A Project On

Hand Gesture Recognition Using Automatic Feature Extraction and Deep Learning Algorithms with Memory

Project submitted in the partial fulfilment of the requirements of the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

ANNAVARAPU GUNA RAMA CHADRA RAO
(20U91A0505)

Under the esteemed guidance of

P L N MANOJ KUMAR, MTech
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SRI MITTAPALLI COLLEGE OF ENGINEERING
(Approved by AICTE & Affiliated to JNTUK Kakinada)
(An ISO 9001:2015 Certified institution and accredited by NAAC & NBA)
THUMMALAPALEM, NH-16, GUNTUR-522 233, A.P.

(2020-2024)

SRI MITTAPALLI COLLEGE OF ENGINEERING

(Approved by AICTE & Affiliated to JNTUK Kakinada)

(An ISO 9001:2015 Certified institution and accredited by NAAC & NBA)

THUMMALAPALEM, NH-16, GUNTUR-522 233, A.P.

CERTIFICATE

This is to certify that a main project report entitled "**Hand Gesture Recognition Using Automatic Feature Extraction and Deep Learning Algorithms with Memory**" being submitted By ANNAVARAPU GUNA RAMA CHANDRA RAO (20U91A0505),for the partial fulfilment for the award of degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** of **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA** during the year 2023-2024.

Project Guide

P L N MANOJ KUMAR MTech

Associate Professor

Head of the Department

Dr. S. GOPI KRISHNA M. Tech, Ph.D.

HOD of CSE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I express my sincere thanks to our beloved Chairman Sri. M. V. KOTESWARA RAO and the Secretary Sri. M. B. SATYANARAYANA for providing their support and simulating environment for developing the project.

I express my deep sense of reverence and profound graduate Dr. S. GOPI KRISHNA M. Tech, Ph.D., Principal for providing us the great support for carrying out this project.

I extend my sincere thanks to Dr. S. GOPI KRISHNA M. Tech, Ph.D., Head of the Department of C.S.E for his co-operation and guidance to make this project successful.

I would like to express my indebted gratitude to our guide P L N MANOJ KUMAR MTech Asst.Professor who has guided a lot and encouraged us in every step of the project work. His moral support and guidance throughout the project helped us to a great extent.

I also place my floral gratitude to all teaching and non-teaching staff for their constant support and advice throughout the project. Last but not least we thank our friends who directly or indirectly helped us in the successful completion of this project work.

ANNAVARAPU GUNA RAMA CHANDRA RAO.

(20U91A0505)

PLACE:

DATE:

DECLARATION

I A. GUNA RAMA CHANDRA RAO, declare that the contents of this project, in full or part, have not been submitted to any other university or institution for the award of any degree or diploma.

I also declare that we have adhered to all principles of academic honest and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

I understand that any violation of the above will cause for disciplinary action by the institution and can also evoke penal action for the sources which have not been properly cited or from whom proper permission has not been taken when needed.

ANNAVARAPU GUNA RAMA CHANDRA RAO.

(20U91A0505)

PLACE:

DATE:

TABLE OF CONTENTS

Abstract

Hand gesture Recognition (HGR) targets on interpreting the sign language into text or speech, so as to facilitate the communication between deaf-mute people and ordinary people. This task has broad social impact, but is still very challenging due to the complexity and large variations in hand actions. Existing methods for HGR use hand-crafted features to describe sign language motion and build classification models based on those features. However, it is difficult to design reliable features to adapt to the large variations of hand gestures. To approach this problem, we propose a novel convolutional neural network (CNN) which extracts discriminative spatial-temporal features from raw video stream automatically without any prior knowledge, avoiding designing features. To boost the performance, multi-channels of video streams, including color information, depth clue, and body joint positions, are used as input to the CNN in order to integrate color, depth and trajectory information. We validate the proposed model on a real dataset collected with Microsoft Kinect and demonstrate its effectiveness over the traditional approaches based on hand-crafted features.

Introduction

Sign language, as one of the most widely used communication means for hearing-impaired people, is expressed by variations of hand-shapes, body movement, and even facial expression. Since it is difficult to collaboratively exploit the information from hand-shapes and body movement trajectory, sign language recognition is still a very challenging task. This paper proposes an effective recognition model to translate sign language into text or speech in order to help the hearing impaired communicate with normal people through sign language.

Technically speaking, the main challenge of sign language recognition lies in developing descriptors to express hand-shapes and motion trajectory. In particular, hand-shape description involves tracking hand regions in video stream, segmenting hand-shape images from complex background in each frame and gestures recognition problems. Motion trajectory is also related to tracking of the key points and curve matching. Although lots of research works have been conducted on these two issues for now, it is still hard to obtain satisfying result for SLR due to the variation and occlusion of hands and body joints. Besides, it is a nontrivial issue to integrate the hand-shape features and trajectory features together. To address these difficulties, we develop a CNNs to naturally integrate hand-shapes, trajectory of action and facial expression. Instead of using commonly used color images as input to networks like [1, 2], we take color images, depth images and body skeleton images simultaneously as input which are all provided by Microsoft Kinect.

Kinect is a motion sensor which can provide color stream and depth stream. With the public Windows SDK, the body joint locations can be obtained in real-time as shown in Fig.1. Therefore, we choose Kinect as capture device to record sign words dataset. The change of color and depth in pixel level are useful information to discriminate different sign actions. And the variation of body joints in time dimension can depict the trajectory of sign actions. Using multiple types of visual sources as input leads CNNs paying attention to the change not only in color, but also in depth and trajectory. It is worth mentioning that we can avoid the difficulty of tracking hands, segmenting hands from background and designing descriptors for hands because CNNs have the capability to learn features automatically from raw data without any prior knowledge .

CNNs have been applied in video stream classification recently years. A potential concern of CNNs is time consuming. It costs several weeks or months to train a CNNs with million-scale in million videos. Fortunately, it is still possible to achieve real-time efficiency, with the help of CUDA for parallel processing. We propose to apply CNNs to extract spatial and temporal features

from video stream for Sign Language Recognition (SLR). Existing methods for SLR use hand-crafted features to describe sign language motion and build classification model based on these features. In contrast, CNNs can capture motion information from raw video data automatically, avoiding designing features. We develop a CNNs taking multiple types of data as input. This architecture integrates color, depth and trajectory information by performing convolution and subsampling on adjacent video frames. Experimental results demonstrate that 3D CNNs can significantly outperform Gaussian mixture model with Hidden Markov model (GMM-HMM) baselines on some sign words recorded by ourselves.

Motivation

For interaction between normal people and D&M people a language barrier is created as sign language structure which is different from normal text. So they depend on vision based communication for interaction.

If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people. So research has been made for a vision based interface system where D&M people can enjoy communication without really knowing each other's language.

The aim is to develop a user-friendly, human computer interface (HCI) where the computer understands the human sign language. There are various sign languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world.

Literature survey

In the recent years there has been tremendous research done on the hand gesture recognition.

With the help of literature survey done we realized the basic steps in hand gesture recognition are: -

●Data acquisition

●Data preprocessing

●Feature extraction

●Gesture classification

Data acquisition:

The different approaches to acquire data about the hand gesture can be done in the following ways:

1. Use of sensory devices

It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

2. Vision based approach

In vision-based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware.

The main challenge of vision-based hand detection is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin-colour possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

Data preprocessing and Feature extraction for vision-based approach:

●In the approach for hand detection combines threshold-based color detection with background subtraction. We can use Adaboost face detector to differentiate between faces and hands as both involve similar skin-color.

●We can also extract necessary image which is to be trained by applying a filter called Gaussian blur. The filter can be easily applied using open computer vision also known as OpenCV and is described in.

●For extracting necessary image which is to be trained we can use instrumented gloves as mentioned in. This helps reduce computation time for preprocessing and can give us more concise and accurate data compared to applying filters on data received from video extraction.

●We tried doing the hand segmentation of an image using color segmentation techniques but as mentioned in the research paper skin color and tone is highly dependent on the lighting conditions due to which output we got for the segmentation we tried to do were no so great. Moreover we have a huge number of symbols to be trained for our project many of which look similar to each other like the gesture for symbol 'V' and digit '2', hence we decided that in order to produce better accuracies for our large number of symbols, rather than segmenting the hand out of a random background we keep

background of hand a stable single color so that we don't need to segment it on the basis of skin color . This would help us to get better results.

Gesture classification:

●In Hidden Markov Models (HMM) is used for the classification of the gestures .This model deals with dynamic aspects of gestures.Gestures are extracted from a sequence of video images by tracking the skin-colour blobs corresponding to the hand into a body– face space centered on the face of the user. The goal is to recognize two classes of gestures: deictic and symbolic.The image is filtered using a fast look–up indexing table. After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x,y) and the colourimetry (Y,U,V) of the skin colour pixels in order to determine homogeneous areas.

●In Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation. Thus, unlike many other

recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video, with a static background. The first step is to segment and label the objects of interest and to extract geometric invariants from them. Next step is the classification of gestures by using a K nearest neighbour algorithm aided with distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes‟ classifier.

●According to paper on “Human Hand Gesture Recognition Using a Convolution Neural Network” by Hsien-I Lin , Ming-Hsiang Hsu, and

Wei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct a skin model to extract the hand out of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to center the image about it. They input this image to a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using their model they produce an accuracy of around 95% for those 7 gestures.

System Analysis

5.1. Existing System:

The existing system for hand gesture recognition using deep learning may consist of a basic Convolutional Neural Network (CNN) architecture trained on a limited dataset of hand gesture images.

