TABLE OF CONTENTS

ABSTRACT

Gesture recognition has emerged as a crucial area of research in the development of assistive communication systems, particularly for individuals with hearing or speech impairments. This project, titled **"Gesture Recognition using CNN with OpenCV"**, presents an intelligent solution that interprets hand gestures and translates them into meaningful words or phrases using advanced machine learning techniques.

The system is designed using **Convolutional Neural Networks (CNN)** in combination with **OpenCV** for real-time video processing and gesture tracking. The application captures hand gestures via webcam or video input, extracts key features using image processing and pose estimation, and maps them to predefined words using a trained deep learning model. A Random Forest classifier is also implemented and compared for performance evaluation.

The experimental results demonstrate high accuracy in gesture classification, enabling precise recognition of dynamic and static hand gestures. This model has been trained on custom datasets, and the performance is visualized using comparison graphs for multiple models.

The application offers real-time translation of sign language from both static images and live webcam feed, enhancing user interactivity. This project contributes significantly to human-computer interaction and holds promise for integration into mobile or embedded systems for broader accessibility.

INTRODUCTION

CHAPTER – 1

INTRODUCTION

1.1 Introduction

Gesture recognition is a subfield of computer vision and human-computer interaction (HCI) that involves the interpretation of human gestures through algorithms. In particular, hand gestures are a powerful form of non-verbal communication that can be leveraged to bridge the gap between humans and machines. With advancements in artificial intelligence and image processing, gesture recognition is now widely used in various domains including virtual reality, robotics, smart home devices, and assistive communication systems.

This project focuses on developing a system that can recognize hand gestures using Convolutional Neural Networks (CNN) and OpenCV, and convert them into words or phrases in real-time. The system captures hand gestures through video or webcam feed, processes the input to detect key points on the hand using OpenCV, and classifies the gesture using a trained deep learning model.

1.2 Objective

The primary objectives of this project are:

- To develop a robust hand gesture recognition system using CNN and OpenCV.
- To convert detected gestures into meaningful words or phrases in real-time.
- To support assistive communication for individuals with hearing or speech impairments.

- To compare the performance of CNN with traditional machine learning algorithms like Random Forest.
- To visualize the system's accuracy and efficiency using comparison graphs.

1.3 Problem Statement

Despite the rapid growth in gesture-based applications, many real-time gesture recognition systems suffer from issues like poor accuracy, high latency, and limited adaptability to varying lighting or background conditions. Additionally, assistive technologies for the speech and hearing impaired often require expensive hardware or are limited in their vocabulary recognition.

This project aims to solve these problems by creating a lightweight and efficient gesture recognition system that operates using a simple webcam and open-source libraries. By employing CNN and OpenCV, the model is trained to accurately detect and classify hand gestures from various angles and lighting conditions, making it accessible, scalable, and practical for real-world deployment.

1.4 Algorithms and Techniques

The system leverages the following algorithms and techniques:

1. Convolutional Neural Network (CNN):

Used for learning spatial hierarchies of features from gesture images. CNN effectively captures edges, shapes, and gesture patterns through multiple convolution and pooling layers.

2. OpenCV:

An open-source computer vision library used for real-time image acquisition, hand detection, contour processing, and keypoint extraction (e.g., using MediaPipe or convex hull methods).

3. Random Forest Classifier:

Implemented as a baseline traditional machine learning model for gesture classification, offering a comparison against CNN's performance.

4. Image Preprocessing:

Techniques like grayscale conversion, thresholding, and Gaussian blurring are applied to enhance the image quality and reduce noise.

5. Feature Extraction and Landmark Detection:

Key features from the hand are extracted using OpenCV and fed into the classifier for accurate gesture recognition.

LITERATURE SURVEY

CHAPTER – 2

LITERATURE SURVEY

Gesture recognition has become an essential domain in the field of Human-Computer Interaction (HCI), providing users with an intuitive and natural way to communicate with machines. Early systems relied on hardware-based solutions such as gloves and sensors to detect hand movements, which, although accurate, were not practical for widespread use due to their cost and complexity. As camera technology and computer vision improved, researchers started focusing on vision-based gesture recognition, which is more convenient and cost-effective. OpenCV, an open-source computer vision library, emerged as a powerful tool for image and video analysis, enabling real-time gesture recognition using simple camera setups.

Convolutional Neural Networks (CNNs) have revolutionized the field of image classification and pattern recognition. CNNs are particularly effective in detecting spatial hierarchies and patterns in images, making them suitable for gesture recognition tasks. Literature suggests that CNNs outperform traditional machine learning models in tasks requiring high-level feature extraction. Researchers such as Simonyan and Zisserman (2014) proposed deep CNN architectures that significantly improved image recognition performance, setting a foundation for real-time gesture interpretation systems using video data.

Several studies have explored combining CNNs with OpenCV for gesture recognition. For instance, work by Molchanov et al. (2015) demonstrated the use of 3D CNNs for dynamic hand gesture recognition using temporal and spatial data. Similarly, the integration of OpenCV for hand segmentation, edge detection, and pose estimation has been shown to improve gesture tracking

accuracy in cluttered backgrounds and varied lighting conditions. By leveraging OpenCV's efficient image processing capabilities, real-time recognition becomes feasible even on standard hardware.

Recent systems have incorporated mediapipe or similar pose estimation tools to track hand landmarks more precisely, enabling detection of fine-grained gestures. These landmarks are then passed through CNN-based classification models that have been trained on diverse gesture datasets. Literature emphasizes the importance of dataset diversity and model generalization, especially for sign language recognition and multilingual gesture systems. Comparison between classifiers, such as Random Forests and CNNs, shows that while traditional classifiers perform well with structured data, deep learning models provide superior results when trained on raw image input.

In conclusion, the convergence of CNN architectures with OpenCV has created a robust framework for real-time gesture recognition. This literature reveals that incorporating deep learning with efficient video processing techniques addresses the challenges of accuracy, latency, and scalability. The current system, as demonstrated in this project, builds upon these advances to create a responsive and accurate gesture-to-text translation system, supporting assistive technologies and HCI applications alike.

SYSTEM ANALYSIS

CHAPTER – 3

SYSTEM ANALYSIS

3.1 Existing System

The existing systems for gesture recognition primarily rely on traditional image processing techniques or rule-based algorithms, which often require manual feature extraction. These systems struggle with variations in lighting conditions, background noise, hand orientations, and dynamic gestures. Many rely on wearable devices or sensor-based gloves, making them inconvenient and costly for everyday use. Additionally, some systems are limited to recognizing only a small set of predefined static gestures, lacking adaptability to real-time environments. The absence of deep learning integration limits their ability to generalize across diverse datasets and user profiles.

3.2 Proposed System

The proposed system introduces a **deep learning-based gesture recognition model using CNN integrated with OpenCV**, designed for real-time and robust performance. Unlike the existing systems, this solution utilizes **Convolutional Neural Networks** to automatically extract spatial features from hand gesture images or video frames, improving recognition accuracy. OpenCV is employed for real-time image acquisition, preprocessing, and hand tracking. The system processes gestures from live webcam input or recorded videos, identifies key hand landmarks using computer vision, and translates them into meaningful text using trained CNN models. This approach significantly enhances user interaction, supports a wider vocabulary of gestures, and operates effectively in varied environments. It aims to provide a scalable and cost-effective solution for assistive communication, especially for individuals with speech or hearing disabilities.

SYSTEM REQUIREMENTS

CHAPTER – 4

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

- Processor: Intel Core i5/i7 or AMD Ryzen 5/7 (or higher)
- RAM: Minimum 8GB (Recommended: 16GB or higher)
- Storage: Minimum 250GB SSD (Recommended: 512GB SSD or higher)
- GPU (if required for ML models): NVIDIA GTX 1650 or higher
- Network: High-speed internet connection for cloud interactions

4.2 Software Requirements

- Operating System: Windows 10/11, Ubuntu 20.04/22.04, or macOS
- Programming Language: Python 3.8+
- Frameworks & Libraries: TensorFlow, PyTorch, NumPy, Pandas, SciPy
- Virtualization Tools: Docker, Kubernetes, VMware/VirtualBox
- Cloud Platforms: AWS, Google Cloud, Microsoft Azure (any federated cloud setup)
- Database: PostgreSQL, MySQL, or MongoDB
- Web Framework (if using a web interface): Django/Flask
- Other Tools: Git, Jupyter Notebook, VS Code/PyCharm

PROPOSED SOLUTIONS

CHAPTER – 5

PROPOSED SOLUTIONS

5.1 Overview

The proposed system aims to recognize hand gestures and convert them into corresponding textual interpretations using **Convolutional Neural Networks (CNN)** and **OpenCV**. It is specifically designed to assist individuals with communication impairments by enabling real-time sign language recognition. The system leverages webcam or video input, processes hand gestures through image preprocessing techniques, and uses trained deep learning models to classify the gestures.

The core objectives of this solution are:

- Accurate gesture recognition using CNN-based deep learning models.
- Real-time prediction from webcam or video feed using OpenCV.
- Enhanced communication through sign-to-text conversion.

5.2 Dataset Description

The dataset used for training and evaluation consists of hand gesture images and videos annotated with corresponding sign language meanings. It includes:

- **Static images** of hand gestures representing words or phrases.

- **Video frames** extracted and labeled for dynamic gesture recognition.
- **Webcam-captured data** to simulate real-world scenarios.

  Each image is preprocessed to isolate hand regions using techniques like background subtraction, skin color filtering, and contour detection. Landmarks from the hand (as shown in the images) are also used as key features for training.

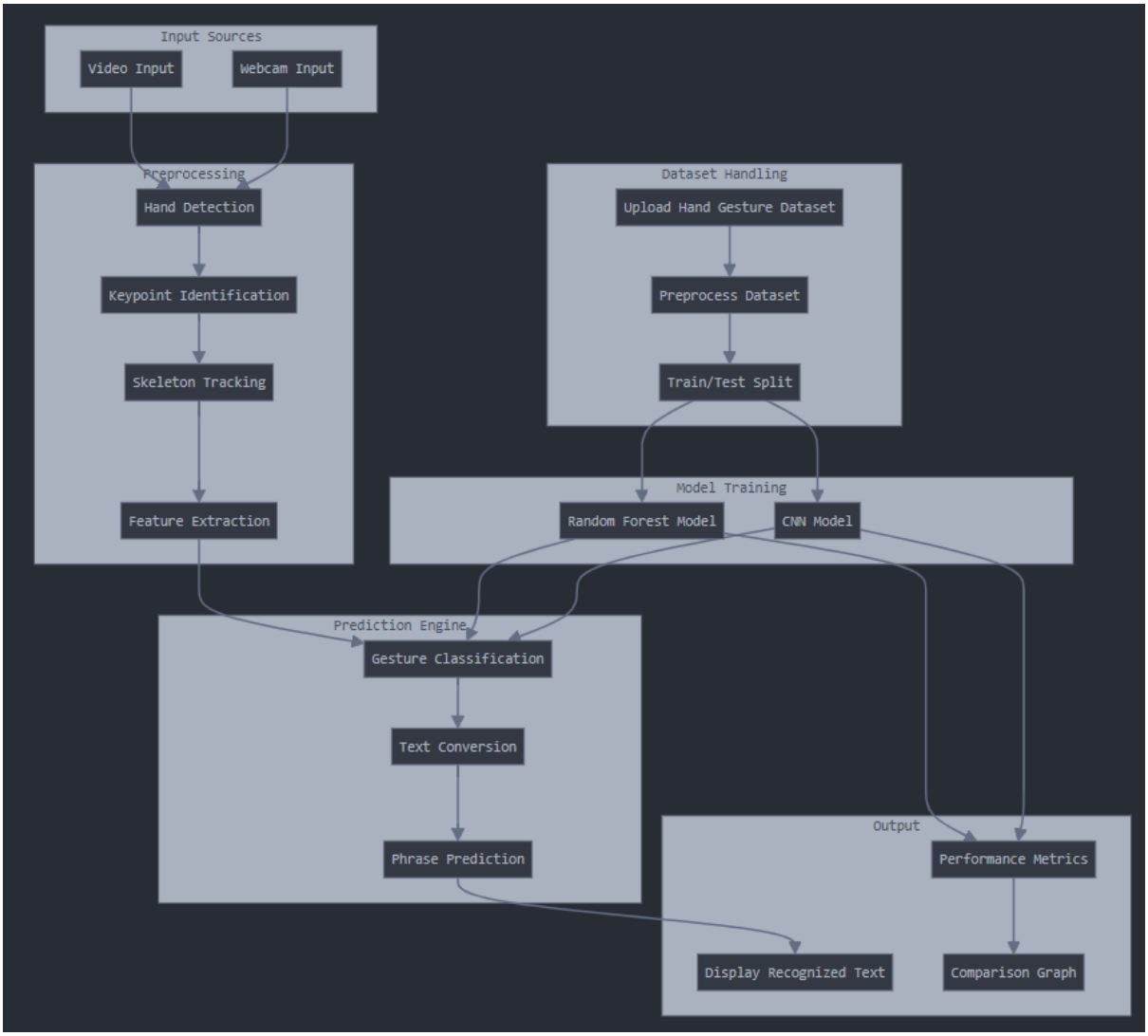


Fig : 5.1 Flowchart of Proposed System

5.3 Process

The overall workflow of the system is divided into the following stages:

Step 1: Data Collection & Preprocessing

- Capture hand gesture images or video from webcam or dataset.
- Detect hand region using **OpenCV** techniques (e.g., color thresholding, hand segmentation).
- Extract key hand landmarks using **mediapipe** or **OpenCV contour/edge detection**.
- Resize and normalize images for input to CNN.

