic.loadUi('UI_Files/dash.ui', self)
        self.setWindowTitle(self.title)
        self.timer = QTimer()
        if(self.scan_sinlge.clicked.connect(self.scanSingle)==True):
            self.timer.timeout.connect(self.scanSingle)
        self.scan_sinlge.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor)
)
        self.exit_button.clicked.connect(self.quitApplication)
        self._layout = self.layout()
        self.label_3 = QtWidgets.QLabel()
        movie = QtGui.QMovie("icons/dashAnimation.gif")
        self.label_3.setMovie(movie)
        self.label_3.setGeometry(0,160,780,441)
        movie.start()
        self._layout.addWidget(self.label_3)
        self.setObjectName('Message_Window')
```

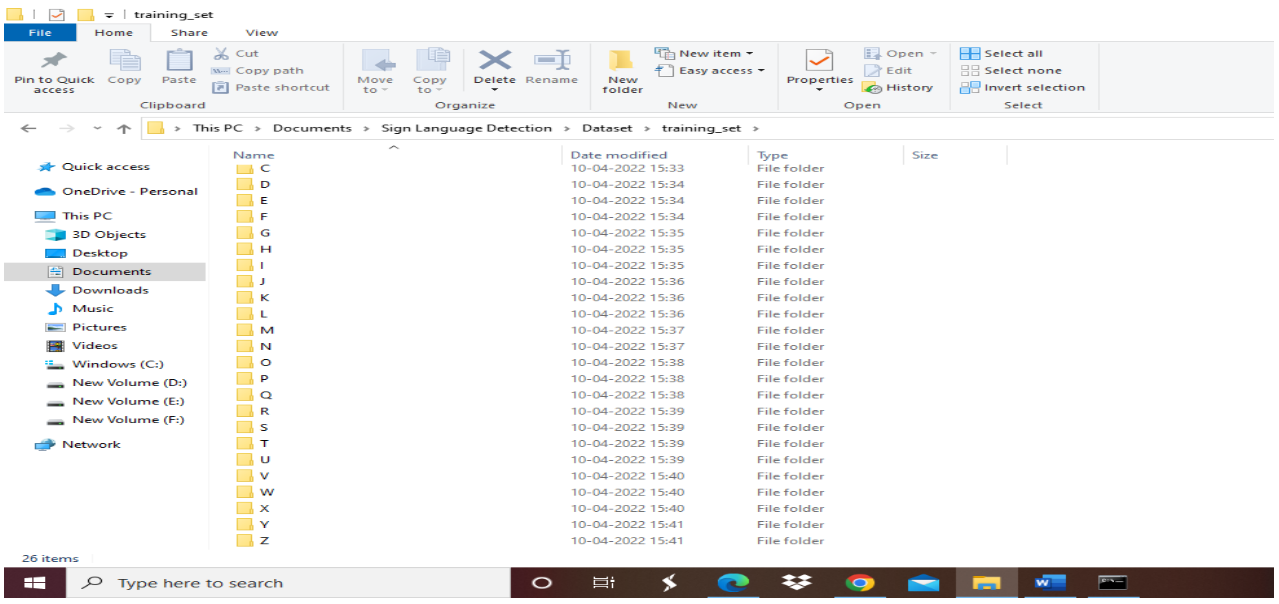
- This scansingle function is used for opening of webcam and capturing of particular gesture by adjusting the light and then passing it to the predictor function to display the particular text

```
def scanSingle(self):
        """Single gesture scanner """
        try:
            clearfunc(self.cam)
        except:
            pass
        uic.loadUi('UI_Files/scan_single.ui', self)
        self.setWindowTitle(self.title)

        if(self.scan_sinlge.clicked.connect(self.scanSingle)):
            controlTimer(self)
        self.pushButton_2.clicked.connect(lambda:clearfunc(self.cam))
        self.linkButton.clicked.connect(openimg)

        self.scan_sinlge.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor)
)

        try:
            self.exit_button.clicked.connect(lambda:clearfunc(self.cam))
        except:
            pass
        self.exit_button.clicked.connect(self.quitApplication)
        img_text = ''
        while True:
            ret, frame = self.cam.read()
            frame = cv2.flip(frame,1)
            try:
                frame=cv2.resize(frame,(321,270))
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                img1 = cv2.rectangle(frame, (150,50),(300,200), (0,255,0),
thickness=2, lineType=8, shift=0)
            except:
                keyboard.press_and_release('esc')

            height1, width1, channel1 = img1.shape
            step1 = channel1 * width1

            qImg1 = QImage(img1.data, width1, height1, step1,
QImage.Format_RGB888)

            try:
                self.label_3.setPixmap(QPixmap.fromImage(qImg1))
                slider1=self.trackbar.value()
            except:
                pass