Faults:

Limited Dataset: The existing system may suffer from a lack of diverse and comprehensive dataset, leading to poor generalization and limited recognition capabilities.

Overfitting: Due to the limited dataset, the model may overfit to the training data, resulting in poor performance on unseen data.

Lack of Real-Time Recognition: The existing system may not provide real-time recognition capabilities, limiting its applicability in interactive systems.

Limited Gesture Classes: The system may only recognize a small number of predefined hand gestures, restricting its usability in scenarios requiring recognition of a wide range of gestures.

5.2. Proposed System:

The proposed system aims to address the faults of the existing system by incorporating the following improvements:

Large and Diverse Dataset: Utilize a large and diverse dataset of hand gesture images covering a wide range of gestures, variations in lighting, backgrounds, and hand orientations. This will help improve the model's ability to generalize to unseen data.

Data Augmentation: Apply data augmentation techniques such as rotation, translation, scaling, and flipping to artificially increase the size of the dataset and improve model robustness.

Advanced CNN Architectures: Implement state-of-the-art CNN architectures such as ResNet, DenseNet, or EfficientNet, pre-trained on large image datasets like ImageNet. Fine-tune these architectures on the hand gesture dataset to leverage transfer learning and improve performance.

Real-Time Recognition: Develop a real-time hand gesture recognition system capable of processing video input from a webcam or other camera sources. This involves optimizing the model and inference pipeline for low-latency processing.

Gesture Localization: Incorporate object detection techniques to localize hand regions within video frames before performing gesture recognition. This helps improve accuracy and robustness by focusing on relevant regions of interest.

Continuous Learning: Implement techniques for continuous learning to adapt the model to new gestures or variations in hand poses over time. This could involve online learning strategies or incremental training on new data.

User Interface Enhancement: Enhance the user interface with intuitive controls, feedback mechanisms, and visualization tools to provide a seamless and interactive gesture recognition experience.

System Requirements

6.1. Hardware Requirements

- **Processor** - Pentium–III
- **Speed** – 2.4GHz
- **RAM** - 512 MB(min)
- **Hard Disk** - 20 GB
- **Floppy Drive** - 1.44MB
- **Key Board** - Standard Keyboard
- **Monitor** – 15 VGA Color

6.2. Software Requirements

- **Python** - version– 3.8 64bit
- **Integrated Development Environment (IDE)** -Visual Studio Code
- **Deep Learning Framework** – TensorFlow, Keras
- **Computer Vision Libraries** – OpenCV (Open-Source Computer Vision Library)
- **Memory-Augmented Neural Networks (MANN) :**

  Memory-Augmented Networks (MANN) libraries

  Memory Networks in TensorFlow or PyTorch
- **Data Preprocessing Libraries –** NumPy, Pandas
- **Gesture Dataset**

6.3. System Study Feasibility Study

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

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY:

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

TECHNICAL FEASIBILITY:

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

SOCIAL FEASIBILITY:

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

6.4. Input and Output Design:

Input design:

considering the requirements, procedures to collect the necessary input data in most efficiently designed. The input design has been done keeping in view that, the interaction of the user with the system being the most effective and simplified way.

Also the measures are taken for the following

- Controlling the amount of input
- Avoid unauthorized access to the application.

- Eliminating extra steps
- Keeping the process simple
- At this stage the input forms and screens are designed.

Output design:

All the screens of the system are designed with a view to provide the user with easy operations in simpler and efficient way, minimum key strokes possible. Instructions and important information is emphasized on the screen. Almost every screen is provided with no error and important messages and option selection facilitates. Emphasis is given for speedy processing and speedy transaction between the screens. Each screen assigned to make it as much user friendly as possible by using interactive procedures. So to say user can operate the system without much help from the operating manual.

UML DIAGRAMS

Detailed Design:

UML is an acronym that stands for Unified Modelling Language. Simply put, UML is a modern approach to modelling and documenting software. In fact, it's one of the most popular business process modelling techniques. It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

GOALS:

The Primary goals in the design of the UML are as follows:

1.Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.

2.Provide extendibility and specialization mechanisms to extend the core concepts.

3.Be independent of particular programming languages and development process.

4.Provide a formal basis for understanding the modelling language.

5.Encourage the growth of OO tools market.

6.Support higher level development concepts such as collaborations, frameworks, patterns and components.

7.Integrate best practices.

7.1. USE CASEDIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

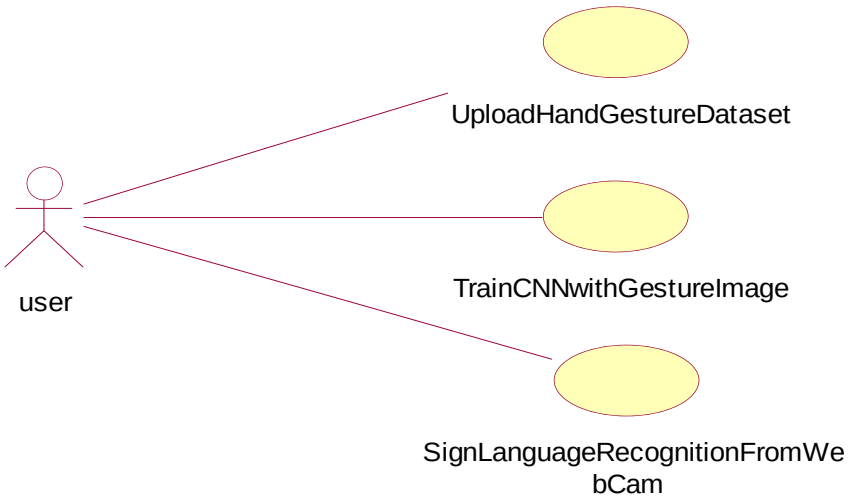


Fig 7.1: Use-Case diagram

7.2. CLASS:

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

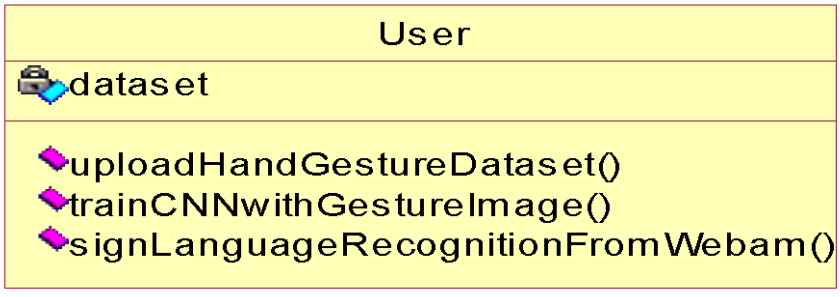


Fig 7.2: Class diagram

7.3. SEQUENCE:

In software engineering, a **sequence diagram** or **system sequence diagram** (**SSD**) shows process interactions arranged in a time sequence. The diagram depicts the processes and objects involved and the sequence of messages exchanged as needed to carry out the functionality. Sequence diagrams are typically associated with use case realizations in the 4+1 architectural view model of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios.**

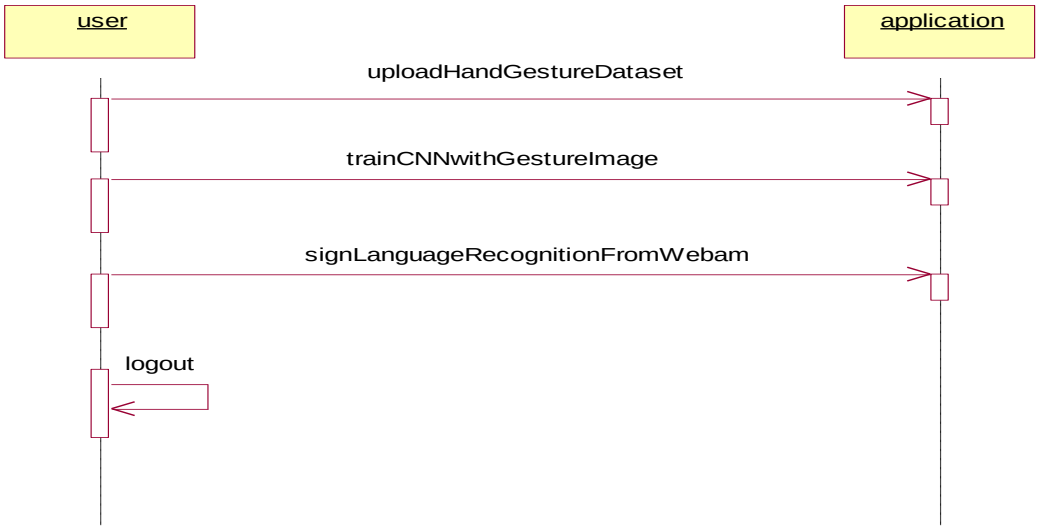


Fig 7.3: Sequence Diagram

7.4. COLLABORATION:

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behaviour of a system.

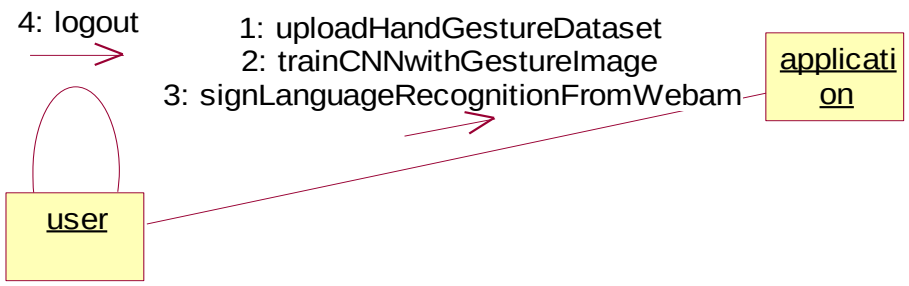


Fig 7.4: Collaboration Diagram

Technology Description

8.1. 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.