Step 2: Model Training

- Build and train a **CNN model** on preprocessed hand gesture images.
- Train a **Random Forest** model for comparison and hybrid use.
- Use a split of training (80%) and testing (20%) datasets to validate accuracy.

Step 3: Real-Time Gesture Detection

- Capture live video feed from webcam.
- Detect and track hand movements in real-time.
- Predict the gesture using the trained CNN model.
- Display the output on-screen with bounding box and prediction label (e.g., *"Predicted As: A LOT"*, *"Are you hiding something"*).

Step 4: Comparison and Evaluation

- Evaluate the performance of CNN vs. traditional ML models (Random Forest).

- Use accuracy, loss, and Mean Squared Error (MSE) for comparison metrics.
- Visualize comparison results via graphs (as shown in earlier figures).

5.4 Conclusion

This project demonstrates a practical and effective method for hand gesture recognition using CNN and OpenCV. By integrating deep learning with real-time video processing, the system successfully converts visual hand gestures into meaningful textual content. It is especially valuable in creating assistive tools for differently-abled individuals, enhancing communication and accessibility. Future enhancements may include deploying the model on edge devices or integrating it into mobile applications for wider usage.

SYSTEM DESIGN

CHAPTER – 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

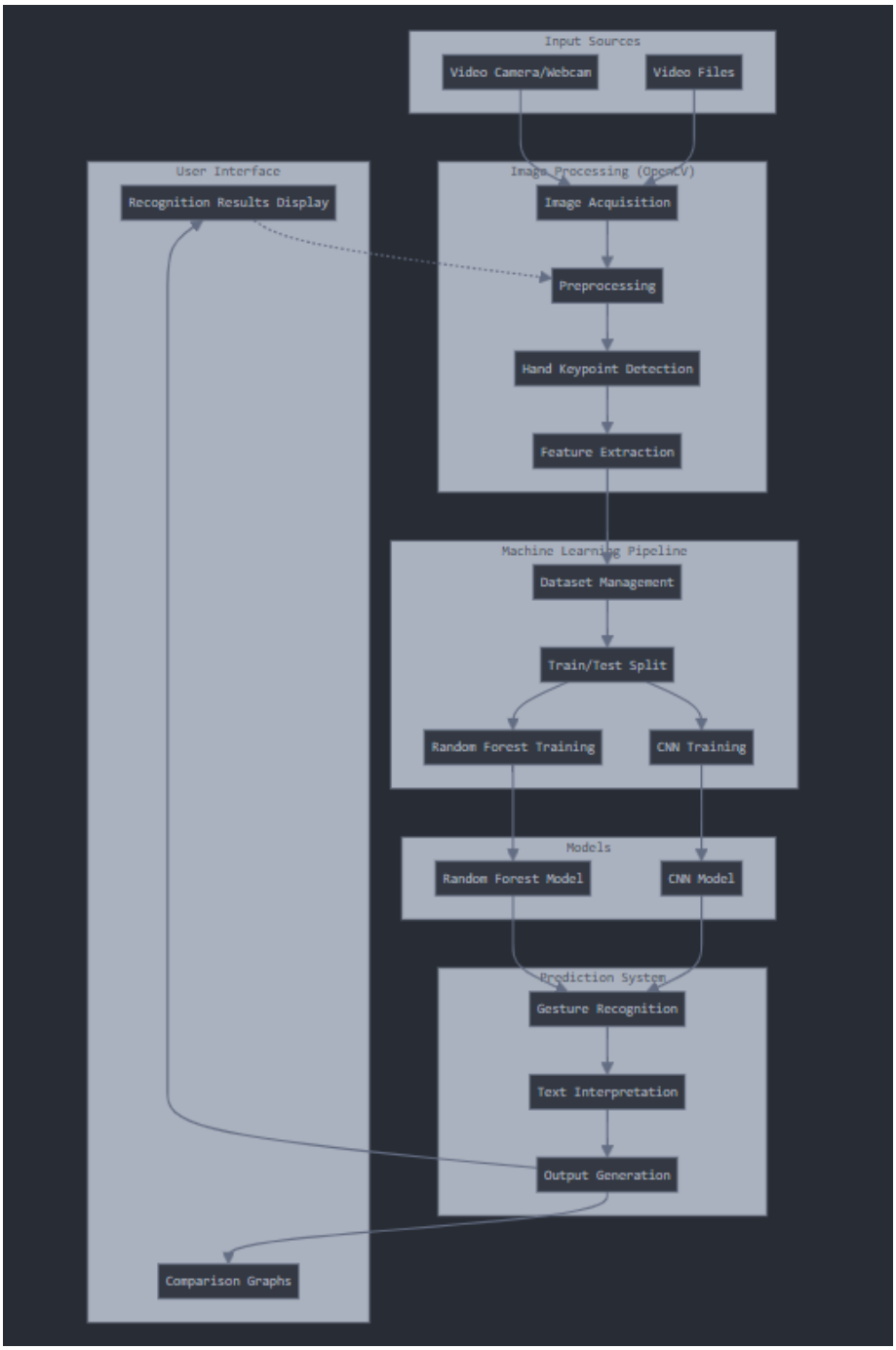
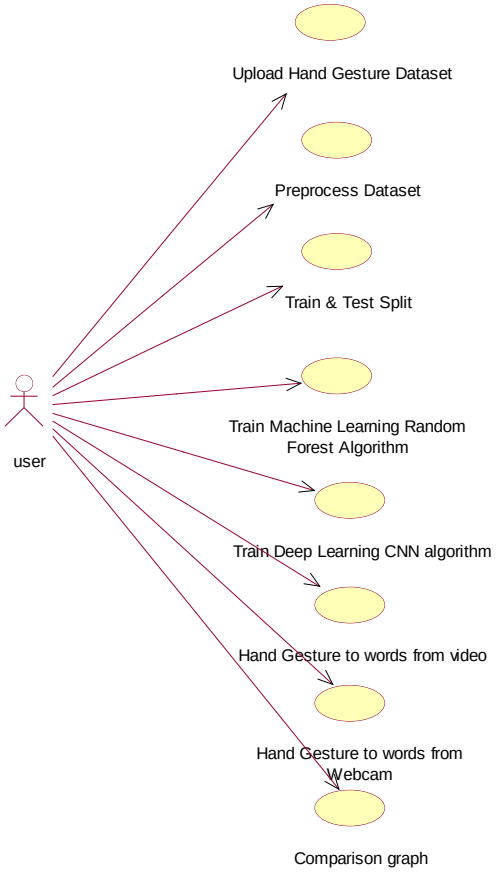


Fig 6.1 : System Architecture

6.2 UML DIAGRAMS

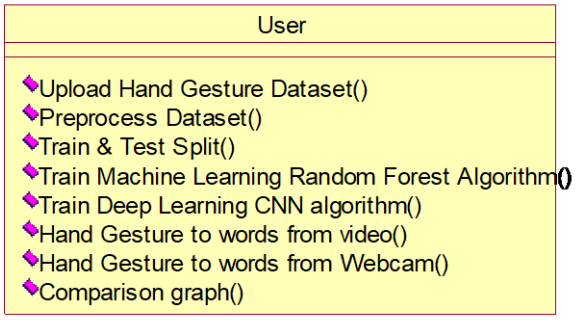
6.2.1 USE CASE DIAGRAM

The Use Case Diagram represents the system's interactions with external users. It identifies the different actors (such as users and the system) and their respective roles, ensuring clarity on how the Whisper model processes speech input and provides transcriptions.



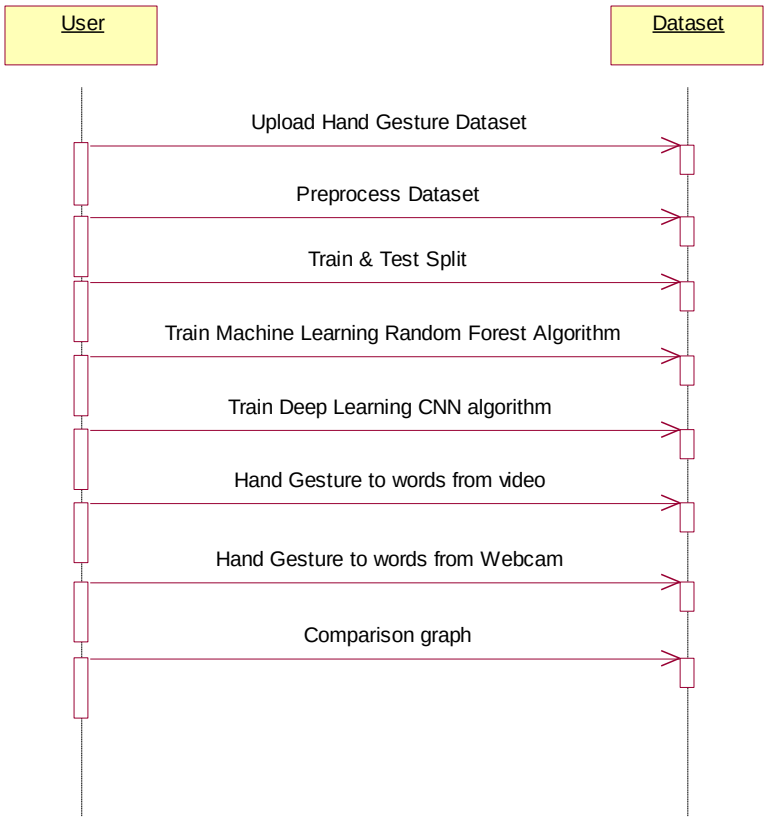
6.2.2 CLASS DIAGRAM

The Class Diagram depicts the structure of the transcription system in terms of its classes, attributes, and relationships. It outlines how different components, such as the audio processor, transcription engine, and text formatter, interact within the system.

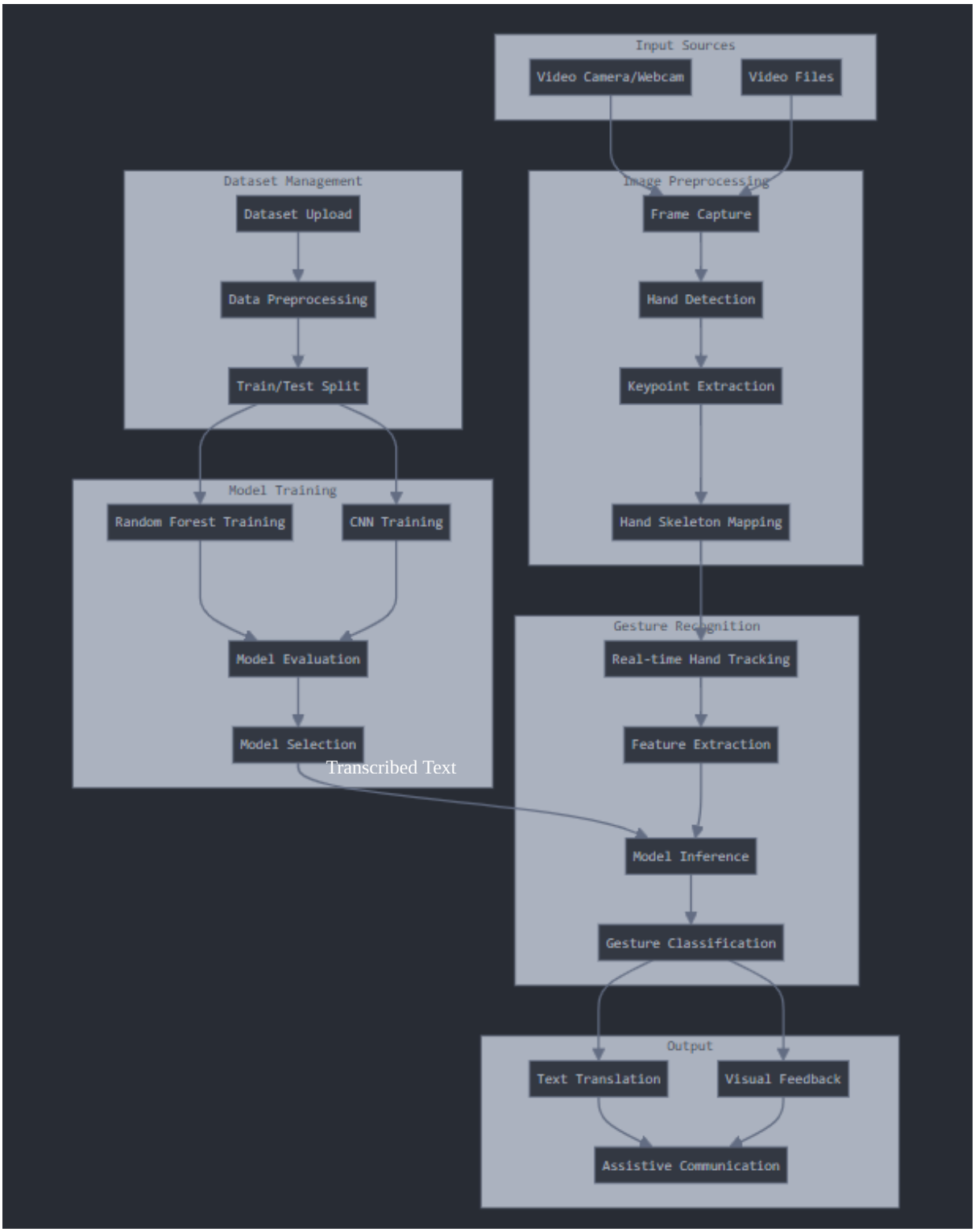


6.2.3 SEQUENCE DIAGRAM

The Sequence Diagram illustrates the step-by-step flow of interactions between system components over time. It demonstrates how an audio file is processed, sent to the Whisper model for transcription, and returned as text output.