            lower_blue = np.array([0, 0, 0])
            upper_blue = np.array([179, 255, slider1])

            imcrop = img1[52:198, 152:298]
            hsv = cv2.cvtColor(imcrop, cv2.COLOR_BGR2HSV)
            SLD = cv2.inRange(hsv, lower_blue, upper_blue)

            cv2.namedWindow("SLD", cv2.WINDOW_NORMAL )
            cv2.imshow("SLD", SLD)
            cv2.setWindowProperty("SLD",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FUL
LSCREEN)
            cv2.resizeWindow("SLD",118,108)
            cv2.moveWindow("SLD", 894,271)

            hwnd = winGuiAuto.findTopWindow("SLD")
            win32gui.SetWindowPos(hwnd, win32con.HWND_TOP,
0,0,0,0,win32con.SWP_NOMOVE | win32con.SWP_NOSIZE | win32con.SWP_NOACTIVATE)
            l = str(img_text)
            self.count += 1
            if(l != "None" and l != '' and l != None):
                print(str(img_text))
                tts(str(img_text),self.count)


            try:
                self.textBrowser.setText("\n\n\t"+str(img_text))
            except:
                pass

            img_name = "1.png"
            save_img = cv2.resize(SLD, (image_x, image_y))
            cv2.imwrite(img_name, save_img)
            img_text = predictor()

            if cv2.waitKey(1) == 27:
                break

        self.cam.release()
        cv2.destroyAllWindows()
```

- The tts function is used to takes the text displayed on screen and then convert into voice.