6. Simple and Easy

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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

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 is easy 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 Venners1, 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."

8.2. 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

8.2.1. NumPy

NumPy, which stands for Numerical Python, is a fundamental package for numerical computing in Python. It provides support for multidimensional arrays (ndarrays), along with a collection of functions and tools for working with these arrays efficiently. Here's a brief overview of NumPy's key features-

Multidimensional Arrays (ndarrays):

NumPy's ndarray is a powerful data structure that represents arrays of any dimension. ndarrays offer efficient storage and manipulation of large datasets, making them ideal for numerical computations.

Array Creation and Manipulation:

NumPy provides various functions for creating arrays, such as numpy.array(), numpy.zeros(), numpy.ones(), numpy.arange(), numpy.linspace(), etc.

It offers extensive capabilities for manipulating arrays, including indexing, slicing, reshaping, concatenating, and splitting.

Element-Wise Operations:

NumPy supports element-wise operations on arrays, enabling efficient computation without the need for explicit loops.

Arithmetic operations, mathematical functions, logical operations, and bitwise operations can be applied element-wise to arrays.

Broadcasting:

Broadcasting is a powerful feature in NumPy that allows operations between arrays of different shapes.

NumPy automatically broadcasts arrays to perform element-wise operations when their shapes are compatible.

Linear Algebra Operations:

NumPy provides a comprehensive suite of linear algebra functions for performing operations such as matrix multiplication (numpy.dot()), matrix inversion (numpy.linalg.inv()), eigenvalue decomposition (numpy.linalg.eig()), and more.

Random Number Generation:

NumPy's numpy.random module offers functions for generating random numbers and random arrays with various distributions.

These functions are useful for tasks such as random sampling, permutation, and generating synthetic data for simulations.

Array Operations and Manipulations:

NumPy includes functions for array manipulation, including sorting (numpy.sort()), searching (numpy.where()), unique elements (numpy.unique()), stacking and splitting arrays, and more.

Performance and Efficiency:

NumPy is implemented in C and Fortran, making it highly efficient and fast for numerical computations.

It leverages optimized algorithms and memory layout for improved performance, especially when working with large datasets.

8.3. Deep Learning Frameworks

Deep learning is a subset of machine learning that focuses on training artificial neural networks with multiple layers to learn intricate patterns and representations from complex data. It aims to automatically discover hierarchical features through successive layers of abstraction, mimicking the structure and function of the human brain. Deep learning has revolutionized various fields such as computer vision, natural language processing, speech recognition, and reinforcement learning, achieving remarkable performance in tasks like image classification, object detection, language translation, and game playing. Key techniques in deep learning include Convolutional Neural Networks (CNNs) for processing spatial data, Recurrent Neural Networks (RNNs) for handling sequential data, and Transformer models for capturing long-range dependencies in sequences. Deep learning has propelled advancements in artificial intelligence, enabling machines to perceive, understand, and generate data in ways previously thought impossible, and continues to drive innovation across diverse domains.

8.3.1 Keras:

- Keras is a high-level neural networks API written in Python, capable of running on top of TensorFlow, Microsoft Cognitive Toolkit (CNTK), or Theano.
- It provides a user-friendly interface for building, training, and deploying deep learning models with minimal code.
- Keras allows easy experimentation with different architectures and hyperparameters, making it ideal for both beginners and experienced researchers.

8.3.2 TensorFlow:

- TensorFlow is an open-source machine learning framework developed by Google.
- It provides a comprehensive ecosystem of tools, libraries, and community resources for building and deploying machine learning models.
- TensorFlow offers flexibility, scalability, and performance for a wide range of tasks, including deep learning, reinforcement learning, and symbolic mathematics.

8.3.3. OpenCV (Open-Source Computer Vision Library):

- OpenCV is a popular open-source computer vision and machine learning software library.
- It provides a wide range of functionalities for image and video processing, including feature detection, object recognition, image segmentation, and camera calibration.
- OpenCV is widely used in various applications such as robotics, augmented reality, face recognition, and medical image analysis.

8.3.4. gTTS (Google Text-to-Speech):

- gTTS is a Python library and CLI tool for converting text to speech using Google Text-to-Speech API.
- It allows users to generate audio files from text strings in various languages and with customizable parameters such as language, speed, and pitch.
- gTTS is useful for applications requiring speech synthesis, such as voice assistants, accessibility tools, and language learning apps.

8.3.5. Playsound:

- Playsound is a Python library for playing audio files.
- It provides a simple interface for playing audio files in various formats (e.g., WAV, MP3) using platform-specific multimedia libraries.
- Playsound is lightweight and easy to use, making it suitable for basic audio playback tasks in Python scripts and applications.

Feature Extraction and Representation:

The representation of an image as a 3D matrix having dimension as of height and width of the image and the value of each pixel as depth (1 in case of Grayscale and 3 in case of RGB). Further, these pixel values are used for extracting useful features using CNN.

Artificial Neural Networks:

Artificial Neural Network is a connection of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer.

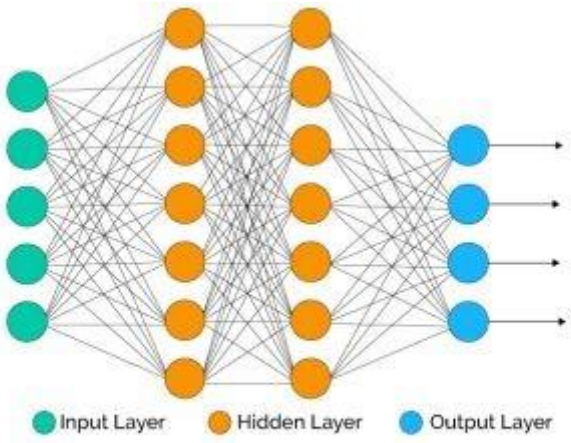


Figure 5.1: Artificial neural networks

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

1.Unsupervised Learning

2.Supervised Learning

3.Reinforcement Learning

8.4. Convolution Neural Network:

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

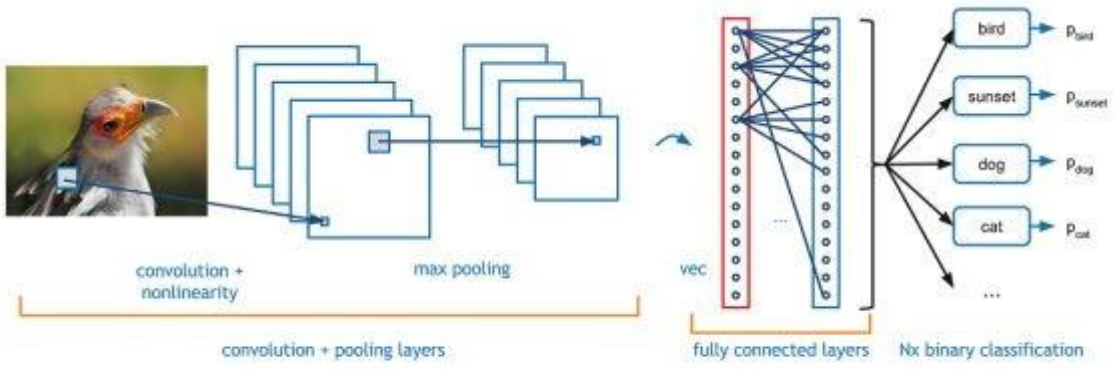


Figure 5.2: Convolution neural networks

1.Convolution 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 well 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 color.

2.Pooling Layer:

We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type 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 a activation matrix half of its original Size.

b) Average Pooling: In average pooling we take average of all values in a window.

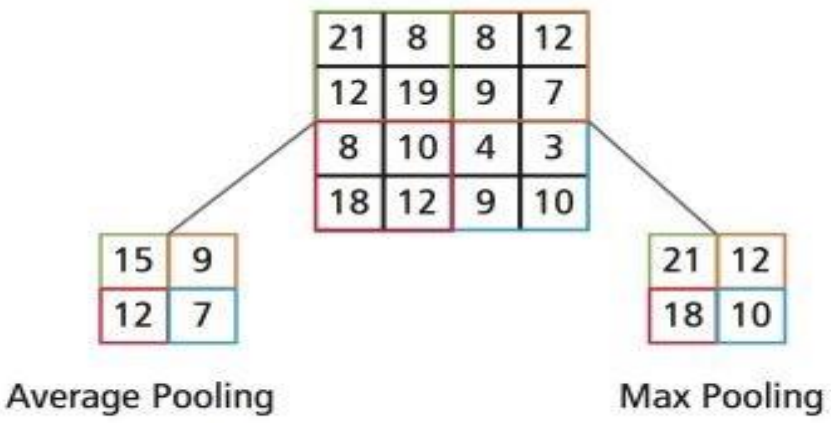


Figure 5.3: Types of pooling

Fully Connected Layer: In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.