6.2.4 DATA FLOW DIAGRAM



SOFTWARE ENVIRONMENT

CHAPTER – 7

SOFTWARE ENVIRONMENT

What is Python :

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)

- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

6. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

7. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

8. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

9. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

10. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages :

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform

data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this iseasy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used

indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with

some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not

been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term "Machine Learning" in 1959 and defined it as a "Field of study that gives computers the capability to learn without being explicitly programmed".

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these,

never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular

language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.
Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

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

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

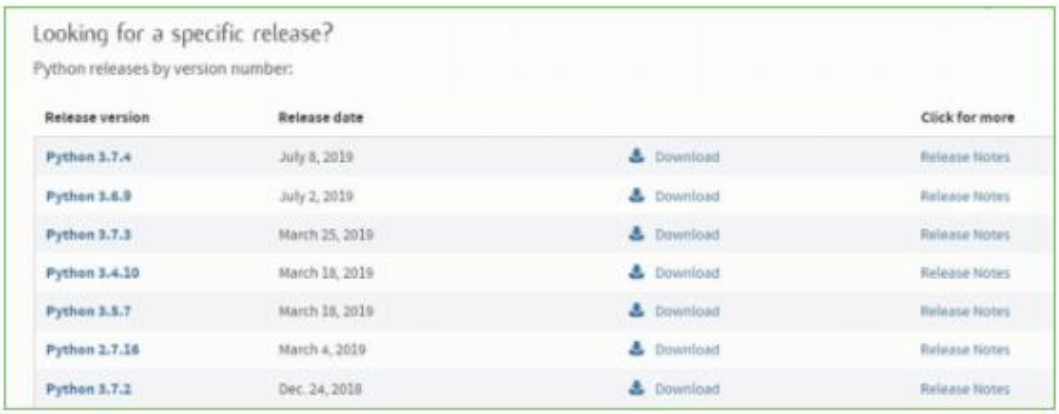
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web **browser.** OR Click on the following **link:** **https://www.python.org**

Now, check for the latest and the correct version for your operating system.

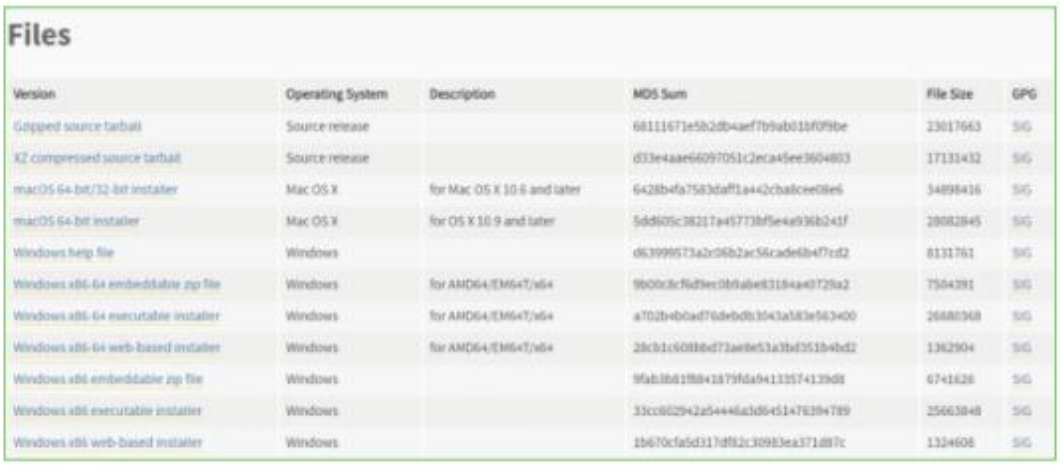
Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab016f0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cbalcee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d639995713a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c3cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76deb6db3043a583e563400	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c6088bd73ae8e53a3bd3518b8d2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda941335741398d8	6741626	SIG
Windows x86 executable installer	Windows		33cc6029425a5444a3a3d64514763947f89	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c3098 3ea371d87c	1324608	SIG

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

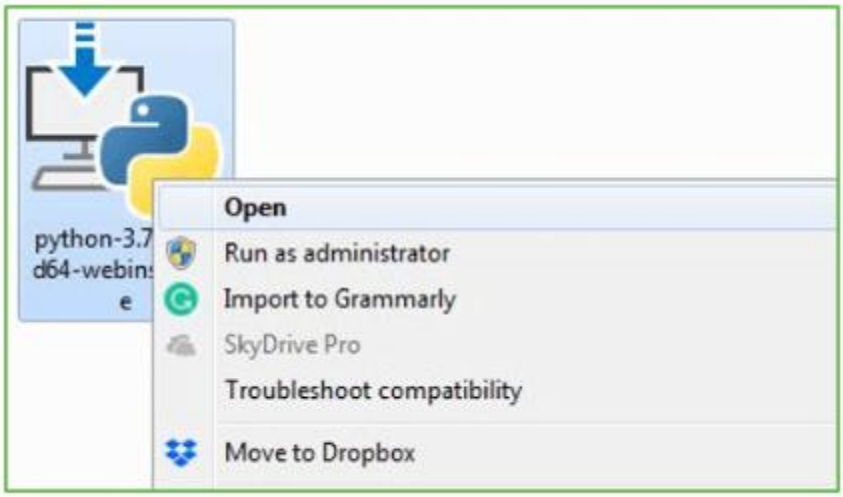
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

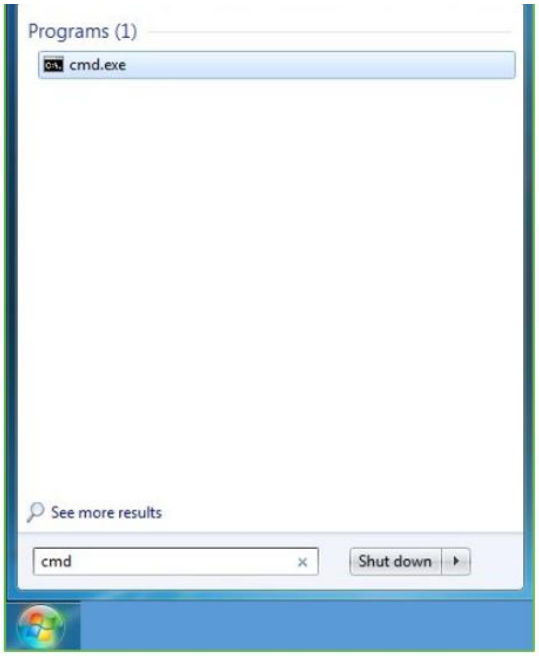
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

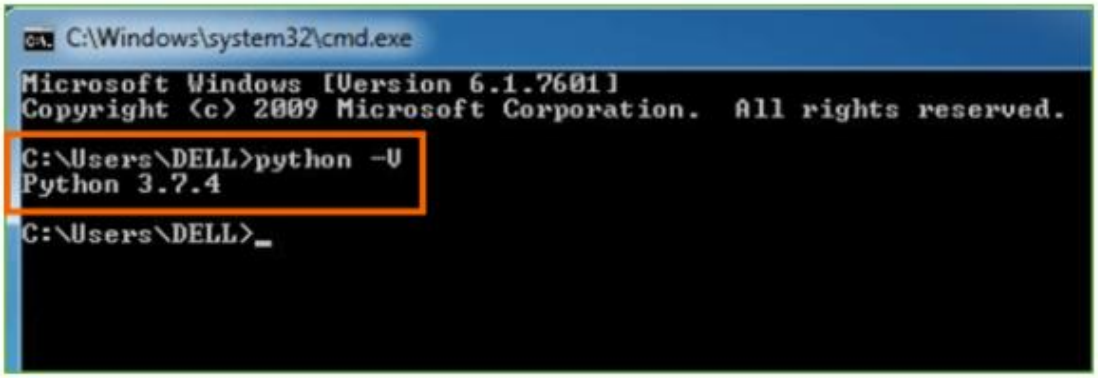
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

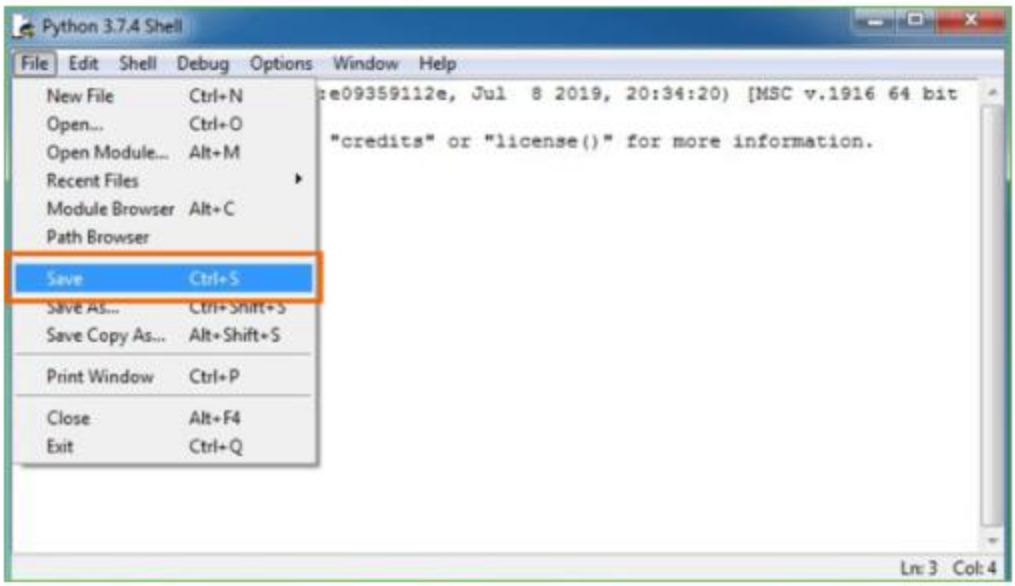
Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print

IMPLEMENTATION

CHAPTER – 8

IMPLEMENTATION

The implementation of the "Gesture Recognition using CNN with OpenCV" project integrates deep learning and computer vision to interpret hand gestures in real time. The system begins by capturing hand gestures through a webcam or pre-recorded video feed, where OpenCV is used for real-time video processing and hand detection. Using Mediapipe or similar libraries, key landmark points of the hand are extracted and passed as input to a Convolutional Neural Network (CNN). The CNN model is trained on a dataset of labeled hand gestures to classify them into corresponding words or phrases. The interface includes functionality to upload datasets, preprocess the data, perform train-test splits, and execute training using both Random Forest and CNN models. Once trained, the model can predict gestures from live input and display the interpreted text, as shown in the predicted output images. A graphical comparison is also presented to evaluate the performance of different models. This system demonstrates a robust and scalable approach to building real-time assistive communication tools.

8.1 MODULES

1. Dataset Collection & Upload

- **Module Name:** Upload Hand Gesture Dataset
- **Description:** Allows users to upload the dataset containing hand gesture images or videos.
- **Screenshot Reference:** First image (GUI with multiple buttons).
- **Tools/Tech:** Tkinter GUI (Python), FileDialog, Custom gesture datasets.

2. Data Preprocessing

- **Module Name:** Preprocess Dataset
- **Description:**
  - Converts raw images into grayscale or RGB.
  - Resizes all images to a fixed dimension.
  - Applies augmentation techniques (rotation, flipping, scaling).
  - Extracts key features using OpenCV (e.g., edge detection, contouring).
- **Tools/Tech:** OpenCV, NumPy, Scikit-image.

3. Train-Test Split

- **Module Name:** Train & Test Split
- **Description:**
  - Divides the dataset into training and testing subsets (e.g., 80-20 split).
  - Ensures balanced class distribution for gesture types.
- **Tools/Tech:** Scikit-learn.

4. Machine Learning Model Training

- **Module Name:** Train ML (Random Forest)
- **Description:**
  - Trains a Random Forest classifier on extracted gesture features.
  - Used as a baseline model for comparison.
- **Tools/Tech:** Scikit-learn.

5. Deep Learning Model Training

- **Module Name:** Train CNN Algorithm

- **Description:**
  - Trains a Convolutional Neural Network on gesture image data.
  - Uses multiple convolutional, pooling, and dense layers for classification.
  - Performance is significantly better than classical ML models.
- **Tools/Tech:** TensorFlow / Keras.

6. Gesture Prediction from Video

- **Module Name:** Hand Gesture to Words from Video
- **Description:**
  - Loads a video file and processes each frame for gesture recognition.
  - Displays predicted gesture as text overlaid on video.
- **Screenshot Reference:** Second image (predicted gesture "A LOT").
- **Tools/Tech:** OpenCV (VideoCapture), Pretrained CNN model.

7. Gesture Prediction from Webcam

- **Module Name:** Hand Gesture to Words from Webcam
- **Description:**
  - Captures real-time video from the webcam.
  - Detects and tracks hand using MediaPipe/OpenCV.
  - Classifies gesture using CNN model.
- **Screenshot Reference:** Third image ("are you hiding something").
- **Tools/Tech:** OpenCV, MediaPipe, Real-time inference using CNN.

8. Gesture Landmark Extraction (Pose Estimation)

- **Description:**
  - Extracts 21 keypoints from the hand for each frame using MediaPipe Hand tracking.
  - Converts hand landmarks into features for CNN classification.
- **Tools/Tech:** MediaPipe, OpenCV.