```
def tts(word,count):
    Message = word
    try:
        speech = gTTS(text = Message)
        strname = 'SLD' + str(count)+'.mp3'
        speech.save(strname)
        playsound(strname)
        os.remove(strname)
    except:
        print("error occured")
```

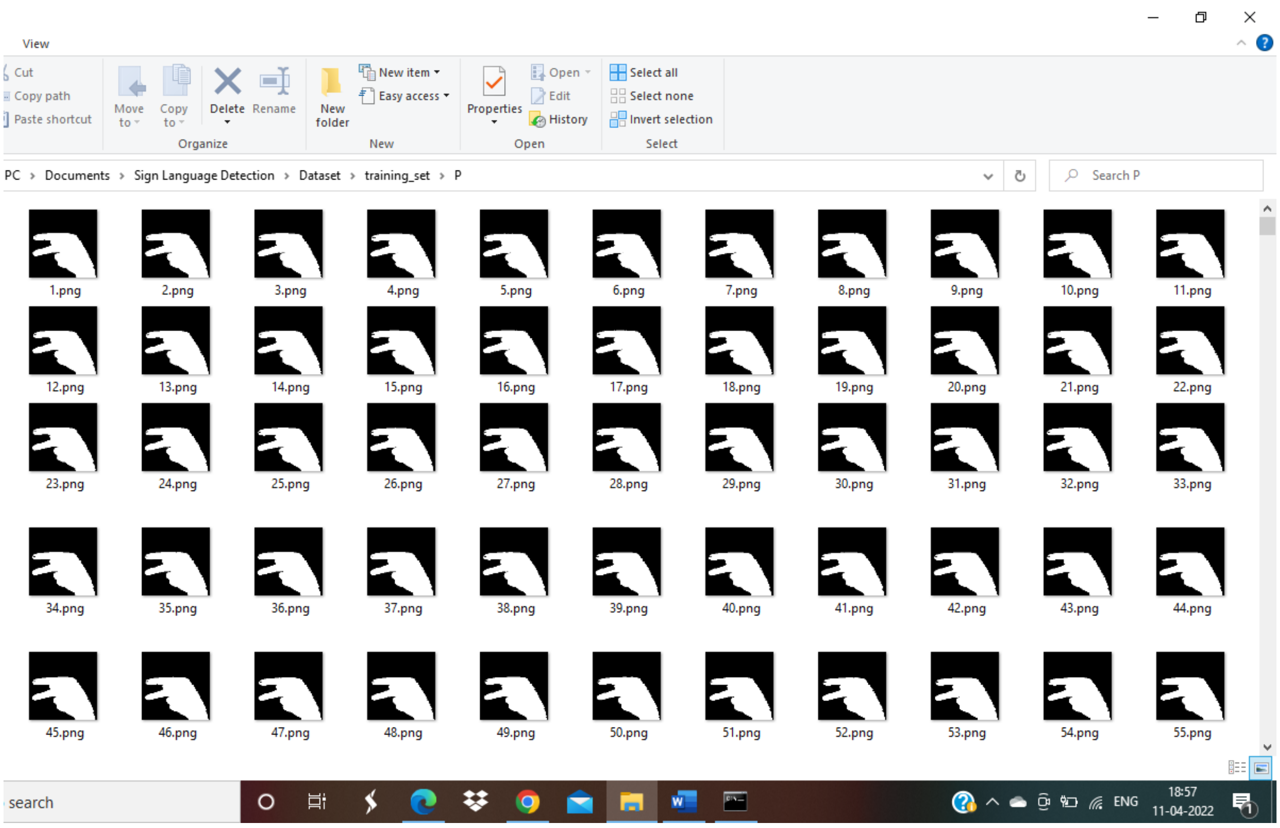
5.4 Screens



This PC › Documents › Sign Language Detection › Dataset › training_set ›

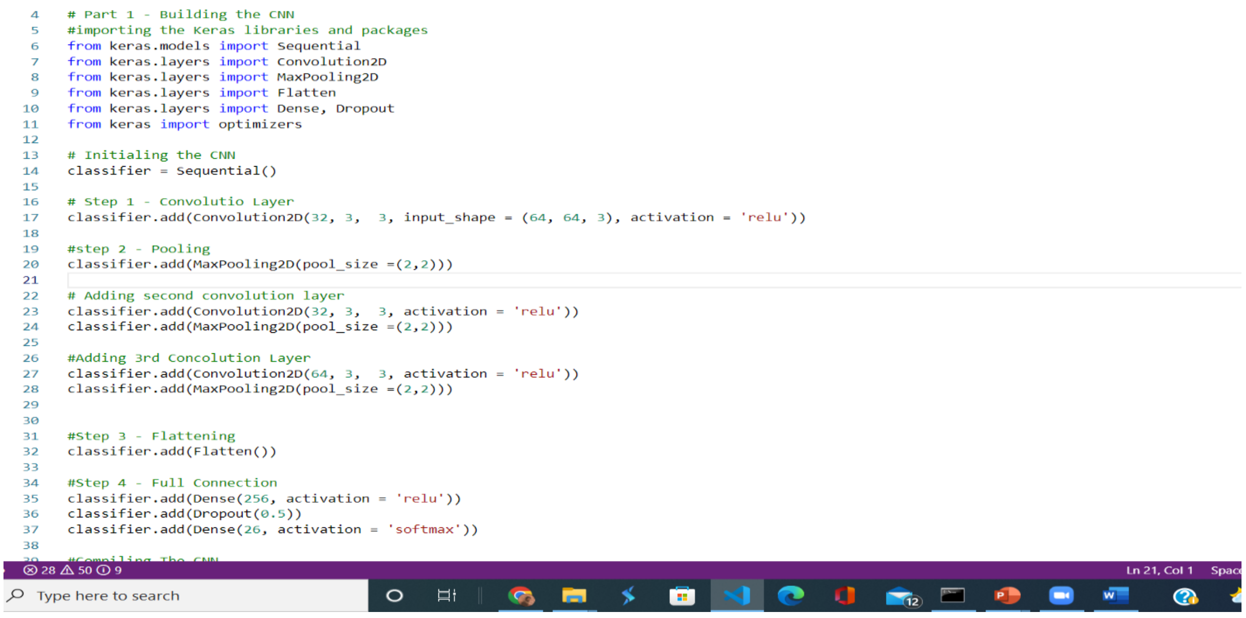
Name	Date modified	Type	Size
C	10-04-2022 15:33	File folder	
D	10-04-2022 15:34	File folder	
E	10-04-2022 15:34	File folder	
F	10-04-2022 15:34	File folder	
G	10-04-2022 15:35	File folder	
H	10-04-2022 15:35	File folder	
I	10-04-2022 15:35	File folder	
J	10-04-2022 15:36	File folder	
K	10-04-2022 15:36	File folder	
L	10-04-2022 15:36	File folder	