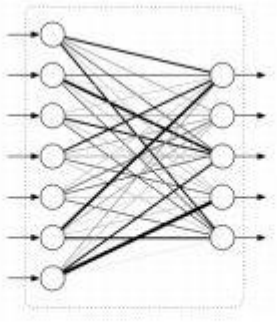


Figure 5.4: Fully Connected Layer

Final Output Layer: After getting values from fully connected layer, well connect them to 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.

8.5. VS code

Visual Studio Code (VS Code) is a free, open-source source code editor developed by Microsoft. It has quickly gained popularity among developers due to its lightweight, fast, and highly customizable nature. Here's a brief overview of VS Code-

Cross-Platform Support:

VS Code is available for Windows, macOS, and Linux, providing a consistent development experience across different operating systems.

Rich Feature Set:

VS Code comes with a rich set of features, including syntax highlighting, code completion, linting, debugging, version control integration, and built-in terminal.

Extensions Ecosystem:

One of the most powerful aspects of VS Code is its extensive extension ecosystem. Developers can enhance their workflow by installing extensions for additional languages, frameworks, tools, and themes.

VS Code supports various programming languages out of the box, including JavaScript, Python, Java, C++, and many more. Additionally, there are thousands of community-contributed extensions available in the Visual Studio Code Marketplace.

Intelligent Code Editing:

VS Code provides intelligent code editing features such as IntelliSense (code completion), code refactoring, and automatic formatting, which help developers write code faster and with fewer errors.

Integrated Terminal:

VS Code includes an integrated terminal that allows developers to run command-line tools and scripts without leaving the editor, streamlining the development workflow.

Built-in Git Integration:

Git integration is built directly into VS Code, allowing developers to perform version control operations (such as committing, branching, and merging) without switching to a separate Git client.

Customizable and Configurable:

VS Code is highly customizable and configurable, enabling developers to personalize their editor experience by adjusting settings, keybindings, and themes according to their preferences.

Active Development and Community Support:

VS Code is actively developed by Microsoft, with regular updates and new features being released frequently. It also has a large and vibrant community of users and contributors who actively participate in discussions, issue reporting, and extension development.

8.6. AWS

Amazon Web Services (AWS) is a comprehensive cloud computing platform offered by Amazon.com. It provides a wide range of services including computing power, storage solutions, networking, databases, machine learning, and more. Here's a brief overview of AWS cloud and the process of creating a virtual machine (EC2 instance) in AWS-

AWS Cloud:

- AWS offers a scalable and flexible cloud computing infrastructure that allows businesses to access computing resources on-demand, without the need to invest in physical hardware or manage data centers.
- With AWS, users can quickly deploy virtual servers, storage solutions, databases, and other services, paying only for the resources they use on a pay-as-you-go basis.
- AWS offers a global network of data centers, enabling users to deploy applications and services closer to their end-users for improved performance and reliability.
- AWS provides a wide range of services and tools for developers, IT professionals, and businesses to build, deploy, and manage applications and infrastructure in the cloud.

Creating a Virtual Machine (EC2 Instance) in AWS:

- Log in to the AWS Management Console: Visit the AWS Management Console at https://console.aws.amazon.com/ and sign in with your AWS account credentials.
- Navigate to EC2 Dashboard: Click on the "Services" dropdown menu and select "EC2"

under the "Compute" section to navigate to the EC2 dashboard.

- Launch Instance: Click on the "Launch Instance" button to start the process of launching a new virtual machine (EC2 instance).
- Choose an Amazon Machine Image (AMI): Select an AMI from the available options. AMIs are pre-configured templates that contain the operating system and other software required to run your instance.
- Choose Instance Type: Select an instance type based on your compute requirements. AWS offers a variety of instance types optimized for different workloads, ranging from general-purpose instances to compute-optimized and memory-optimized instances.
- Configure Instance Details: Configure instance settings such as network, subnet, security groups, and storage options. You can also specify additional configuration details such as IAM role, monitoring, and advanced settings.
- Add Storage: Specify the storage options for your instance, including the root volume size and additional attached volumes if needed.
- Configure Security Group: Define security group rules to control inbound and outbound traffic to your instance. You can specify which ports are open and the source IP addresses allowed to access your instance.
- Review and Launch: Review your instance configuration settings and click on the "Launch" button to launch the instance.
- Create Key Pair: Create a new key pair or select an existing key pair to securely connect to your instance via SSH (Linux) or RDP (Windows).
- Access Instance: Once the instance is launched, you can access it using SSH (for Linux instances) or RDP (for Windows instances) using the provided public IP address or DNS hostname.

Module Description

Our provided code is designed to perform hand gesture recognition using a Convolutional Neural Network (CNN) with Keras and OpenCV. Below is the module design summarizing the functionalities of different parts of the code-

1. User Interface Module:

- This module is responsible for creating a graphical user interface (GUI) using the Tkinter library.
- It includes buttons for uploading the hand gesture dataset, training the CNN model, uploading test images for classification, and recognizing gestures from video input.
- The module displays text information and updates the GUI accordingly.

2. Data Processing Module:

- The data processing module involves functions for loading and preprocessing the hand gesture dataset.
- It includes functionality to resize, convert to grayscale, and normalize the images before feeding them into the CNN model.

3. CNN Model Module:

- This module defines the architecture of the CNN model using Keras Sequential API.
- It includes convolutional layers, max-pooling layers, flattening layer, fully connected layers, and an output layer.
- The model is compiled with appropriate optimizer, loss function, and evaluation metrics.

4. Training Module:

- The training module is responsible for training the CNN model using the hand gesture dataset.
- It loads the preprocessed dataset, initializes the CNN model, and trains it using the fit method.
- The trained model and its weights are saved for future use.

5. Image Classification Module:

- This module provides functionality to upload test images and classify hand gestures using the trained CNN model.
- It preprocesses the test images similar to the training images and feeds them into the trained model for prediction.

- The predicted gesture labels are displayed on the GUI along with the original test images.

6. Video Classification Module:

- The video classification module is responsible for real-time gesture recognition from video input.
- It captures video frames using OpenCV, preprocesses them, and feeds them into the trained CNN model for prediction.
- The predicted gesture labels are overlaid on the video frames and displayed in real-time on the GUI.

7. Audio Feedback Module:

- This module provides audio feedback by converting the predicted gesture labels into speech using the gTTS (Google Text-to-Speech) library.
- It generates audio files for each recognized gesture label and plays them using the playsound library.

8. Background Subtraction Module:

- This module implements background subtraction using OpenCV to isolate hand gestures from the background in video input.
- It removes background noise and enhances the accuracy of gesture recognition.

9. Utility Functions Module:

- The utility functions module includes helper functions used throughout the code, such as functions to get gesture labels, play audio files, and handle file dialogs.
- By modularizing the code in this way, each module encapsulates specific functionalities, making the codebase more organized, maintainable, and reusable. Additionally, it allows for easier debugging, testing, and future enhancements.

CODE

Code for hand gesture recognition:

```
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.np_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
from keras.models import model_from_json
import pickle
import os
from gtts import gTTS
from playsound import playsound
from threading import Thread

main = tkinter.Tk()
main.title("HAND GESTURE RECOGNITION")
main.geometry("1300x1200")

global filename
global classifier

bg = None
playcount = 0
names = ['Palm','I','Fist','Fist Moved','Thumb','Index','OK','Palm Moved','C','Down']
```

```
bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)

def getID(name):
    index = 0
    for i in range(len(names)):
        if names[i] == name:
            index = i
            break
    return index

def play(playcount,gesture):
    class PlayThread(Thread):
        def __init__(self,playcount,gesture):
            Thread.__init__(self)
            self.gesture = gesture
            self.playcount = playcount

        def run(self):
            t1 = gTTS(text=self.gesture, lang='en', slow=False)
            t1.save("play/"+str(self.playcount)+".mp3")
            playsound("play/"+str(self.playcount)+".mp3")


    newthread = PlayThread(playcount,gesture)
    newthread.start()


def remove_background(frame):
    fgmask = bgModel.apply(frame, learningRate=0)
    kernel = np.ones((3, 3), np.uint8)
    fgmask = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=fgmask)
    return res

def uploadDataset():
```

```
    global filename
    global labels
    labels = []
    filename = filedialog.askdirectory(initialdir=".")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n\n");


def trainCNN():
    global classifier
    text.delete('1.0', END)
    X_train = np.load('model/X.txt.npy')
    Y_train = np.load('model/Y.txt.npy')
    text.insert(END,"CNN is training on total images : "+str(len(X_train))+"\n")

    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            classifier = model_from_json(loaded_model_json)
        classifier.load_weights("model/model_weights.h5")
        classifier._make_predict_function()
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[19] * 100
        text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))
    else:
        classifier = Sequential()
        classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
        classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

```
    classifier.add(Flatten())
    classifier.add(Dense(output_dim = 256, activation = 'relu'))
    classifier.add(Dense(output_dim = 5, activation = 'softmax'))
    print(classifier.summary())
    classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
    hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=10, shuffle=True, verbose=2)
    classifier.save_weights('model/model_weights.h5')
    model_json = classifier.to_json()
    with open("model/model.json", "w") as json_file:
        json_file.write(model_json)
    f = open('model/history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    acc = data['accuracy']
    accuracy = acc[19] * 100
    text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy = "+str(accuracy))



def classifyFlower():
    filename = filedialog.askopenfilename(initialdir="testImages")
    img = cv2.imread(filename, cv2.IMREAD_COLOR)
    img = cv2.flip(img, 1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (41, 41), 0)  #tuple indicates blur value
    ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    thresh = cv2.resize(thresh, (224, 224))
    thresh = np.array(thresh)
    frame = np.stack((thresh,)*3, axis=-1)
    frame = cv2.resize(frame, (64, 64))
    frame = frame.reshape(1, 64, 64, 3)
    frame = np.array(frame, dtype='float32')
```