9. Prediction Output Display

- **Description:**
  - Displays the predicted gesture as a phrase or word in the GUI/video.
  - Uses overlays like cv2.putText() to show results.
- **Tools/Tech:** OpenCV GUI tools.

10. Comparison Graph

- **Module Name:** Comparison Graph
- **Description:**
  - Compares performance metrics (accuracy, precision, F1-score) of ML vs DL models.
  - Visualized using matplotlib (bar or line graph).
- **Tools/Tech:** Matplotlib, Seaborn.
- **Screenshot Reference:** Included in your earlier submissions.

8.2 SOURCE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import cv2
import random
import numpy as np
from keras.utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential, load_model
import pickle
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from keras.callbacks import ModelCheckpoint
from sklearn.preprocessing import StandardScaler
import mediapipe as mp
import copy
import itertools
import os
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import pandas as pd
import matplotlib.pyplot as plt
import time

global filename
global model, sc
global accuracy, precision, recall, fscore
min_detection_confidence = 0.7
min_tracking_confidence = 0.5
mp_hands = mp.solutions.hands
```

```
hands = mp_hands.Hands(static_image_mode=True, max_num_hands=1,
min_detection_confidence=min_detection_confidence,
min_tracking_confidence=min_tracking_confidence)
drawingModule = mp.solutions.drawing_utils

min_detection_confidence = 0.7
min_tracking_confidence = 0.5

global X, Y, labels
global X_train, X_test, y_train, y_test

def getLabel(name):
    index = -1
    for i in range(len(labels)):
        if labels[i] == name:
            index = i
            break
    return index

def calc_bounding_rect(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]
    landmark_array = np.empty((0, 2), int)
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width
- 1)
        landmark_y = min(int(landmark.y * image_height),
image_height - 1)
        landmark_point = [np.array((landmark_x, landmark_y))]
        landmark_array = np.append(landmark_array, landmark_point,
axis=0)
    x, y, w, h = cv2.boundingRect(landmark_array)
    return [x, y, x + w, y + h]


def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]
    landmark_point = []
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width
- 1)
        landmark_y = min(int(landmark.y * image_height),
image_height - 1)
```

```
        landmark_point.append([landmark_x, landmark_y])
    return landmark_point


def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)
    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
    temp_landmark_list =
list(itertools.chain.from_iterable(temp_landmark_list))
    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

def uploadDataset():
    global filename, labels, X, Y, dataset
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n")
    counter = 0
    for root, dirs, directory in os.walk(filename):
        for j in range(len(directory)):
            name = os.path.basename(root)
            if name not in labels:
                labels.append(name.strip())
                counter += 1

    if os.path.exists("model/X.txt.npy"):
        X = np.load('model/X.txt.npy')
        Y = np.load('model/Y.txt.npy')
    else:
        X = []
```

```
        Y = []
        for root, dirs, directory in os.walk(filename):
            for j in range(len(directory)):
                name = os.path.basename(root)
                img = cv2.imread(root+"/"+directory[j])
                img = cv2.flip(img, 1)
                debug_image = copy.deepcopy(img)
                mp_hands = mp.solutions.hands
                hands = mp_hands.Hands(static_image_mode=True,
max_num_hands=1, min_detection_confidence=min_detection_confidence,
min_tracking_confidence=min_tracking_confidence)
                results = hands.process(cv2.cvtColor(img,
cv2.COLOR_BGR2RGB))
                if results.multi_hand_landmarks is not None:
                    for hand_landmarks, handedness in
zip(results.multi_hand_landmarks, results.multi_handedness):
                        brect = calc_bounding_rect(debug_image,
hand_landmarks)
                        landmark_list =
calc_landmark_list(debug_image, hand_landmarks)
                        pre_processed_landmark_list =
pre_process_landmark(landmark_list)
                        pre_processed_landmark_list =
np.asarray(pre_processed_landmark_list)
                        X.append(pre_processed_landmark_list)
                        label = getLabel(name)
                        Y.append(label)
        X = np.asarray(X)
        Y = np.asarray(Y)
        np.save('model/X.txt',X)
        np.save('model/Y.txt',Y)
    print(np.unique(Y, return_counts=True))
    text.insert(END,"Hand Gesture Class labels found in Dataset :
"+str(labels)+"\n\n")
    text.insert(END,"Total Hand Gesture found in dataset :
"+str(counter))

def processDataset():
    global X, Y, sc
    text.delete('1.0', END)
    sc = StandardScaler()
    X = sc.fit_transform(X)
```

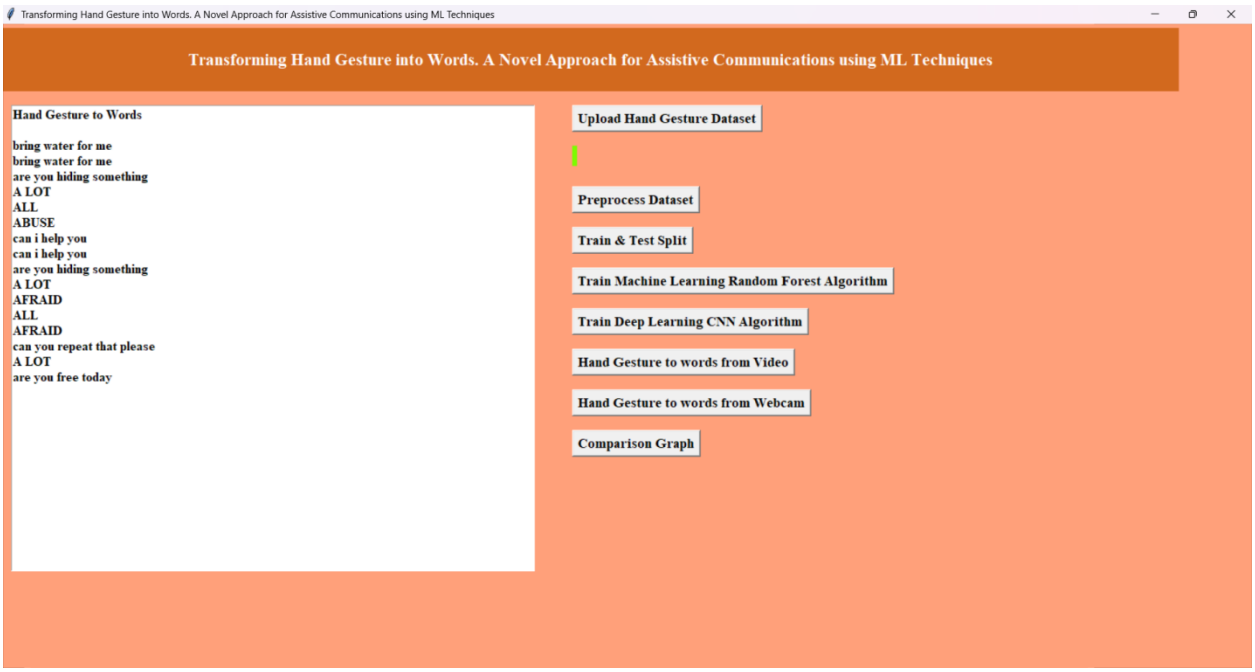
```
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    text.insert(END,"Hand Gesture Features Shuffling & Normalization
processing Completed")