M	10-04-2022 15:37	File folder	
N	10-04-2022 15:37	File folder	
O	10-04-2022 15:38	File folder	
P	10-04-2022 15:38	File folder	
Q	10-04-2022 15:38	File folder	
R	10-04-2022 15:39	File folder	
S	10-04-2022 15:39	File folder	
T	10-04-2022 15:39	File folder	
U	10-04-2022 15:39	File folder	
V	10-04-2022 15:40	File folder	
W	10-04-2022 15:40	File folder	
X	10-04-2022 15:40	File folder	
Y	10-04-2022 15:41	File folder	
Z	10-04-2022 15:41	File folder	

26 items



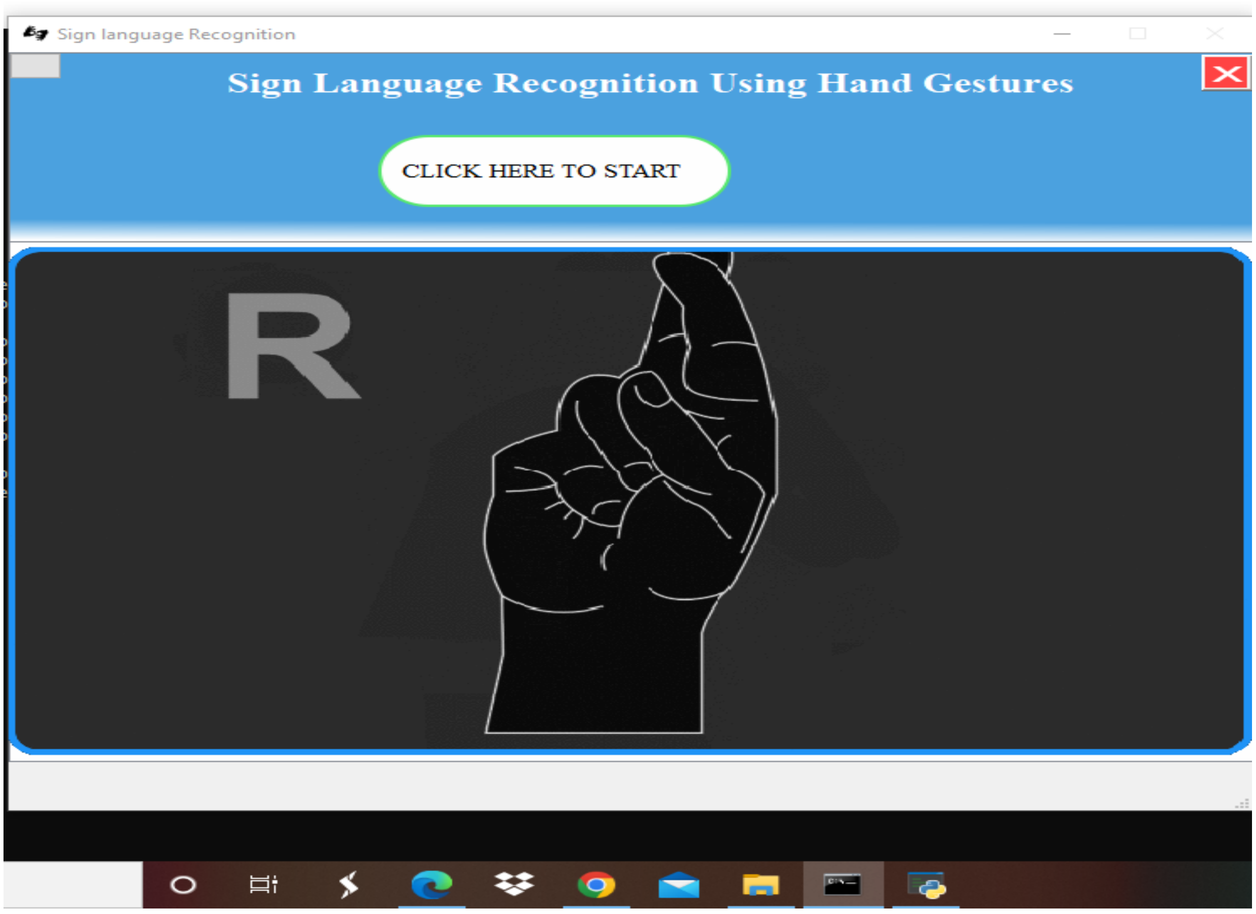
PC › Documents › Sign Language Detection › Dataset › training_set › P

Datasets



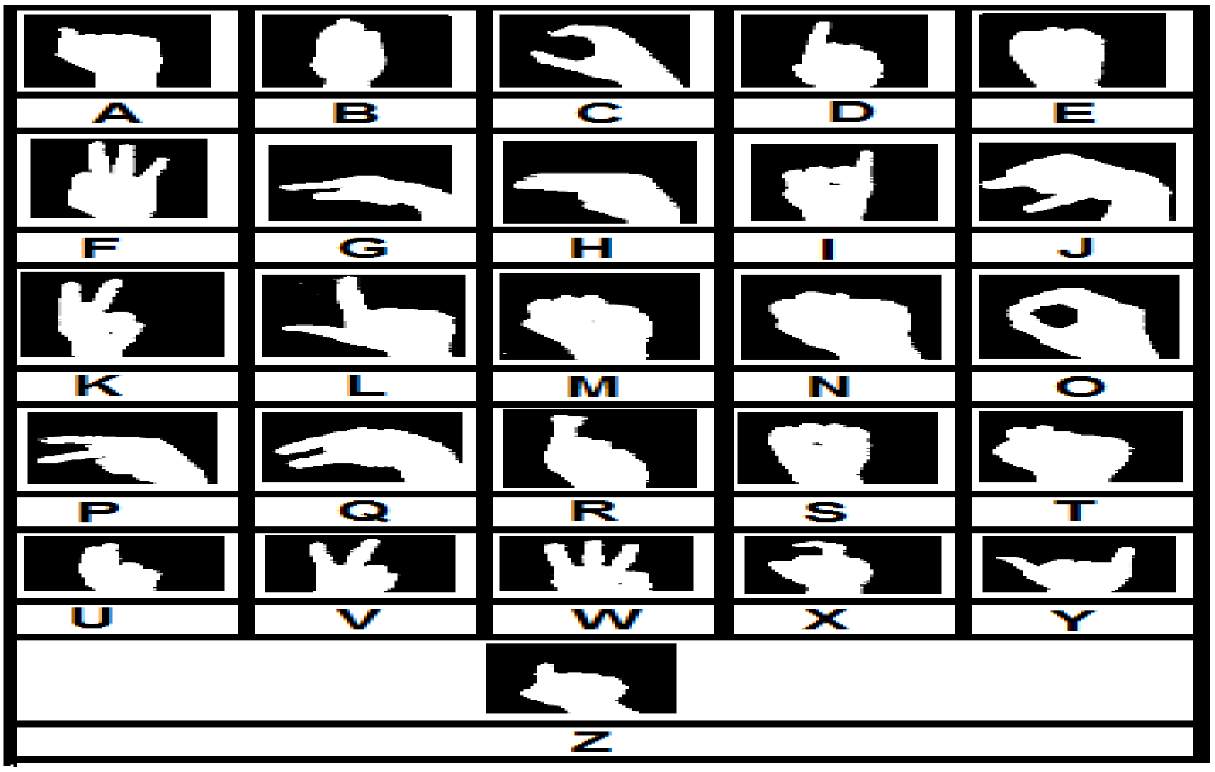
```
# Part 1 - Building the CNN
#importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense, Dropout
from keras import optimizers