```
    frame /= 255
    predict = classifier.predict(frame)
    result = names[np.argmax(predict)]
    img = cv2.imread(filename)
    img = cv2.resize(img, (600,400))
    cv2.putText(img, 'Hand Gesture Classified as : '+result, (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
    cv2.imshow('Hand Gesture Classified as : '+result, img)
    cv2.waitKey(0)



def webcamPredict():
    global playcount
    oldresult = 'none'
    videofile = askopenfilename(initialdir = "video")
    video = cv2.VideoCapture(videofile)
    while(video.isOpened()):
        ret, frame = video.read()
        if ret == True:
            img = frame
            img = cv2.flip(img, 1)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray, (41, 41), 0)  #tuple indicates blur value
            ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            thresh = cv2.resize(thresh, (64, 64))
            thresh = np.array(thresh)
            img = np.stack((thresh,)*3, axis=-1)
            img = cv2.resize(img, (64, 64))
            img = img.reshape(1, 64, 64, 3)
            img = np.array(img, dtype='float32')
            img /= 255
            predict = classifier.predict(img)
            print(np.argmax(predict))
            value = np.amax(predict)
            result = names[np.argmax(predict)]
```

```
            if value >= 0.99:
                print(str(value)+" "+str(result))
                cv2.putText(frame, 'Gesture Recognize as : '+str(result), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
                if oldresult != result:
                    play(playcount,result)
                    oldresult = result
                    playcount = playcount + 1
            else:
                cv2.putText(frame, '', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 255), 2)
            cv2.imshow("video frame", frame)

            # cv2.putText(frame, 'Gesture Recognize as : '+str(result), (10, 25),
cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 255, 255), 2)
            # cv2.imshow("video frame", frame)

            if cv2.waitKey(950) & 0xFF == ord('q'):
                break
        else:
            break
    video.release()
    cv2.destroyAllWindows()
font = ('times', 20, 'bold')
title = Label(main, text='''                    Hand Gesture Recognition Using Automatic Feature
Extraction and Deep Learning Algorithms with Memory''',anchor=W, justify=CENTER)
title.config(bg='#FBCEB1', fg='#660000')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=0)

font1 = ('times', 13, 'bold')
upload = Button(main, text="Upload Hand Gesture Dataset", command=uploadDataset)
upload.place(x=50,y=150)
upload.config(font=font1)

pathlabel = Label(main)
```

```
pathlabel.config(bg='yellow4', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50,y=250)

markovButton = Button(main, text="Train CNN with Gesture Images", command=trainCNN)
markovButton.place(x=50,y=350)
markovButton.config(font=font1)

lexButton = Button(main, text="Upload Test Image & Recognize Gesture", command=classifyFlower)
lexButton.place(x=50,y=450)
lexButton.config(font=font1)

predictButton = Button(main, text="Recognize Gesture from Video", command=webcamPredict)
predictButton.place(x=50,y=550)
predictButton.config(font=font1)


font1 = ('times', 12, 'bold')
text=Text(main,height=15,width=78)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=450,y=150)
text.config(font=font1)

main.config(bg='#FFFF99')
main.mainloop()
```

Code for test dataset:

```
import os
import cv2
import numpy as np
from keras.utils.np_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
from keras.models import model_from_json
import pickle

# Non-Binary Image Classification using Convolution Neural Networks

names = ['Palm','I','Fist','Fist Moved','Thumb','Index','OK','Palm Moved','C','Down']

'''
path = 'Dataset'

labels = []
X_train = []
Y_train = []

for root, dirs, directory in os.walk(path):
    for j in range(len(directory)):
        name = os.path.basename(root)
        print(name+" "+root+"/"+directory[j])
        if 'Thumbs.db' not in directory[j]:
            img = cv2.imread(root+"/"+directory[j], cv2.IMREAD_COLOR)
            img = cv2.flip(img, 1)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray, (41, 41), 0)  #tuple indicates blur value
            ret, thresh = cv2.threshold(blur, 150, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
            thresh = cv2.resize(thresh, (64, 64))
            thresh = np.array(thresh)
            X_train.append(thresh)
            Y_train.append(int(name))

X_train = np.asarray(X_train)

Y_train = np.asarray(Y_train)
print(Y_train)
print(X_train.shape)
```

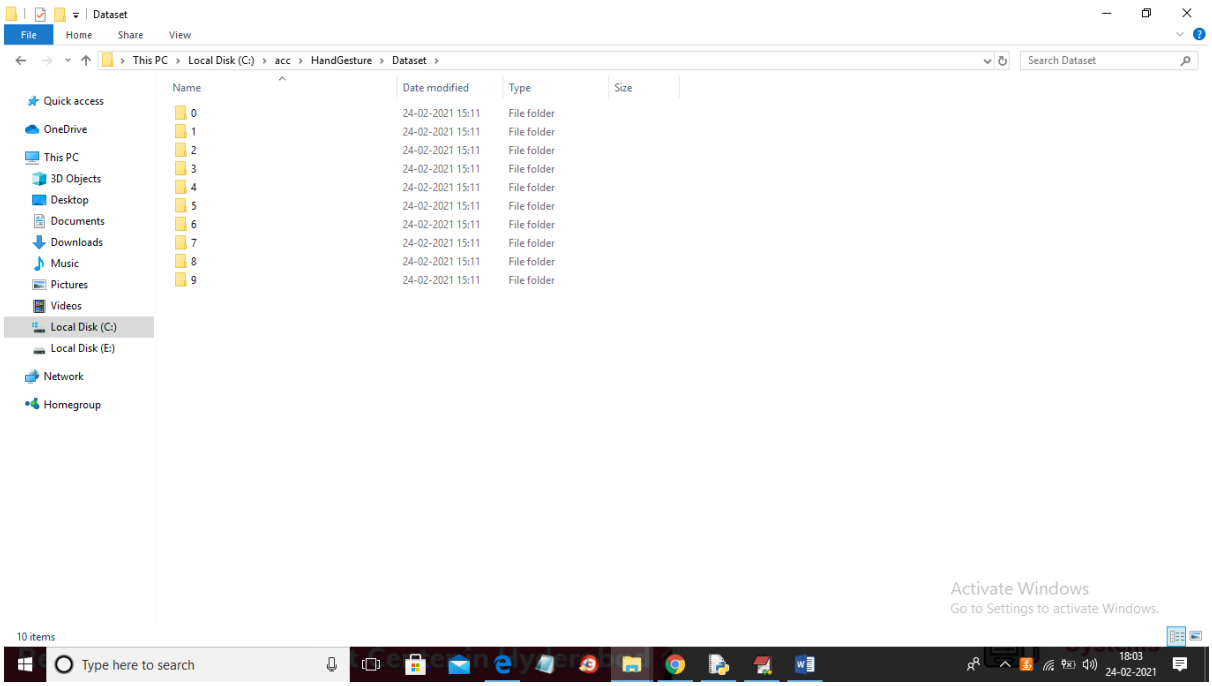
```
X_train = X_train.astype('float32')
X_train = X_train/255

test = X_train[3]
cv2.imshow("aa",test)
cv2.waitKey(0)
indices = np.arange(X_train.shape[0])
np.random.shuffle(indices)
X_train = X_train[indices]
Y_train = Y_train[indices]
Y_train = to_categorical(Y_train)
np.save('model/X.txt',X_train)
np.save('model/Y.txt',Y_train)
'''

X_train = np.load('model/X.txt.npy')
Y_train = np.load('model/Y.txt.npy')
X_train = np.stack((X_train,)*3, axis=-1)
print(Y_train)
print(X_train.shape)
if os.path.exists('model/model.json'):
    with open('model/model.json', "r") as json_file:
        loaded_model_json = json_file.read()
        classifier = model_from_json(loaded_model_json)
    classifier.load_weights("model/model_weights.h5")
    classifier._make_predict_function()
    print(classifier.summary())
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    acc = data['accuracy']
    accuracy = acc[19] * 100
    print("Training Model Accuracy = "+str(accuracy))
else:
```