def splitDataset():
    global X_train, X_test, y_train, y_test
    global X, Y
    text.delete('1.0', END)
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2) #split dataset into train and test
    text.insert(END,"80% Hand Gesture Features used for training :
"+str(X_train.shape[0])+"\n")
    text.insert(END,"20% Hand Gesture Features used for testing :
"+str(X_test.shape[0])+"\n\n")
main.config(bg='light salmon')
main.mainloop()
```

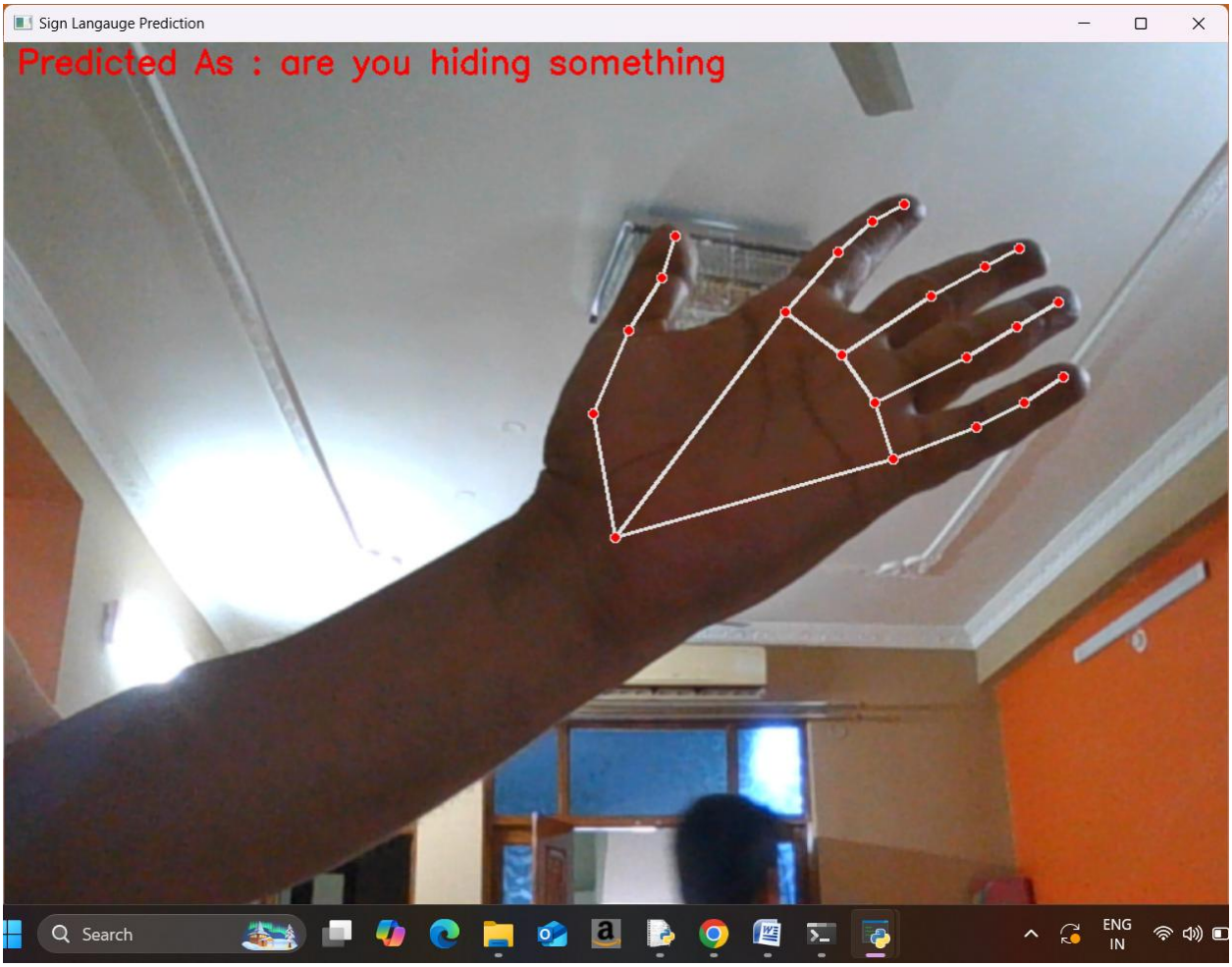
RESULTS

CHAPTER – 9

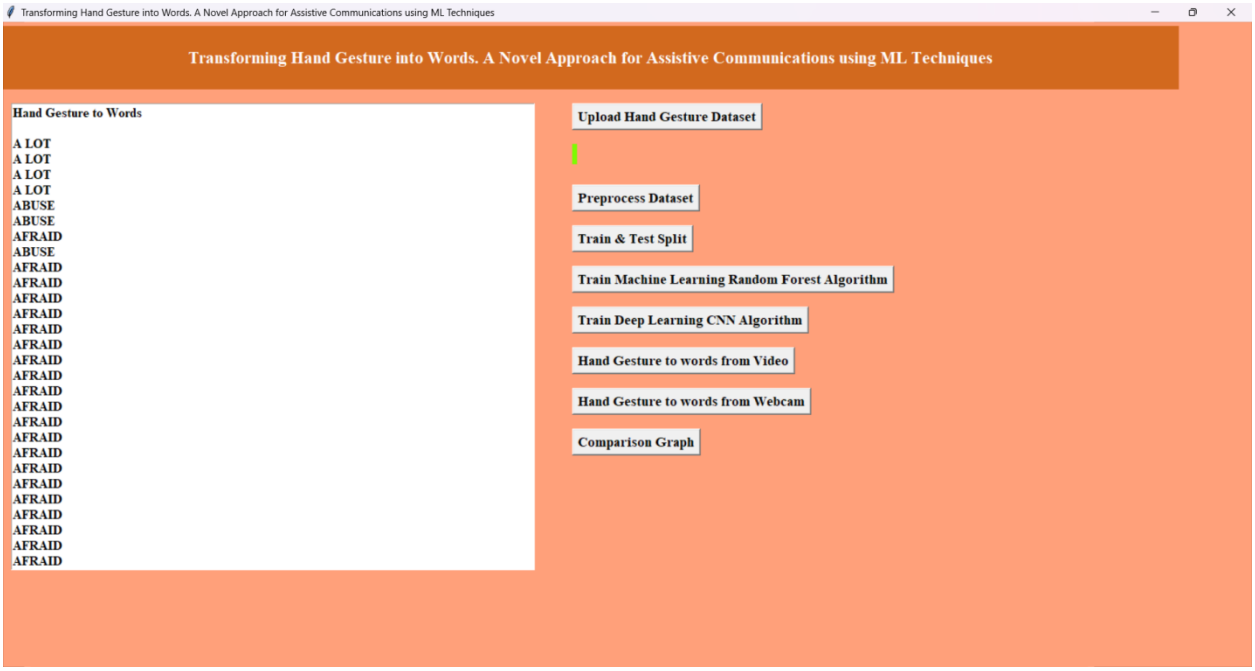
RESULTS



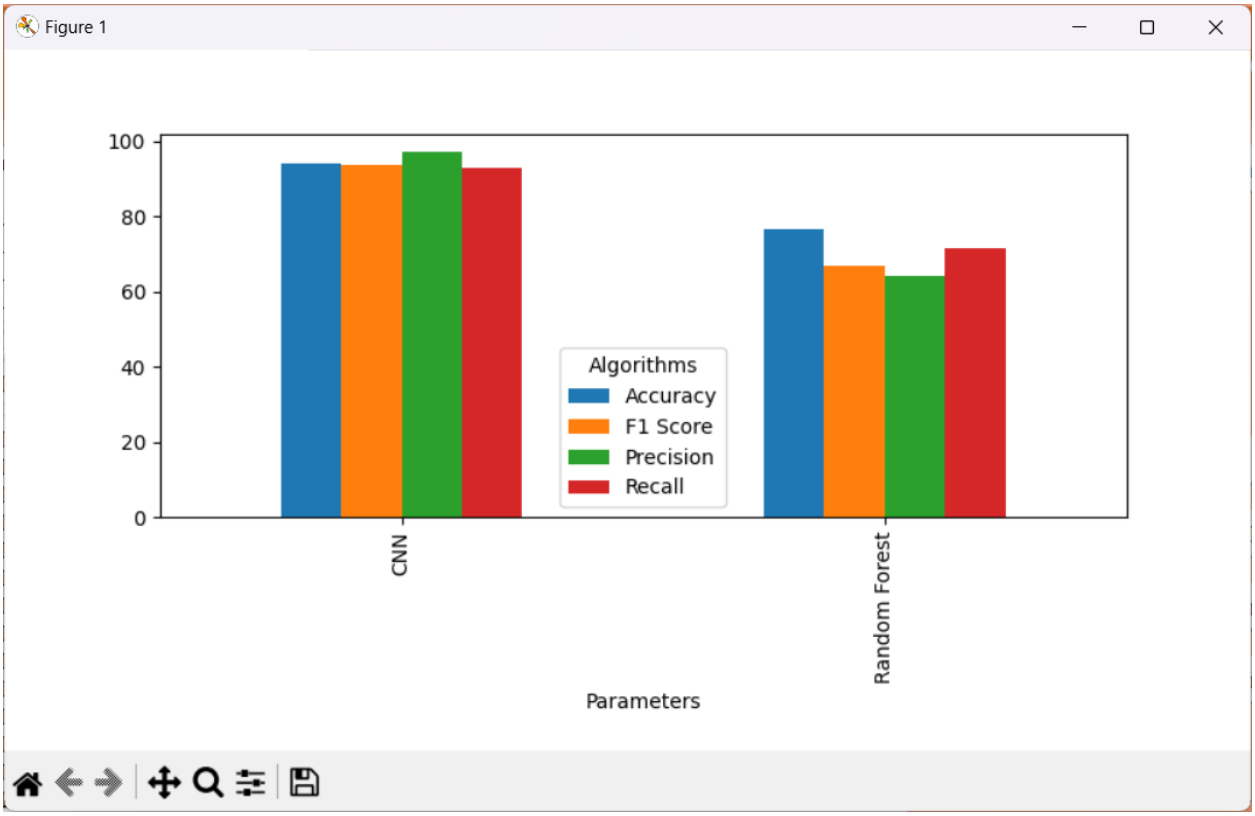
In above screen can see all converted hand gesture words and now click on 'Hand Gesture to words from Webcam' button to start webcam and generate words



In above screen from webcam also hand gestures are identified and converted into words which can see in below screen.



In above screen can see all converted words and now click on ‘Comparison Graph’ button to get below graph



In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different color bars and in both algorithms CNN got high performance.

SCREENSHORTS

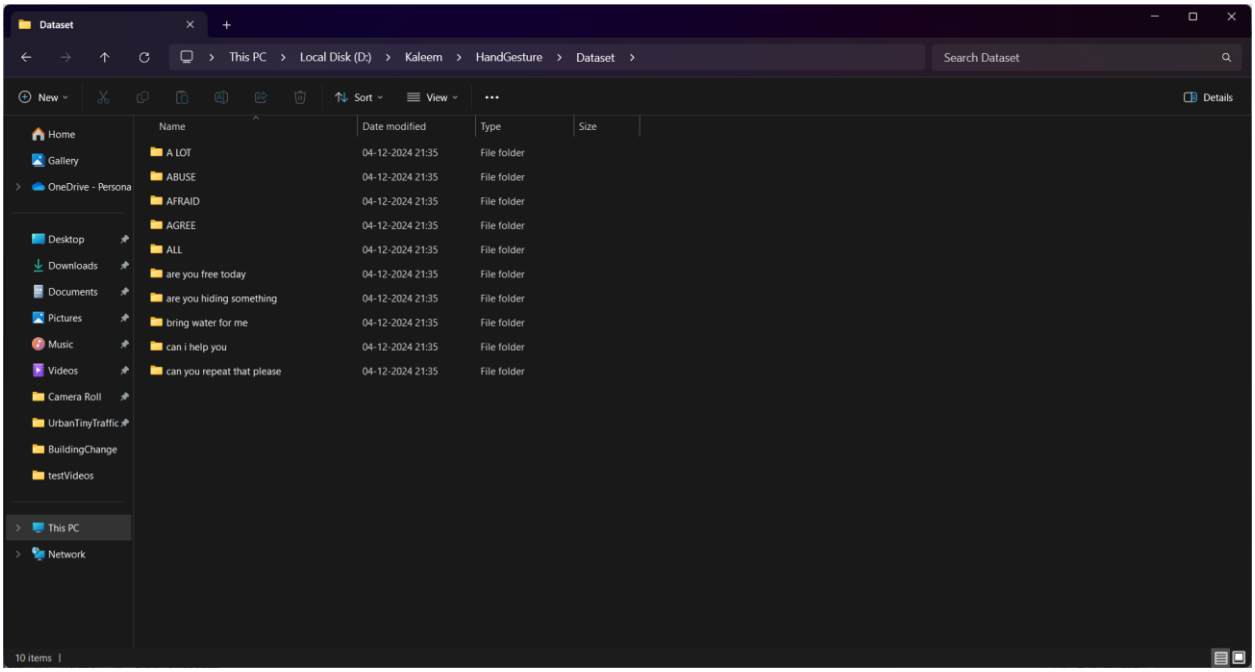
CHAPTER – 10

SCREENSHORTS

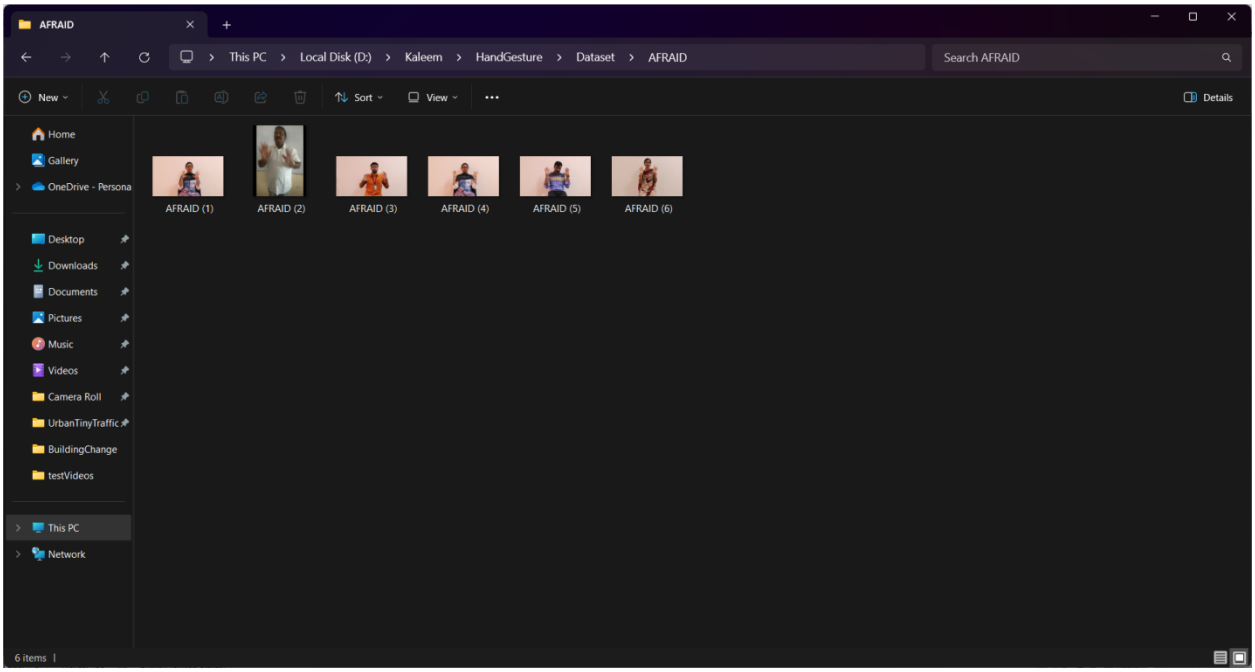
Transforming Hand Gesture into Words: A Novel Approach for Assistive Communications using ML Techniques

In propose project we are employing Machine & Deep Learning algorithms to convert human gesture into words. Each algorithm performance is evaluated in terms of accuracy, precision, recall and FSCORE. As Machine Learning we have utilized Random Forest algorithm and as deep learning we have utilized CNN algorithm. Among both algorithms CNN is getting highest accuracy.

To train above algorithms we have utilized Hand Gesture dataset which consists of 10 different Hand Signs which is showing below



In above screen each folder contains different Hand Signs and its names can be seen as folder name. Several Hand Signs are available but on internet we manage to get this many words hand signs and just go inside any folder to view sign images.



In above screen can see Hand Gesture signs for 'Afraid' word. To extract features from above signs we have utilize Media-pipe algorithm to identify hand signs and to obtain features.

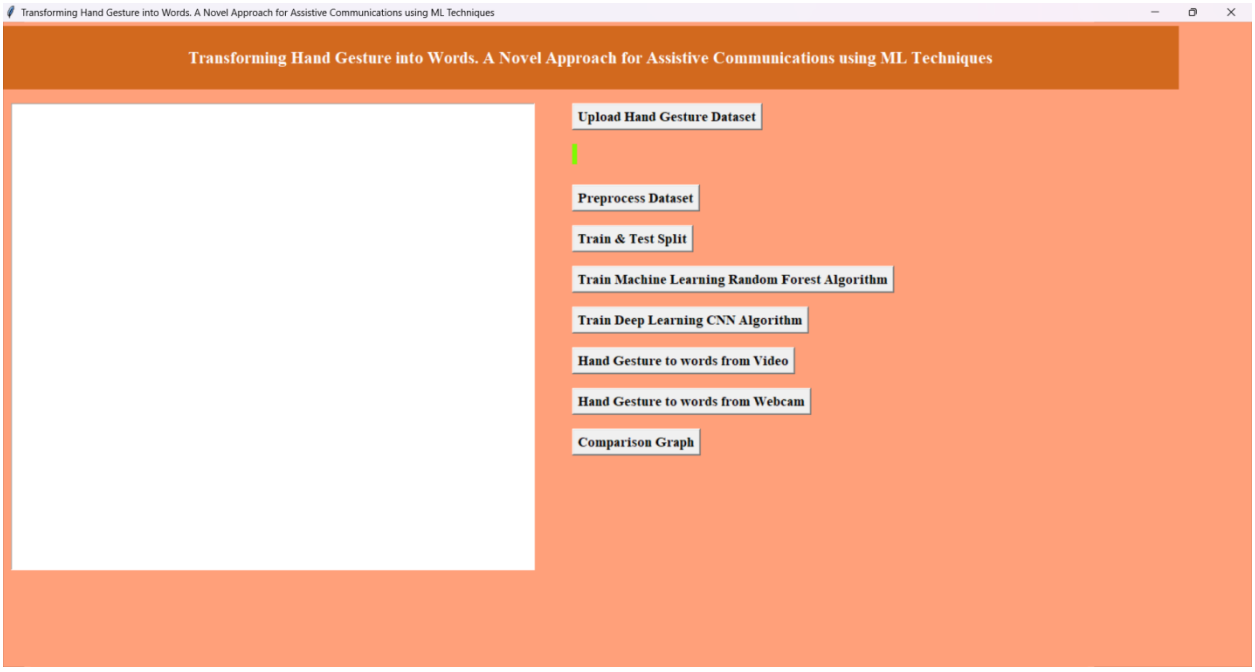
To implement this project we have designed following modules

1) Upload Hand Gesture Dataset: using this module user can upload 'Dataset' folder with different hand signs. This module will apply Media-Pipe algorithm to detect hand positions and then extract features
2) Preprocess Dataset: this module will apply processing techniques such as normalization and shuffling on extracted features