# Initialing the CNN
classifier = Sequential()

# Step 1 - Convolutio Layer
classifier.add(Convolution2D(32, 3,  3, input_shape = (64, 64, 3), activation = 'relu'))

#step 2 - Pooling
classifier.add(MaxPooling2D(pool_size =(2,2)))

# Adding second convolution layer
classifier.add(Convolution2D(32, 3,  3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size =(2,2)))

#Adding 3rd Concolution Layer
classifier.add(Convolution2D(64, 3,  3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size =(2,2)))


#Step 3 - Flattening
classifier.add(Flatten())

#Step 4 - Full Connection
classifier.add(Dense(256, activation = 'relu'))
classifier.add(Dropout(0.5))
classifier.add(Dense(26, activation = 'softmax'))

#Compiling The CNN
```

CNN Model

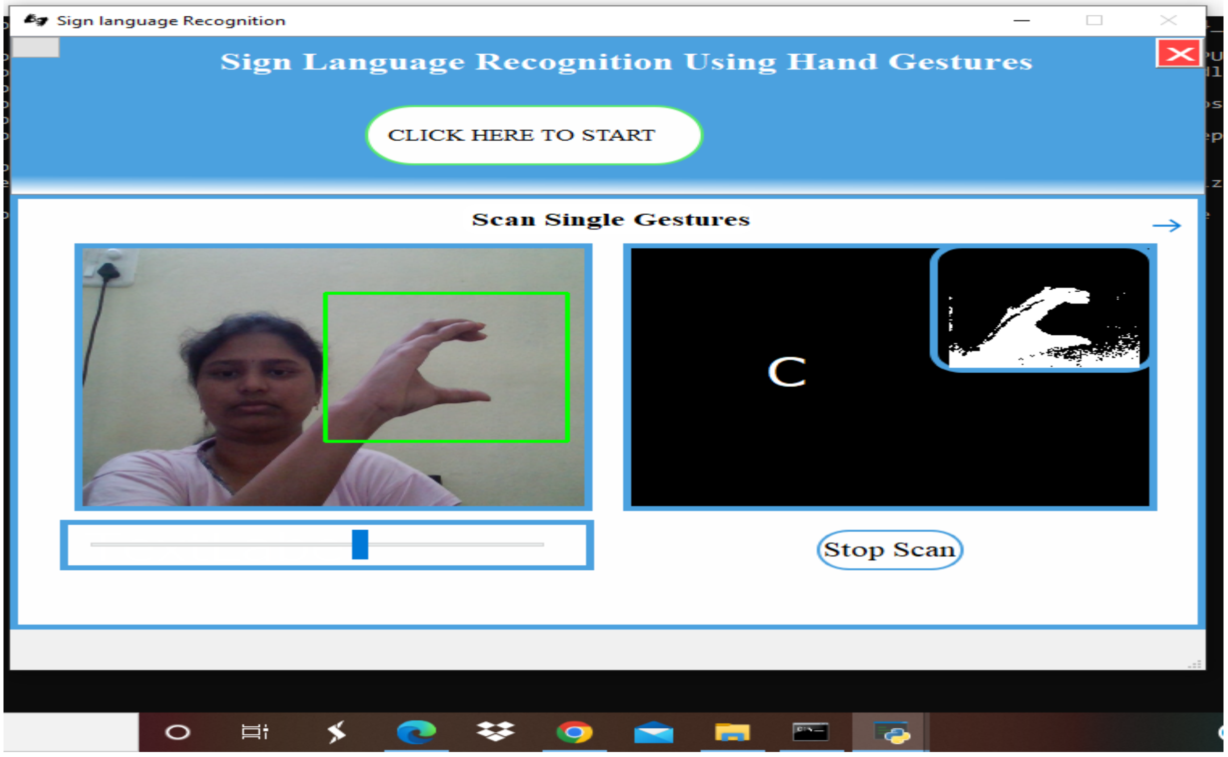


Sign language Recognition
Sign Language Recognition Using Hand Gestures
CLICK HERE TO START
R

Home Screen Page



Showing Gestures


Sign language Recognition
Sign Language Recognition Using Hand Gestures
CLICK HERE TO START
Scan Single Gestures
C
Stop Scan

Live Translation of Gesture to Text

SYSTEM TESTING

6. SYSTEM TESTING

6.1. INTRODUCTION

Testing is the process of finding difference between the expected behaviour specified by the system models and the observed behaviour of the system.

Software Testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for III planned through testing.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability.

Software Testing can be broadly classified into two types. They are:

1. **Manual Testing**: Manual testing includes testing software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behaviour or bug.
2. **Automation Testing**: Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses other software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

TESTING OBJECTIVES

These are several rules that can save as testing objectives:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- To ensure that during operation the system will perform as per specification.
- To make sure that the system meets the user requirements during operation.
- To make sure that during the operation, incorrect input, processing and output will be detected.
- To see that when correct inputs are fed to the system, the outputs are correct.
- To verify that the controls incorporated in the same system as intended.

TEST LEVELS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or darkness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

TESTING ACTIVITIES

- Inspecting a component, this finds faults in an individual component through the manual inspection of its source code.
- Unit testing, which finds faults by isolating an individual component using test stubs and drivers and by exercising the components using test cases.
- Integration testing, which finds faults by integrating several components together.
- System testing, which focuses on the complete system and its functional requirements and non-functional requirements and its target environment.

Testing is the phase where the errors remaining from all the previous phases must be detected. Hence testing performs a very critical role for quality assurance and for ensuring the reliability of software of providing the software with a set of test outputs and observing if the software behaves as expected, then that conditions under which the failure occurs or needed for debugging and correction.

Error

The term error is used in two different ways. It refers to the disciplinary between a computed, observed or measured value and true, specified or theoretically correct value. Error is also used to refer to human actions that result in software containing or fault. This definition is quite general and encompasses all the phases.

Failure

Failure is the inability of the system or the components to perform a required function according to its specifications. In other words, a failure is manifestation of error. But the mere presence of an error may not cause a failure presence of an error implies that a fault must be present in the system. However, the presence of a default does not imply that a failure must occur.

A test case is the triplet [i, s, o] where i stand for the data input to the system, s is the state of the system at which the data is input, and o is the expected output of the system. A test suite is the set of all test cases with a given software product is to be tested.

Verification and Validation

Verification is the process of determining whether one phase of the software product confirm to its previous. Verification is concerned with the phase containing the errors.

6.2. TESTING METHODS

A testing strategy is a road way, giving how to conduct a test. Our testing strategy is flexible enough to promote customization that be necessary in due course of development process. For instance, during code we find that a change in design. We maintain a change log and refer to it at appropriate time during testing. The first approach is what is known as black box testing and the second is called white box testing. We'll be using a mixed approach, more popularly known as sandwich testing. We apply white box testing techniques to ascertain the functionalities top-down and then we use black box testing to demonstrate everything that runs as expected.

A strategy for testing integrates software test case design methods into a well-planned series of steps that result in the construction of software. The objective of the testing is to reduce risks inherent in the computer software.

The various testing methods are:

- White box testing
- Black box testing

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purposeful. It is used to test areas that cannot be reached from a black box level.