```
classifier = Sequential()
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Flatten())
classifier.add(Dense(output_dim = 256, activation = 'relu'))
classifier.add(Dense(output_dim = 10, activation = 'softmax'))
print(classifier.summary())
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
hist = classifier.fit(X_train, Y_train, batch_size=64, epochs=20, shuffle=True, verbose=2)
classifier.save_weights('model/model_weights.h5')
model_json = classifier.to_json()
with open("model/model.json", "w") as json_file:
    json_file.write(model_json)
f = open('model/history.pckl', 'wb')
pickle.dump(hist.history, f)
f.close()
f = open('model/history.pckl', 'rb')
data = pickle.load(f)
f.close()
acc = data['accuracy']
accuracy = acc[19] * 100
print("Training Model Accuracy = "+str(accuracy))
```

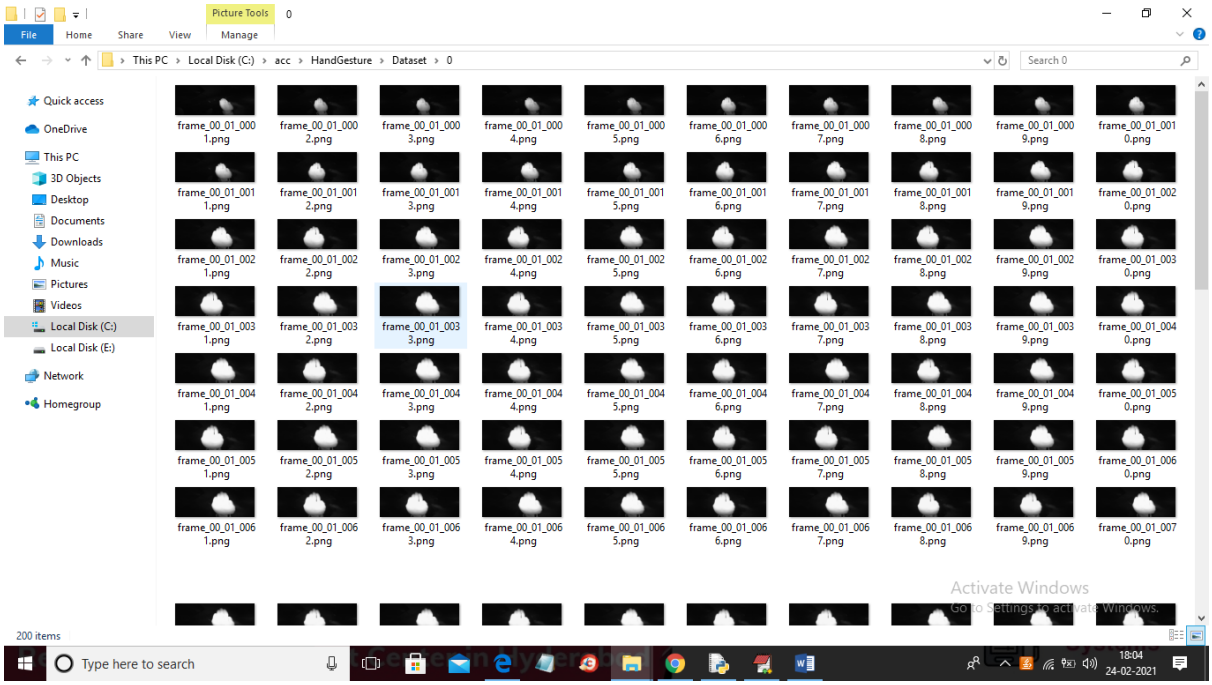
RESULTS

Screen Shots:

In this project using CNN we are recognizing hand gesture movement and to train CNN we are using following images shown in below screen shots

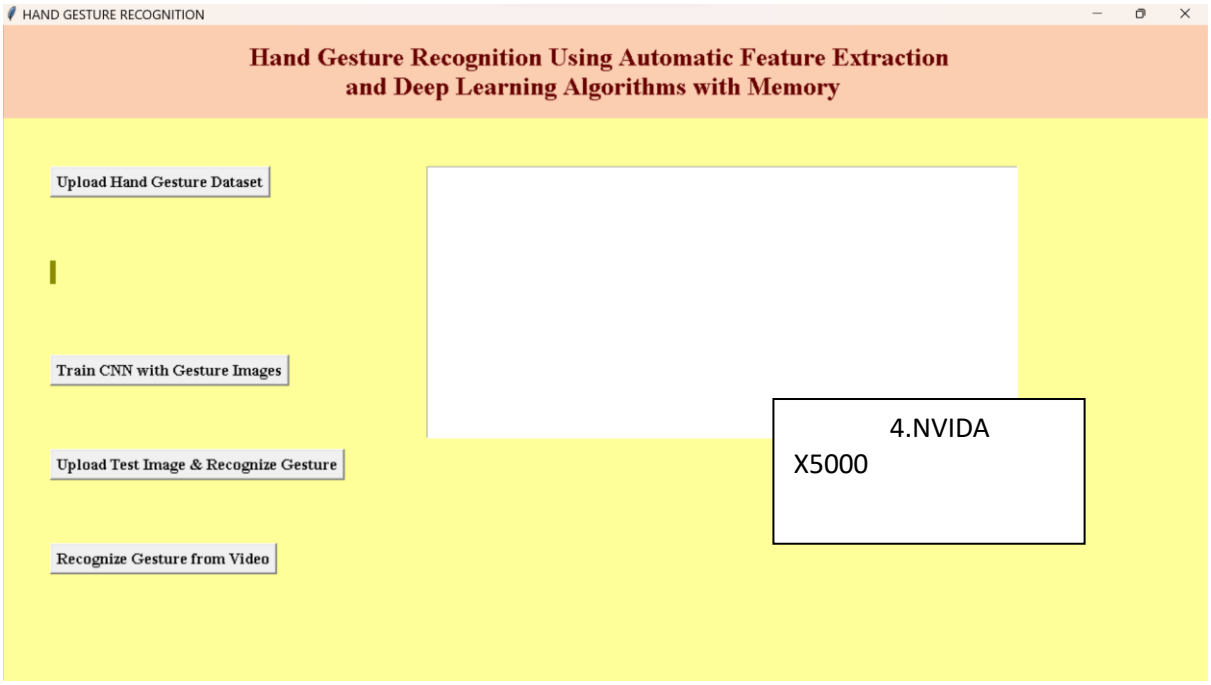


In above screen we can see we have 10 different types of hand gesture images and to see those images just go inside any folder

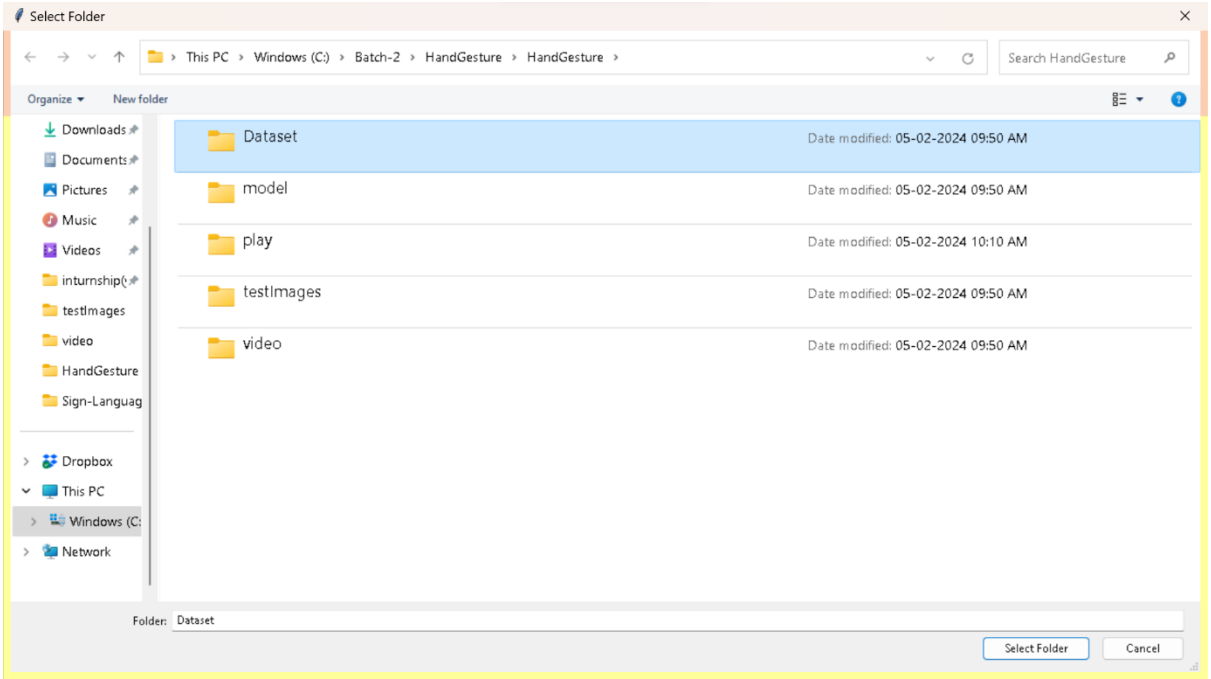


In above screen showing images from 0 folder and similarly you can see different images in different folders.

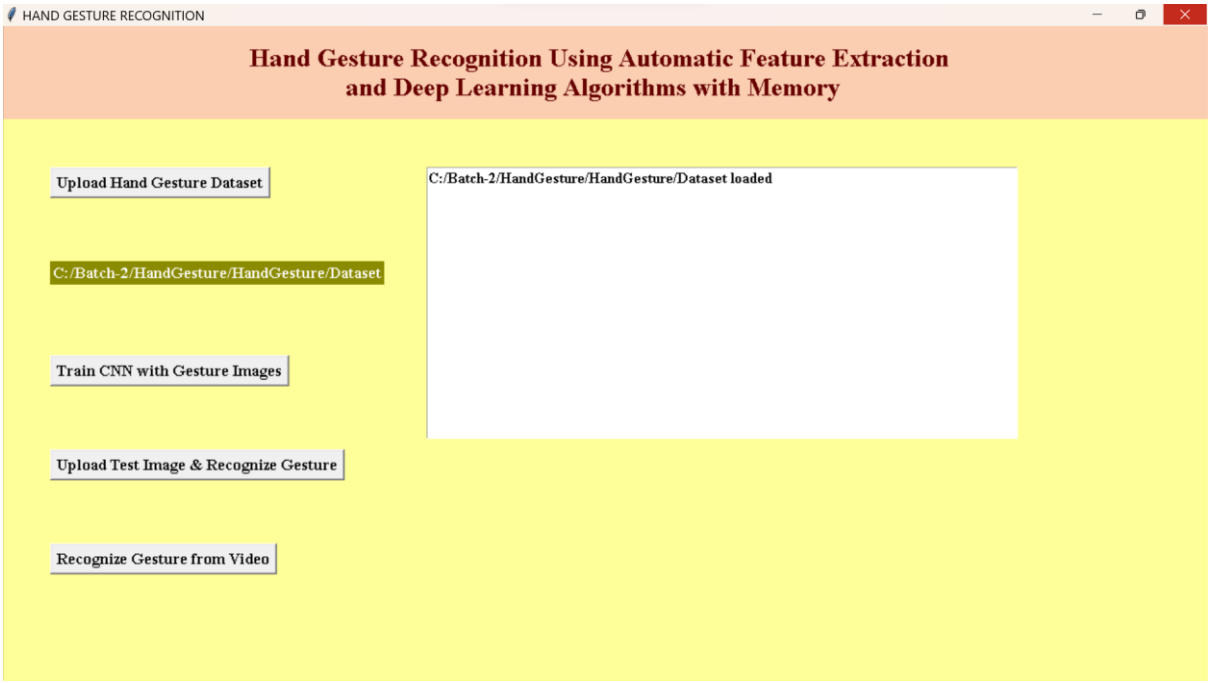
To run project double click on run.bat file to get below screen



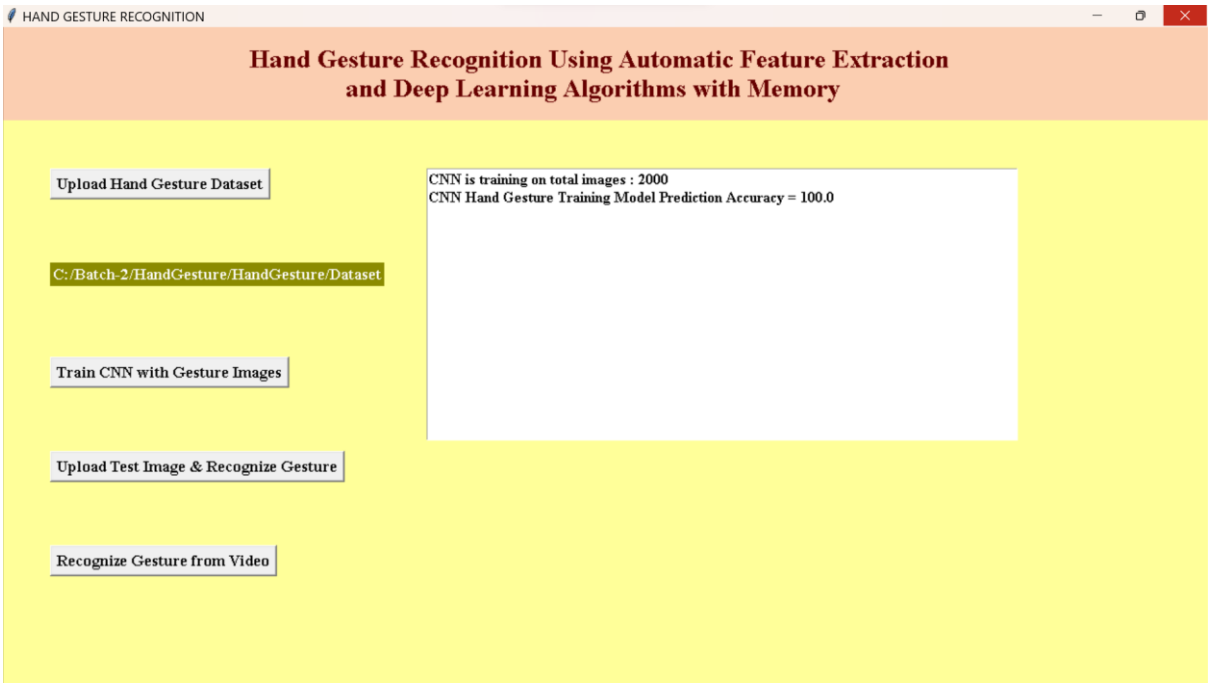
In above screen click on 'Upload Hand Gesture Dataset' button to upload dataset and to get below screen



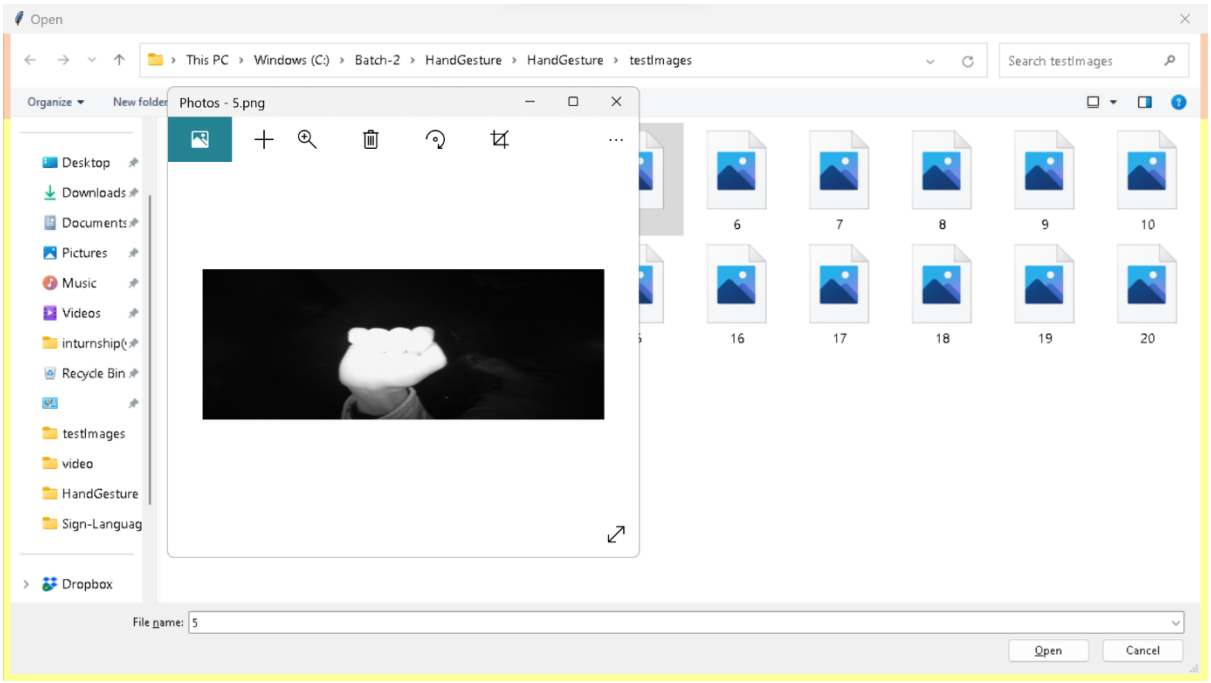
In above screen selecting and uploading 'Dataset' folder and then click on 'Select Folder' button to load dataset and to get below screen



In above screen dataset loaded and now click on 'Train CNN with Gesture Images' button to trained CNN model and to get below screen

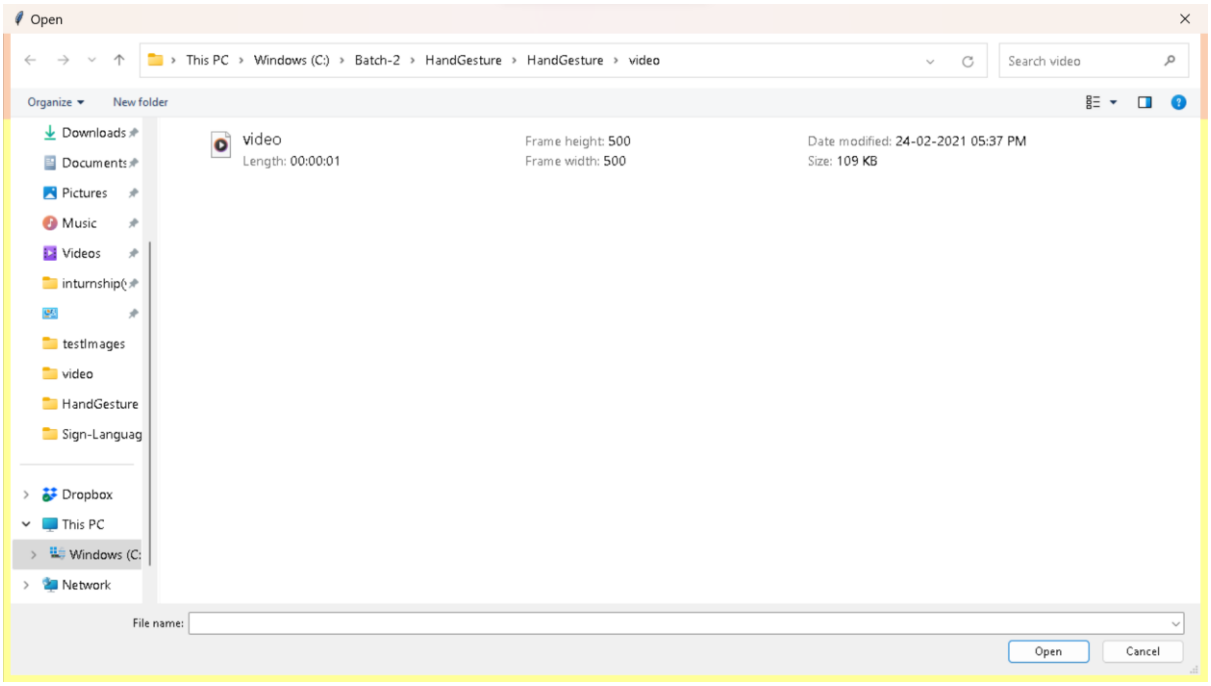


In above screen CNN model trained on 2000 images and its prediction accuracy we got as 100% and now model is ready and now click on 'Upload Test Image & Recognize Gesture' button to upload image and to gesture recognition

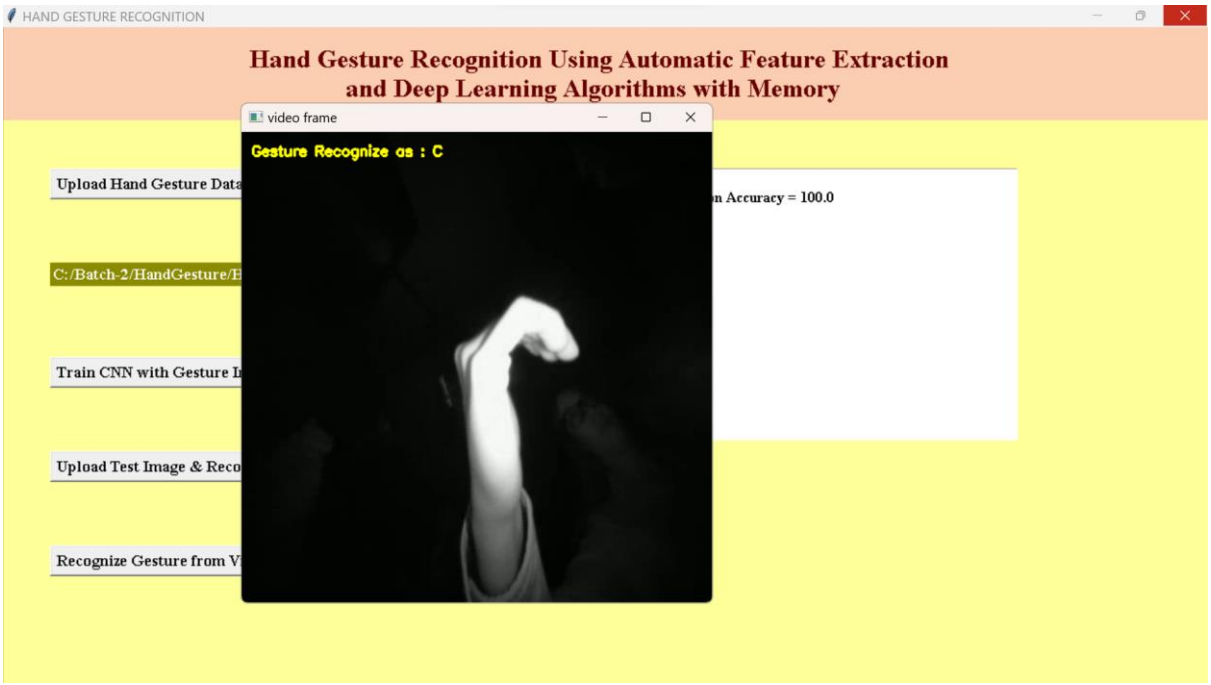


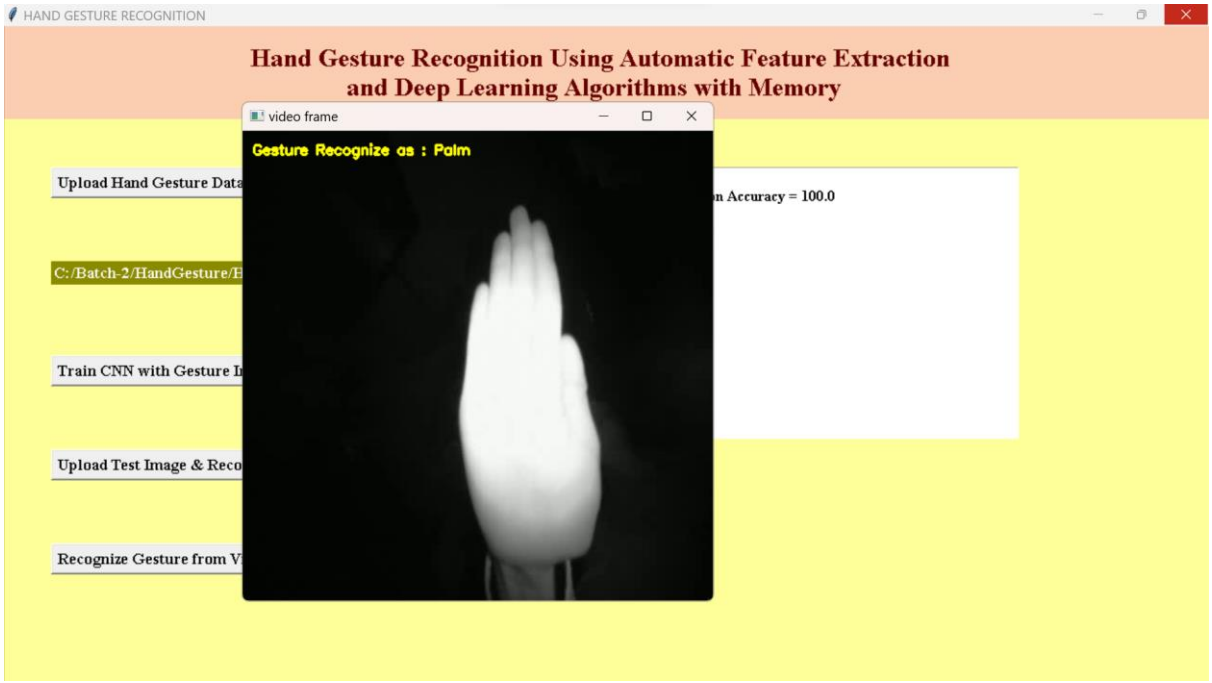