3) Train & Test Split: this module will split processed features into train and test where application using 80% dataset features for training and 20% for testing
4) Train Machine Learning Random Forest Algorithm: 80% training features will be input to Random Forest ML algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy
5) Train Deep Learning CNN Algorithm: 80% training features will be input to CNN algorithm to train a model and this model will be applied on 20% test data to calculate prediction accuracy

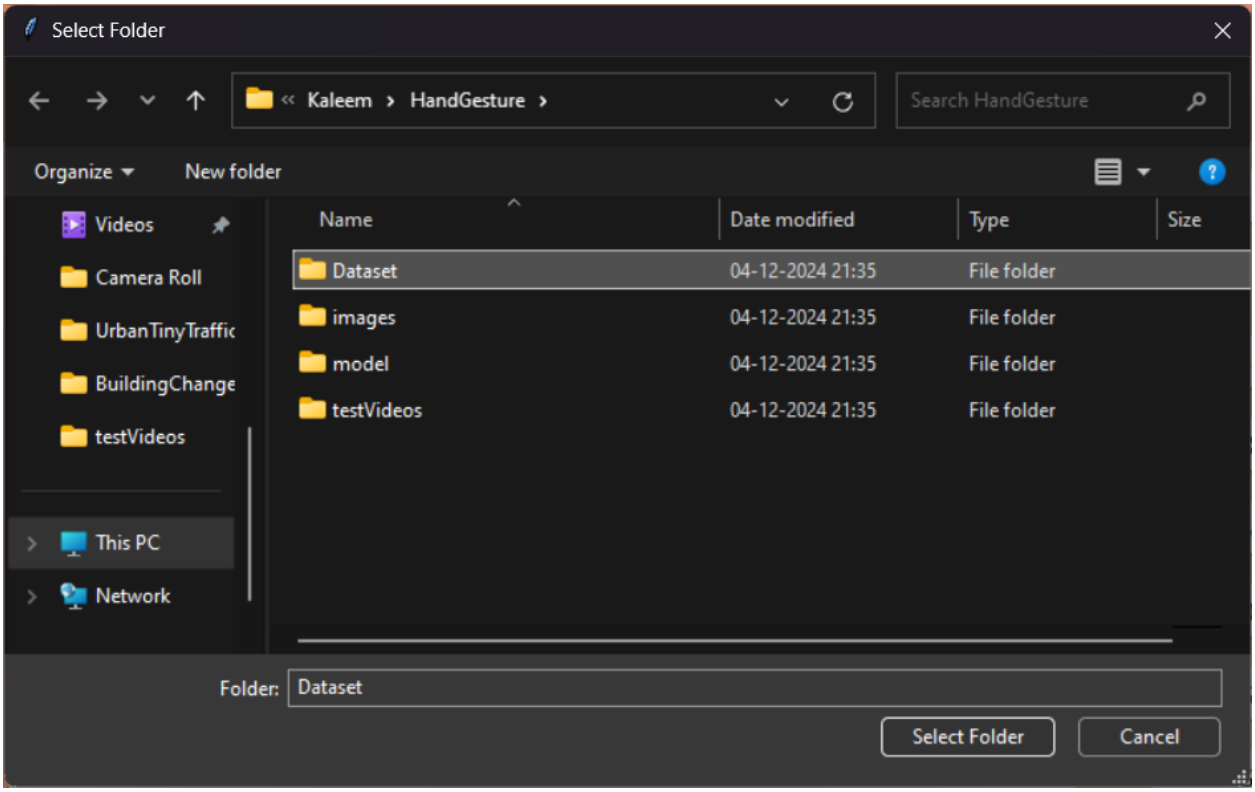
6) Hand Gesture to words from Video: using this module we can upload test video with different hand gestures and then CNN will predict type of gesture and converts into words
7) Hand Gesture to words from Webcam: using this module we can open webcam video where user can show different hand signs on webcam and then CNN will predict type of gesture and converts into words
8) Comparison Graph: this module will plot comparison graph between all algorithms.

SCREEN SHOTS

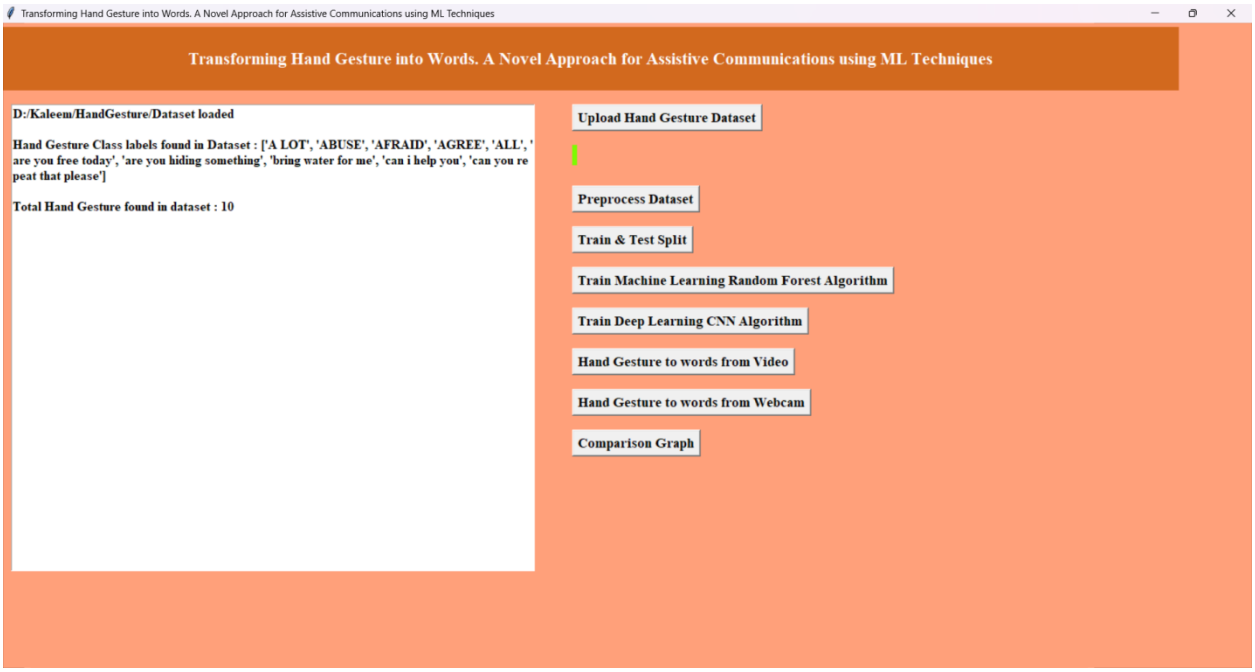
To run project double click on 'run.bat' file to get below screen



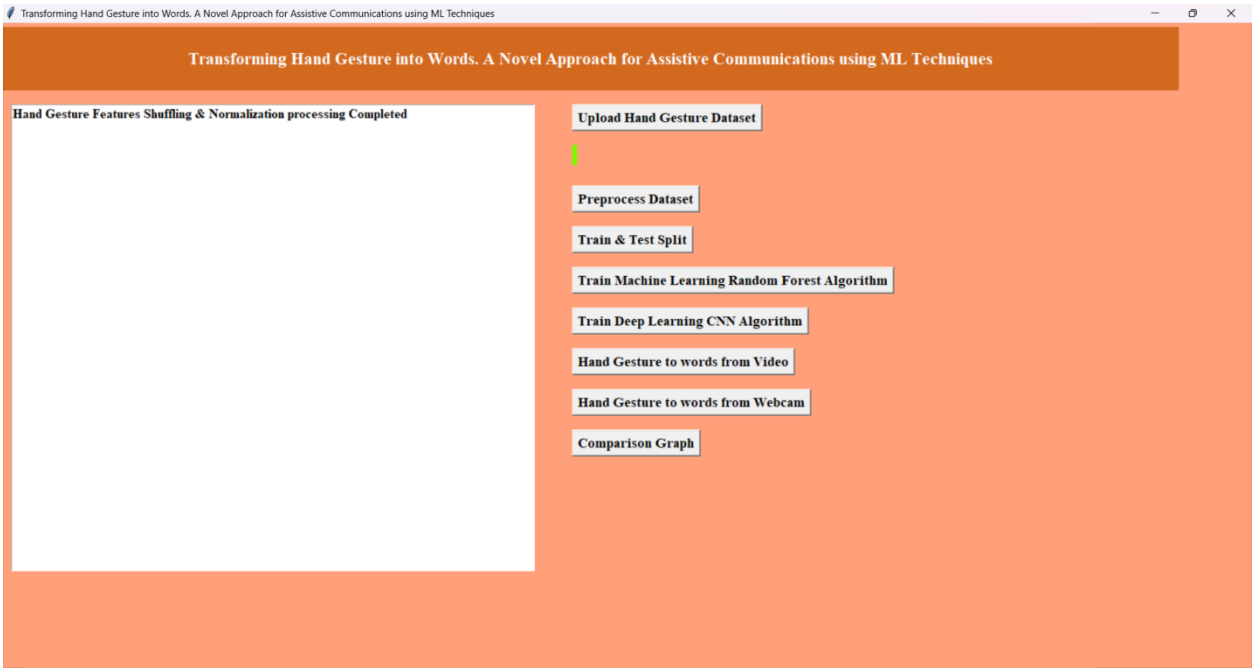
In above screen click on 'Upload Hand Gesture Dataset' button to upload dataset and get below page



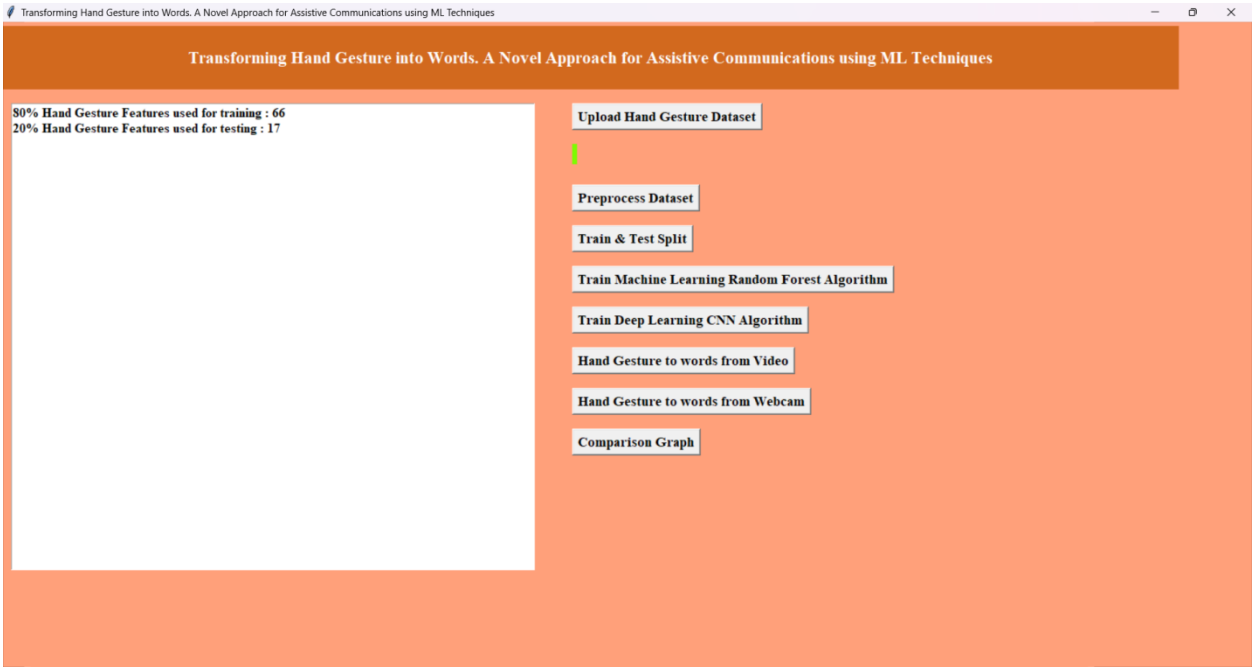
In above screen selecting and uploading entire 'Dataset' folder and then click on 'Select Folder' button to load dataset and get below page



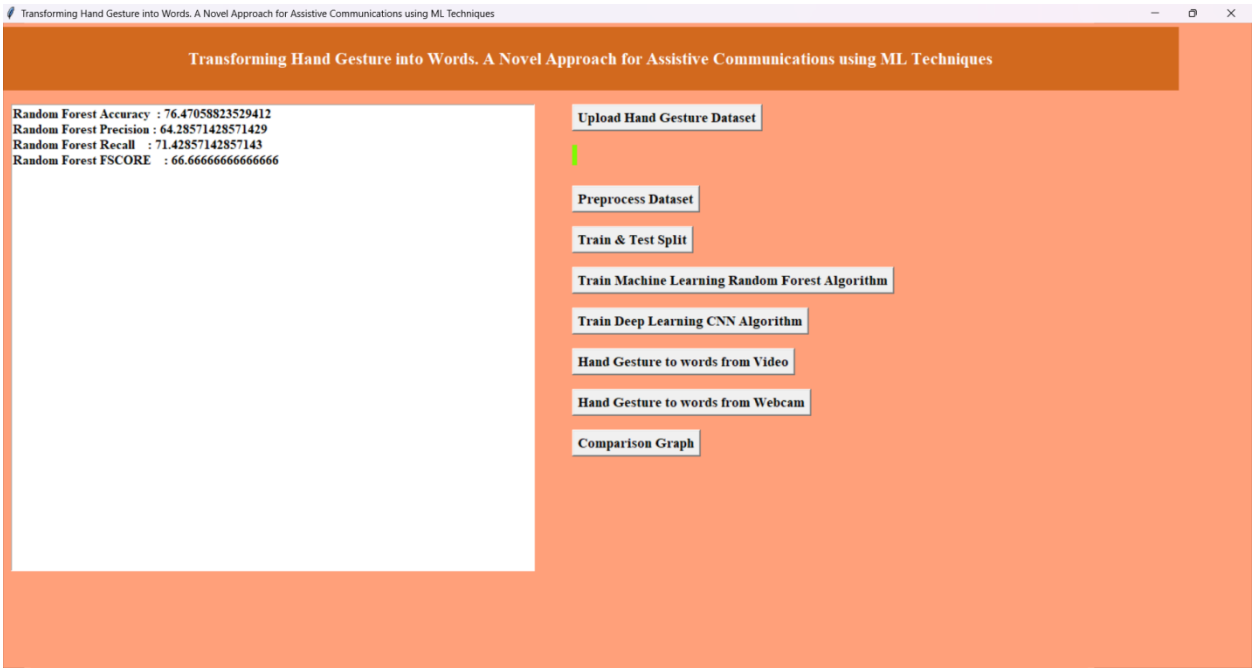
In above screen can see different hand gesture images loaded to application and can see number of loaded different hand gestures and now click on 'Pre-process Dataset' button to normalize features and get below page



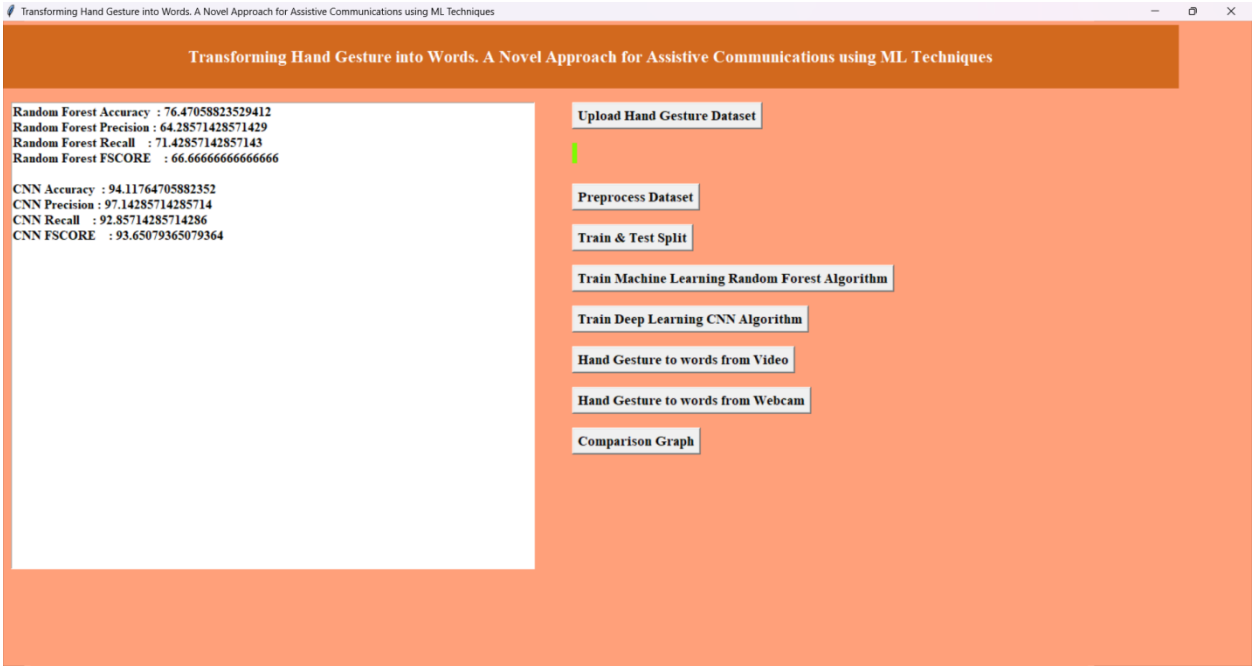
In above screen dataset features processing completed and now clicks on 'Train & Test Split' button to split dataset and to get below page



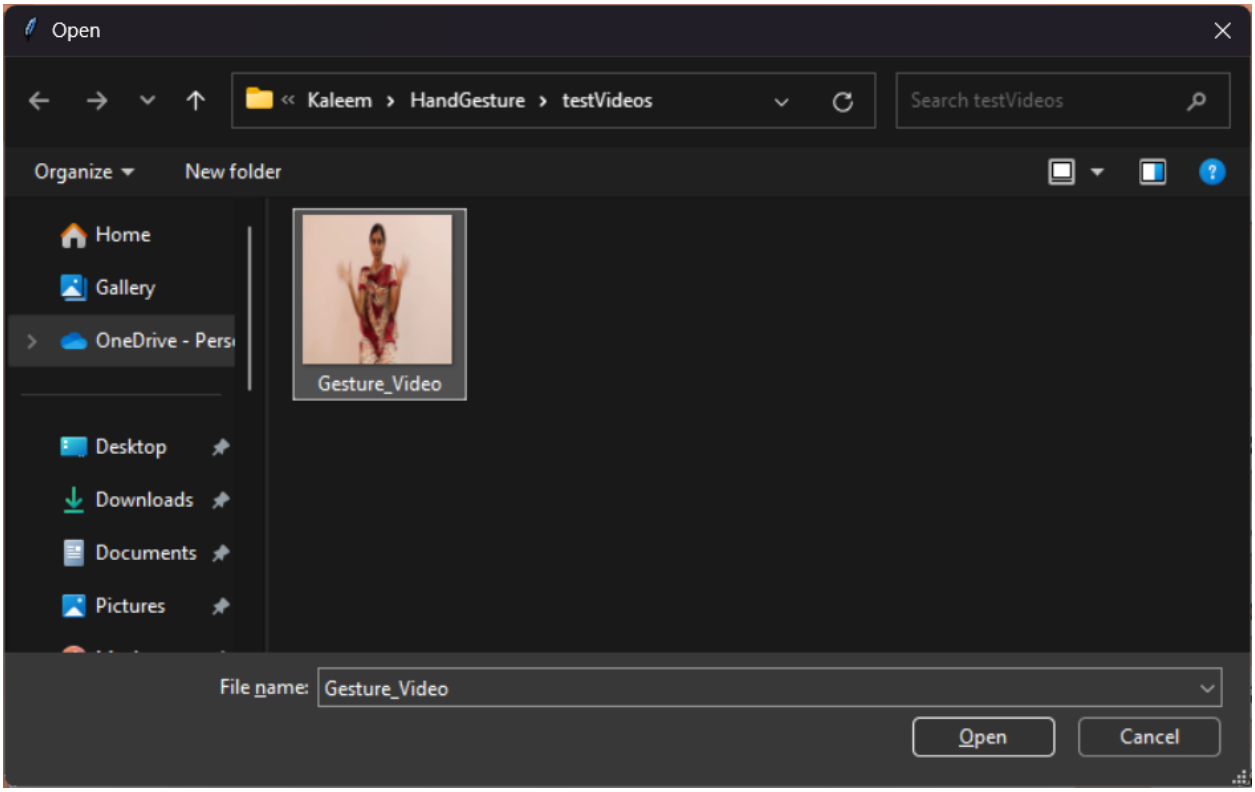
In above screen can see dataset train and test size and now click on 'Train Machine Learning Random Forest Algorithm' button to train Random Forest and get below page



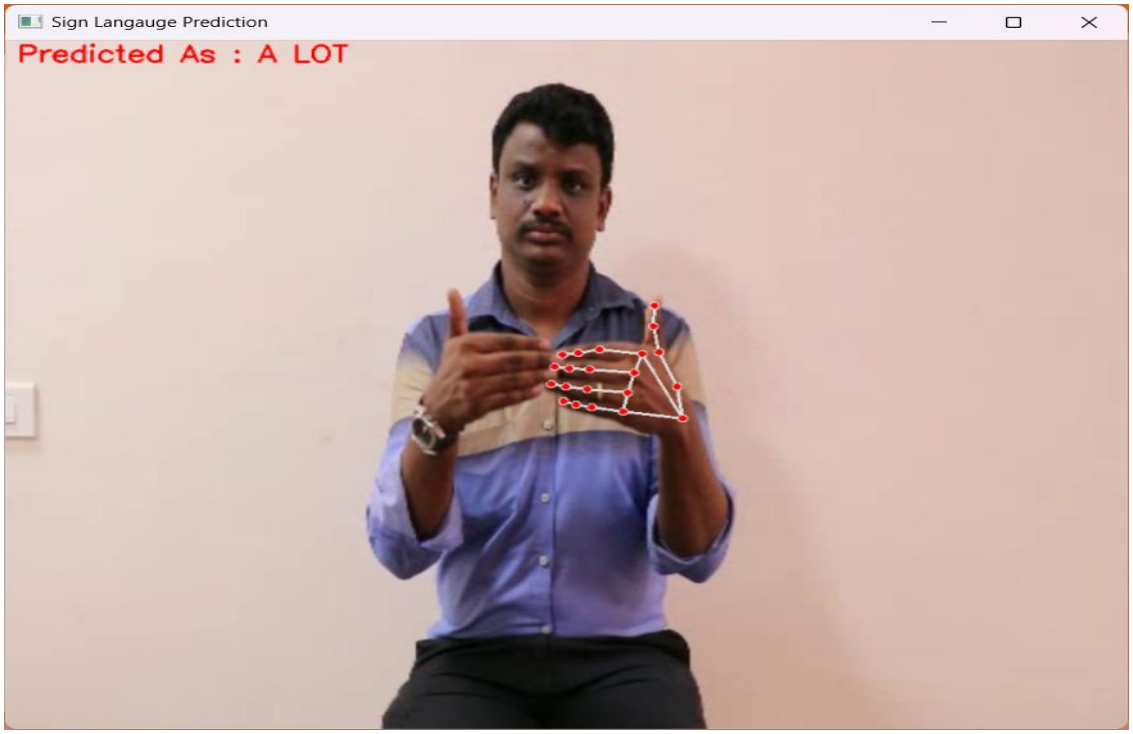
In above screen Random Forest got 76% accuracy and can see other metrics like precision, recall and FSCORE. Now click on 'Train Deep Learning CNN Algorithm' button to train CNN algorithm and get below page



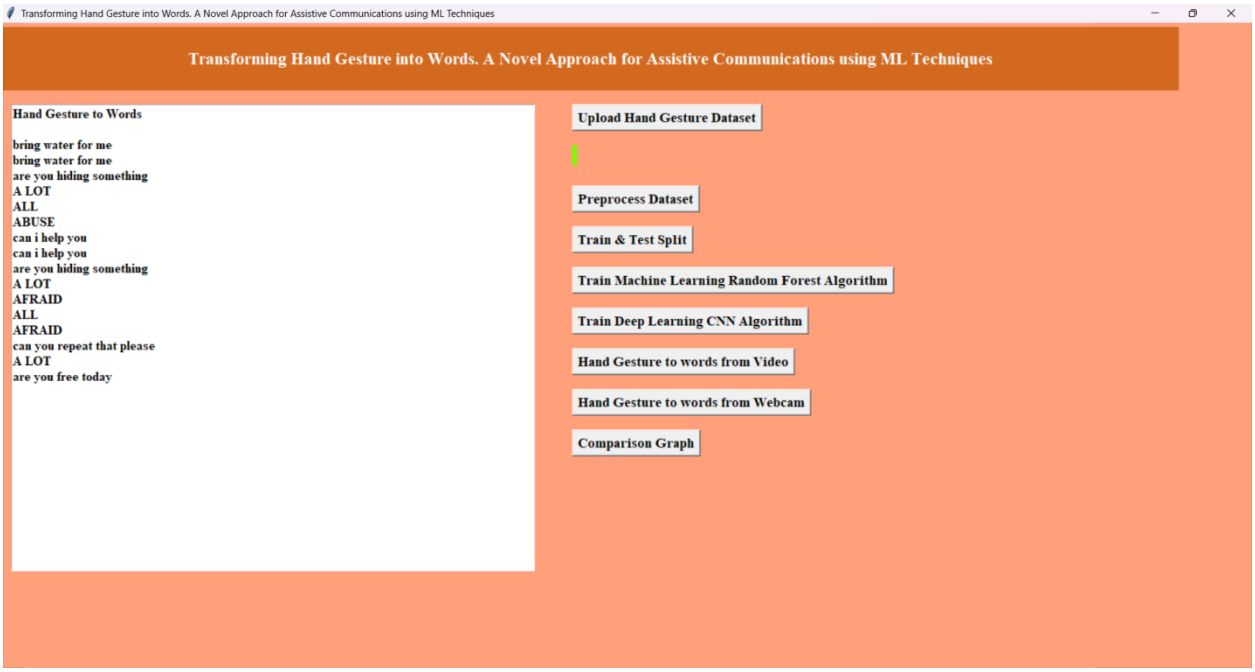
In above screen Deep learning CNN algorithm got 94% accuracy and can see other metrics also. Now click on 'Hand Gesture to words from Video' button to upload video and get below output



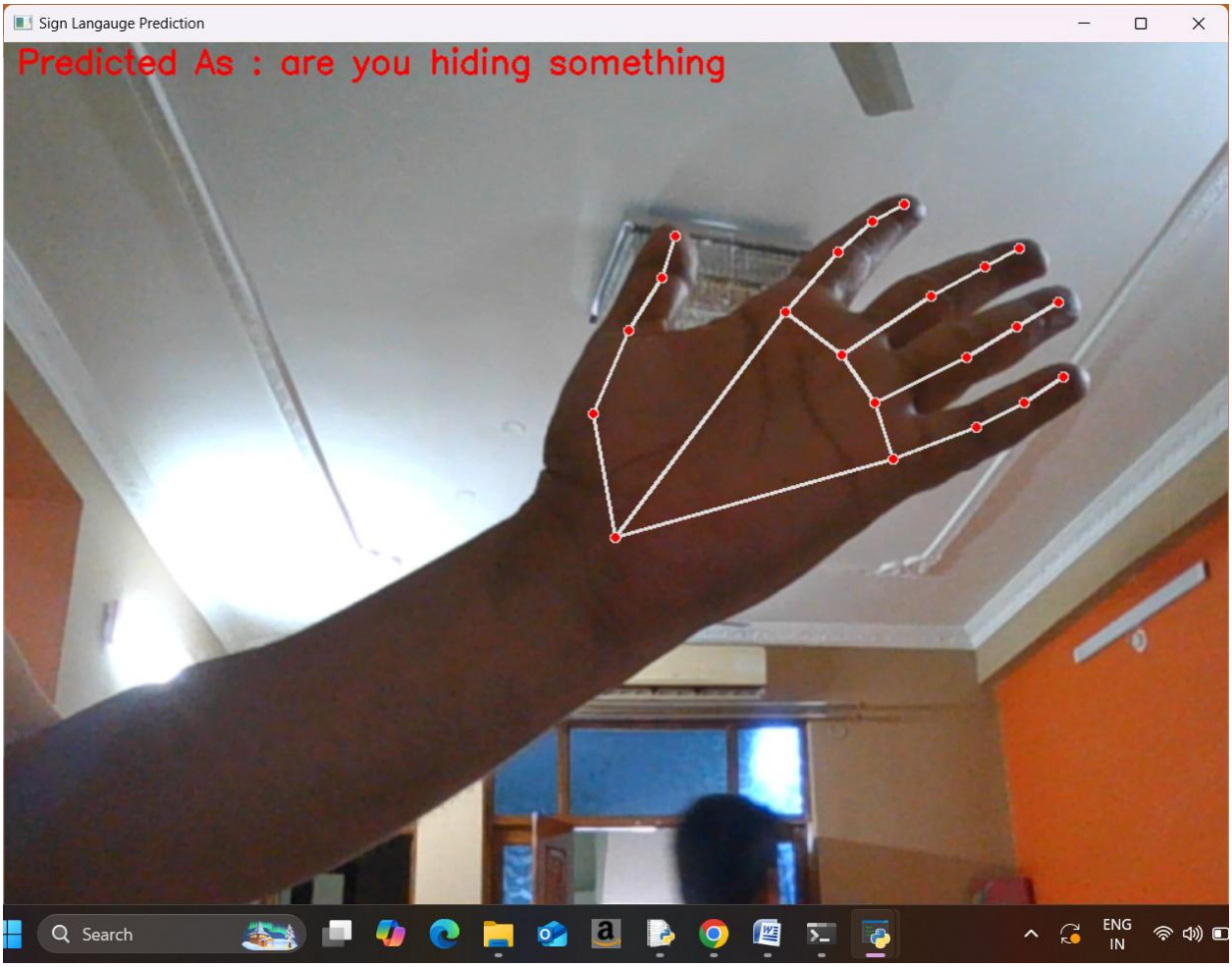
In above screen uploading video with different Hand Gesture and now click on ‘Open’ button to play video and then convert detected gesture into words