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independent paths in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here, the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

Black box testing attempts to find errors in following areas or categories, incorrect or missing functions, interface errors, errors in data structures, performance errors and initialization and termination errors. Here all the input data must match the data type to become a valid entry.

6.3. TESTING TYPES

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.

Unit testing is essential for the verification of the code produced during the coding phase and the goal is to test the internal logic of the module or program. In the generic code project, the unit testing is done during the coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested whether they are rightly connected or not.

Integration Testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

All the tested modules are combined into sub-system, which are then tested. The goal is to see if the modules properly integrated and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

Validation Testing:

This testing concentrates on confirming that the software is error-free in all aspects. All the specified validations are verified and the software is subjected to hard core testing. It also aims to determine the degree of deviation that exists in the software design from the specification; they are listed out and are corrected.

System Testing:

System tests are designed to validate fully developed system with a view to assure that it meets specified requirements.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user.

6.4 TEST CASES

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Test Case 1:

Test Case	Here user has to press click to start to get the gesture
Test Requirement	To check whether the system is redirecting to correct page.
Input Description	The user need to click on the click here to start tab
Sample Input	Click event
Sample Output	Redirected to required page
Expected Output	Redirecting to required page
Result	System execution successful

Test Case 2:

Test Case	Here user need to perform the any particular gesture
Test Requirement	The system should be able to recognise the correct gesture and produce text
Input Description	The user performs the any gesture of the alphabet
Sample Input	User performing the gesture
Sample Output	Detected correct gesture produce text
Expected Output	Detecting the correct gesture produce text
Result	System executed successfully

CONCLUSION

7.CONCLUSION

There are no pre-registrations required to use this system. Anyone can access the system for free and get benefitted with application.

This project was undertaken to solve the underlying issue that hearing and speech impaired people face. They often don’t even stand a chance in the competitive global arena because of communication hurdles.

The need for an interpreter is eradicated and the smooth flowing of a conversation is well developed. The application provides the necessary platform to communicate with much ease and gives them the ability to interact without any external help.

FUTURE SCOPE

8. FUTURE SCOPE

- It can be integrated with various search engines and texting application such as google, WhatsApp. So that even the illiterate people could be able to chat with other persons, or query something from web just with the help of gesture.
- This project is working on image currently, further development can lead to detecting the motion of video sequence and assigning it to a meaningful sentence with TTS assistance.
- Presently our system can be applicable to only ASL but can be extended to various sign language recognition system and various gestures can also be added.
- The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying alphabets labels it will be more appropriate to display sentences as more appropriate translation of language. This also increases readability. The scope of different sign languages can be increased. More training data can be added to detect the letter with more accuracy.

BIBLIOGRAPHY

9. BIBLIOGRAPHY

- https://docs.python.org/3.10/installing/index.html
- https://numpy.org/install/
- https://matplotlib.org/
- https://opencv.org/
- https://keras.io/getting_started/
- https://en.wikipedia.org/wiki/TensorFlow
- https://en.wikipedia.org/wiki/Convolutional_neural_nework
- https://docs.python.org/3/library/tkinter.html

APPENDIX

10.APPENDIX

10.1 Introduction to python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This **tutorial** gives enough understanding on **Python programming** language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python

Following are important characteristics of **Python Programming** −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

10.2 Introduction to Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No.	Operator & Description
1	Button The Button widget is used to display buttons in your application.
2	Canvas The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	Checkbutton The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	Entry

	The Entry widget is used to display a single-line text field for accepting values from a user.
5	Frame The Frame widget is used as a container widget to organize other widgets.
6	Label The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	Listbox The Listbox widget is used to provide a list of options to a user.
8	Menubutton The Menubutton widget is used to display menus in your application.
9	Menu The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	Message The Message widget is used to display multiline text fields for accepting values from a user.