In above screen selecting and uploading '14.png' file and then click Open button to get below result

In above screen gesture recognize as OK and similarly you can upload any image and get result and now click on 'Recognize Gesture from Video' button to upload video and get result



In above screen selecting and uploading ‘video.avi’ file and then click on ‘Open’ button to get below result





In above screen as video play then we will get recognition result

CONCLUSION

We developed a CNN model for sign language recognition. Our model learns and extracts both spatial and temporal features by performing 3D convolutions. The developed deep architecture extracts multiple types of information from adjacent input frames and then performs convolution and subsampling separately. The final feature representation combines information from all channels. We use multilayer perceptron classifier to classify these feature representations. For comparison, we evaluate both CNN and GMM-HMM on the same dataset. The experimental results demonstrate the effectiveness of the proposed method.

FUTURE SCOPE:

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

CHALLENGES FACED:

There were many challenges faced by us during the project. The very first issue we faced was of dataset. We wanted to deal with raw images and that too square images as CNN in Keras as it was a lot more convenient working with only square images. We couldn't find any existing dataset for that hence we decided to make our own dataset. Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provide that image as input for CNN model. We tried various filter including binary threshold, canny edge detection, gaussian blur etc. but finally we settled with gaussian blur filter. More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[2] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in CVPR, 2014. [3] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick ´ Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] Hueihan Jhuang, Thomas Serre, Lior Wolf, and Tomaso Poggio, "A biologically inspired system for action recognition," in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. Ieee, 2007, pp. 1–8. [5] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, "3D convolutional neural networks for human action recognition," IEEE TPAMI, vol. 35, no. 1, pp. 221–231, 2013.