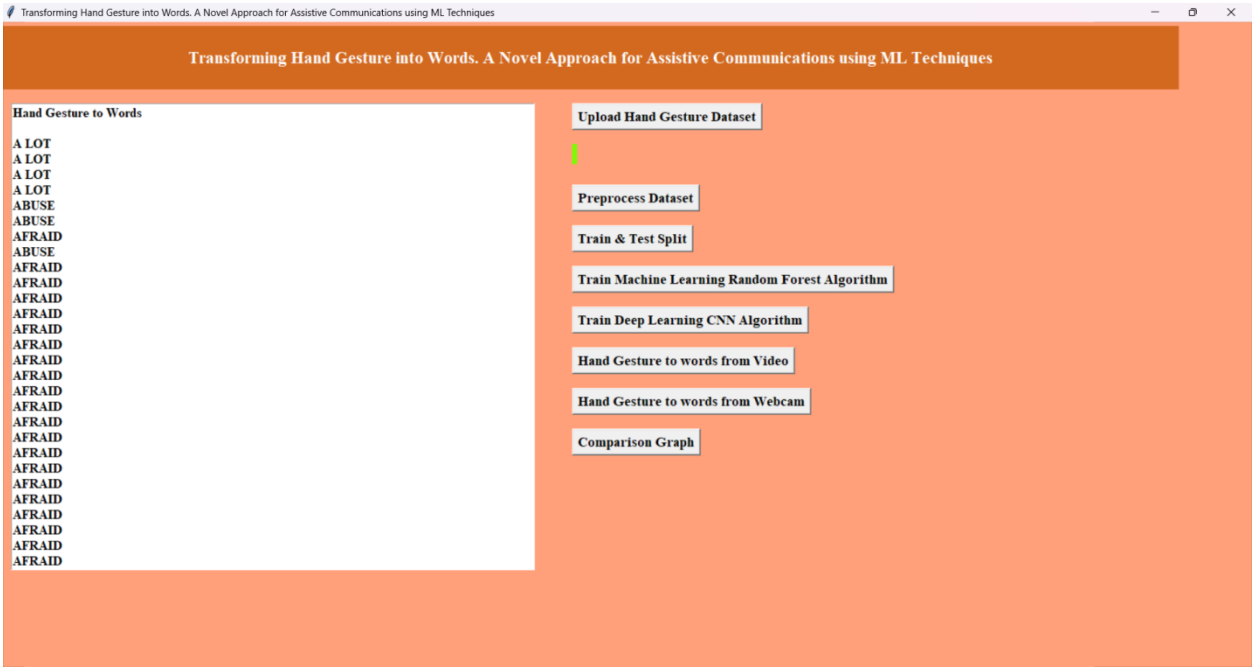
In above screen as video play then algorithm will identify hand gesture and then convert into words which are showing in red color text and after entire video playing completed then will get below generated ‘words’.



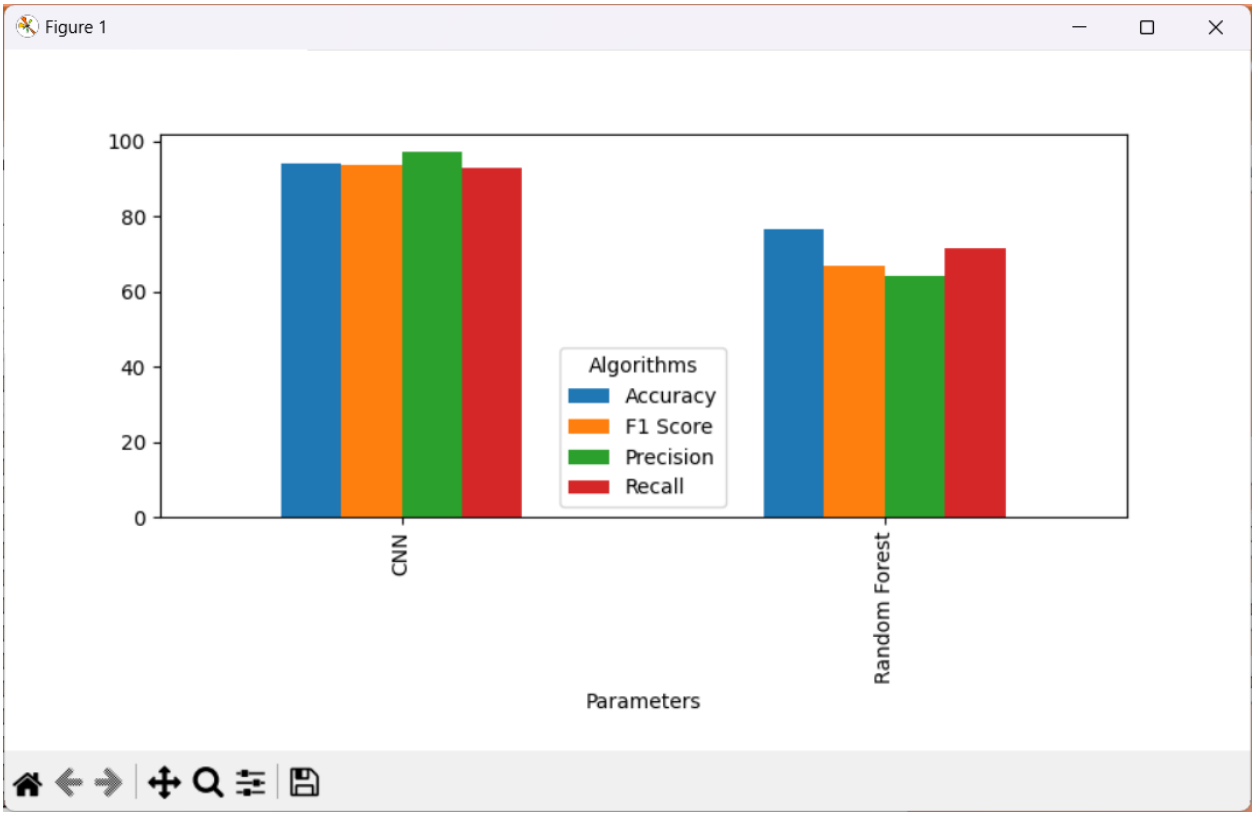
In above screen can see all converted hand gesture words and now click on 'Hand Gesture to words from Webcam' button to start webcam and generate words



In above screen from webcam also hand gestures are identified and converted into words which can see in below screen.



In above screen can see all converted words and now click on 'Comparison Graph' button to get below graph



In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different color bars and in both algorithms CNN got high performance.

SYSTEM TESTING

CHAPTER – 11

SYSTEM TESTING

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

TYPES OF TESTS

Unit testing

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

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by 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.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before

functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

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

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. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

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.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CONCLUSION

&

FUTURE SCOPE

CHAPTER - 12

CONCLUSION AND FUTURE SCOPE

In this project, a robust gesture recognition system has been successfully developed using Convolutional Neural Networks (CNN) integrated with OpenCV for real-time hand gesture detection and classification. The system effectively translates hand gestures into meaningful words or phrases, making it a valuable tool for assistive communication, especially for individuals with hearing and speech impairments. By leveraging computer vision techniques and deep learning models, the application can accurately recognize both static and dynamic gestures from video and webcam input. The model's performance was evaluated and visualized, showing promising results in terms of accuracy and reliability. The integration of pose estimation and CNNs contributed significantly to the precision of gesture recognition, enabling seamless human-computer interaction.

FUTURE SCOPE:

Although the current system demonstrates effective gesture recognition capabilities, there is scope for further enhancements. Future work can focus on expanding the gesture dataset to include a wider variety of sign languages and cultural variations. Additionally, integrating Natural Language Processing (NLP) techniques could enable full sentence construction and contextual understanding of gestures. Improving the user interface for mobile and embedded platforms can further enhance accessibility and portability.