11	Radiobutton The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	Scale The Scale widget is used to provide a slider widget.
13	Scrollbar

	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	Text The Text widget is used to display text in multiple lines.
15	Toplevel The Toplevel widget is used to provide a separate window container.
16	Spinbox The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	PanedWindow A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	LabelFrame A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	tkMessageBox This module is used to display message boxes in your applications.

10.3 Introduction to Neural Networks

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible resultwithout needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

Areas of Application Followings are some of the areas, where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications. Speech Recognition Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand. Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area. Following ANNs have been used for speech recognition

10.4 Introduction to Deep Learning

Deep-learning networks are distinguished from the more commonplace single-hiddenlayer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as "deep" learning. So deep is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven't heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, highdimensional data sets with billions of parameters that pass through nonlinear functions.Above all, these neural nets are capable of discovering latent structures within unlabeled, unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing and clustering the world's raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to.

For example, deep learning can take a million images, and cluster them according to their similarities: cats in one corner, ice breakers in another, and in a third all the photos of your grandmother. This is the basis of so-called smart photo albums.

Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms. Given that feature extraction is a task that can take teams of data scientists years to accomplish, deep learning is a way to circumvent the chokepoint of limited experts. It augments the powers of small data science teams, which by their nature do not scale.

When training on unlabeled data, each node layer in a deep network learns features automatically by repeatedly trying to reconstruct the input from which it draws its samples, attempting to minimize the difference between the network's guesses and the probability distribution of the input data itself. Restricted Boltzmann machines, for examples, create so-called reconstructions in this manner.

In the process, these neural networks learn to recognize correlations between certain relevant features and optimal results – they draw connections between feature signals and what those features represent, whether it be a full reconstruction, or with labeled data. A deep-learning network trained on labeled data can then be applied to unstructured data, giving it access to much more input than machine-learning nets.

10.5 Introduction to PyQt5

PyQt is a python binding of the open-source widget-toolkit Qt, which also functions as a cross-platform application development framework. Qt is a popular C++ framework for writing GUI applications for all major desktop, mobile, and embedded platforms (supports Linux, Windows, MacOS, Android, iOS, Raspberry Pi, and more).

PyQt is a free software developed and maintained by Riverbank Computing, a company based in England, whereas Qt is developed by a Finnish firm called The Qt Company.

Features of PyQt:

Learn PyQt which consists of more than six hundred classes covering a range of features such as

- Graphical User Interfaces
- SQL Databases
- Web toolkits
- XML processing
- Networking

These features can be combined to create advanced UIs as well as standalone applications. A lot of major companies across all industries use Qt. Some examples are LG, Mercedes, AMD, Panasonic, Harman, etc.

Components and Widgets:

There is a large number of widgets available in PyQt for creating GUI apps. However, with PyQt5, there has been a reshuffling of classes into different modules and revisions in the licenses.

Therefore, it’s crucial to have a high-level view of the structure of PyQt5. In this section, you will see how PyQt5 is organized internally and learn about the different modules, libraries, and API classes provided by PyQt5.

These are the fundamental modules used by Python’s Qt binding, specifically PyQt5.

- **Qt**: It combines all the classes/modules mentioned below into a single module. It considerably increases the memory used by the application. However, it’s easier to manage the framework by only importing one module.

- **QtCore**: Contains the core non-graphical classes used by other modules. This is where the Qt event loop, signals, and slot-connectivity, etc. are implemented.
- **QtWidgets**: Contains most of the widgets available in Pyqt5.
- **QtGui**: Contains GUI components and extends the QtCore module.
- **QtNetwork**: Contains classes used to implement network programming through Qt. It supports TCP servers, TCP sockets, UDP sockets, SSL handling, network sessions, and DNS lookups.
- **QtMultimedia** provides low-level multimedia functionality.
- **QtSql**: implements database integration for SQL databases. Supports ODBC, MySQL, Oracle, SQLite, and PostgreSQL.

PyQt5 Widgets

Here is a list of the most frequently used widgets in PyQt5

- **QLineEdit**: This is an input field which allows one line of text to be entered by the user.