[6] Kirsti Grobel and Marcell Assan, "Isolated sign language recognition using hidden markov models," in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. IEEE, 1997, vol. 1, pp. 162–167.

[7] Thad Starner, Joshua Weaver, and Alex Pentland, "Realtime american sign language recognition using desk and wearable computer based video," IEEE TPAMI, vol. 20, no. 12, pp. 1371–1375, 1998.

[8] Christian Vogler and Dimitris Metaxas, "Parallel hidden markov models for american sign language recognition," in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. IEEE, 1999, vol. 1, pp. 116–122.

[9] Kouichi Murakami and Hitomi Taguchi, "Gesture recognition using recurrent neural networks," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 1991, pp. 237–242.

[10] Chung-Lin Huang and Wen-Yi Huang, "Sign language recognition using model-based tracking and a 3D hopfield neural network," Machine vision and applications, vol. 10, no. 5-6, pp. 292–307, 1998.

[11] Jong-Sung Kim, Won Jang, and Zeungnam Bien, “A dynamic gesture recognition system for the korean sign language (ksl),” Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 26, no. 2, pp. 354–359, 1996.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” arXiv preprint arXiv:1311.2524, 2013.

[13] Ronan Collobert and Jason Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in ICML. ACM, 2008, pp. 160–167.

[14] Clement Farabet, Camille Couprie, Laurent Najman, ´ and Yann LeCun, “Learning hierarchical features for scene labeling,” IEEE TPAMI, vol. 35, no. 8, pp. 1915– 1929, 2013. [15] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung, “Convolutional networks can learn to generate affinity graphs for image segmentation,” Neural Computation, vol. 22, no. 2, pp. 511– 538, 2010.

[16] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li, “A real-time hand posture recognition system using deep neural networks,” ACM Transactions on Intelligent Systems and Technology, 2014.

[17] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt, “Sequential deep learning for human action recognition,” in Human Behavior Understanding, pp. 29–39. Springer, 2011.