Incorporating multi-modal inputs such as voice or facial expressions could lead to the development of a more comprehensive and intelligent assistive communication system. Finally, deploying the system on edge devices using lightweight CNN architectures like MobileNet can optimize real-time performance in low-resource environments.

REFERENCES

CHAPTER – 13

REFERENCES

1. **Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2015)** *Hand gesture recognition with 3D convolutional neural networks.* In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1-7.
   👉 [DOI: 10.1109/CVPRW.2015.7301347]
2. **Simonyan, K., & Zisserman, A. (2014)** *Two-stream convolutional networks for action recognition in videos.* In *Advances in neural information processing systems*, 27.
   👉 [https://arxiv.org/abs/1406.2199]
3. **Ohn-Bar, E., & Trivedi, M. M. (2014)** *Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations.* *IEEE Transactions on Intelligent Transportation Systems*, 15(6), 2368–2377.
   👉 [DOI: 10.1109/TITS.2014.2311657]
4. **Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2016)** *Deep sign: Hybrid CNN-HMM for continuous sign language recognition.* In *British Machine Vision Conference (BMVC)*.
   👉 [https://arxiv.org/abs/1604.06543]
5. **Mittal, A., & Yadav, A. (2019)** *Real-time hand gesture recognition using deep learning.* In *International Journal of Computer Sciences and Engineering*, 7(4), 536–541.
   👉 [DOI: 10.26438/ijcse/v7i4.536541]

6. **Ghorbel, M., Ghoualmi, Z., & Mahfoudhi, A. (2020)** *CNN-based hand gesture recognition using open-source frameworks.* In *Procedia Computer Science*, 176, 1956–1965. 👉 [DOI: 10.1016/j.procs.2020.09.231]
7. **Ahmed, M., Ali, H., & Khan, A. (2018)** *Sign language recognition using deep learning on custom dataset.* In *International Journal of Advanced Computer Science and Applications*, 9(9). 👉 [DOI: 10.14569/IJACSA.2018.090969]
8. **Tayal, S., & Sharma, K. (2021)** *Vision-based hand gesture recognition using CNNs.* *Procedia Computer Science*, 173, 388–395. 👉 [DOI: 10.1016/j.procs.2020.06.044]
9. **Zhang, Y., & Tian, Y. (2016)** *RGB-D camera-based daily living activity recognition with multi-feature fusion and deep learning.* *Multimedia Tools and Applications*, 75, 10395–10414. 👉 [DOI: 10.1007/s11042-015-3129-7]
10. **Li, Y., Wang, M., & Kang, Y. (2020)** *Real-time hand gesture recognition system using CNNs based on OpenCV and Python.* *International Journal of Computer Applications*, 177(36), 1–6. 👉 [DOI: 10.5120/ijca2020919750]