  ```
  line = QLineEdit()
  ```

- **QRadioButton**: This is an input field with a selectable button, similar to the radio buttons in html.

  ```
  rad = QRadioButton("button title")
  rad.setChecked(True)  #to select the button by default.
  ```

- **QComboBox**: It is used to display a dropdown menu with a list of selectable items.

  ```
  drop = QComboBox(w)
  drop.addItems(["item one", "item two", "item three"])
  ```

- **QCheckBox**: Displays a selectable square box in front of the label that is ticked if selected, similar to radio buttons.

  ```
  check = QCheckBox("button title")
  ```

- **QMenuBar**: it displays a horizontal menu bar at the top of a window. You can only add objects of the QMenu class to this bar. Those QMenu objects can further contain strings, QAction objects or other QMenu objects.

- **QToolBar**: It is a horizontal bar or pane which can be moved within the window. It may contain buttons and other widgets.

- **QTab**: it is used to break down the contents of a window into multiple pages that can be accessed through different tabs on top of the widget. It consists of two sections: the tab bar and the tab page.

- **QScrollBar**: It is used to create scroll bars which allow the user to scroll up and down within a window. It consists of a movable slider, a slider track, and two buttons to scroll the slider up or down.

  ```
  scroll = QScrollBar()
  ```

- **QSplitter**: Splitters are used to separate the contents of a window so that the widgets are grouped properly and do not appear cluttered. QSplitter is one of the primary layout handlers available in PyQt5 and is used to split the content both horizontally and vertically.

- **QDock**: A dock widget is a sub-window with two properties:
  - It can be moved within the main window and
  - It can be docked outside the parent window to another location on the screen.