An Industrial Oriented Mini Project Report

On

"Sign Language Recognition Using CNN"

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirement for the award of Degree of

BACHELOR OF TECHNOLOGY

In

"Computer Science & Engineering"

By

D Neeraja [21E31A0526]

B Sriven [21E31A0512]

MD Aqeeb [21E31A0557]

Under the guidance of

Ms. G.Rajini

Assistant Professor, C.S.E

Department of Computer Science & Engineering

MAHAVEER INSTITUTE OF SCIENCE & TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE,Accridited with NAAC A Grade)

Vyasapuri, Bandlaguda, Post: Keshavgiri, Hyderabad-500005

2024-2025

MAHAVEER INSTITUTE OF SCIENCE & TECHNOLOGY

(Affiliated to JNTU Hyderabad, Approved by AICTE,Accridited with NAAC A Grade)

Vyasapuri, Bandlaguda, Post: Keshavgiri, Hyderabad-500005

CERTIFICATE

This is to certify that the industrial oriented mini project work report entitled **"Sign Language Recognition Using CNN"** which is being submitted by **D Neeraja [21E31A0526]**,**B Sriven [21E31A0512] , MD Aqeeb [21E31A0557]** in partial fulfilment for the award of the Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING** of **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,** is a record of the Bonafide work carried out by them under our guidance and supervision.

Ms. G.Rajini

Assistant Professor

Dr. R. Nakkeeran, M. E, Ph.D.

Professor & HOD

External Examiner

Dr. V. Usha Shree

Principal

ACKNOWLEDGEMENT

We would like to express our deep felt appreciation and gratitude to **G.Rajini, Assistant. Professor**, Department of C.S.E, our project guide, for his/her skilful guidance, constant supervision, timely suggestion, keen interest and encouragement in completing the individual seminar within the stipulated time.

We wish to express our gratitude to **M. Sreenivas Reddy,** Project Co-ordinator, who has shown keen interest and even rendered his valuable guidance in terms of suggestions and encouragement extended to us with an immense care and zeal.

We express our profound sense of gratitude to **Dr. R. Nakkeeran, M. E, Ph.D**, Professor & Head of Department, C.S.E, who has served as a host of valuable corrections and for providing us time and amenities to complete this project.

We express our thanks to **Dr. V. Usha Shree**, Principal of our college and the management of **Mahaveer Institute of Science & Technology** for providing excellent academic environment in the college.

We wish to express our gratitude to the **Members of staff** and all others who helped us in more than one way. We would also like to thank the **Lab assistants and Programmers** for helping us through our project.

D Neeraja [21E31A0526]

B Sriven [21E31A0512]

MD Aqeeb [21E31A0557]

ABSTRACT

Conversing to a person with hearing disability is always a major challenge. Sign language has indelibly become the ultimate panacea and is a very powerful tool for individuals with hearing and speech disability to communicate their feelings and opinions to the world. It makes the integration process between them and others smooth and less complex. However, the invention of sign language alone, is not enough . There are many strings attached to this boon.The sign gestures often get mixed and confused for someone who has never learnt it or knows it in a different language. However, this communication gap which has existed for years can now be narrowed with the introduction of various techniques to automate the detection of sign gestures . In this paper, we introduce a Sign Language recognition using American Sign Language. In this study, the user must be able to capture images of the hand gesture using web camera and the system shall predict and display the name of the captured image. We use the HSV colour algorithm to detect the hand gesture and set the background to black. The images undergo a series of processing steps which include various Computer vision techniques such as the conversion to grayscale, dilation and mask operation. And the region of interest which, in our case is the hand gesture is segmented. The features extracted are the binary pixels of the images. We make use of Convolutional Neural Network(CNN) for training and to classify the images. We are able to recognise 10 American Sign gesture alphabets with high accuracy. Our model has achieved a remarkable accuracy of above 90%.

TABLE OF COTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter-1

Introduction

As well stipulated by Nelson Mandela[1], “Talk to a man in a language he understands, that goes to his head. Talk to him in his own language, that goes to his heart”, language is undoubtedly essential to human interaction and has existed since human civilisation began. It is a medium humans use to communicate to express themselves and understand notions of the real world. Without it, no books, no cell phones and definitely not any word I am writing would have any meaning. It is so deeply embedded in our everyday routine that we often take it for granted and don’t realise its importance. Sadly, in the fast changing society we live in, people with hearing impairment are usually forgotten and left out. They have to struggle to bring up their ideas, voice out their opinions and express themselves to people who are different to them. Sign language, although being a medium of communication to deaf people, still have no meaning when conveyed to a non-sign language user. Hence, broadening the communication gap.To prevent this from happening, we are putting forward a sign language recognition system. It will be an ultimate tool for people with hearing disability to communicate their thoughts as well as a very good interpretation for non sign language user to understand what the latter is saying. Many countries have their own standard and interpretation of sign gestures. For instance, an alphabet in Korean sign language will not mean the same thing as in Indian sign language. While this highlights diversity, it also pinpoints the complexity of sign languages. Deep learning must be well versed with the gestures so that we can get a decent accuracy. In our proposed system, American Sign Language is used to create our datasets.

Identification of sign gesture is performed with either of the two methods. First is a glove based method whereby the signer wears a pair of data gloves during the capture of hand movements.Second is a vision based method, further classified into static and dynamic recognition[2]. Static deals with the 2dimensional representation of gestures while dynamic is a real time live capture of the gestures

1.1 Aim

The primary aim of this project is to develop an efficient and accurate sign language recognition system using Convolutional Neural Networks (CNNs). This system is designed to bridge the communication gap between individuals using sign language and those unfamiliar with it by interpreting hand gestures and converting them into understandable text or speech.

1.2 Objective

The objective of developing a sign language recognition system using CNNs is to design and implement an efficient, accurate, and user-friendly solution that can automatically interpret and translate sign language gestures into text or speech. This aims to enhance communication for individuals with hearing or speech impairments.The objectives for sign language recognition using Convolutional Neural Networks (CNNs) can be outlined as follows:

1. Gesture Classification: Develop a CNN model capable of accurately classifying various sign language gestures from input images or video frames. This includes recognizing individual signs as well as combinations of signs that form sentences or phrases.

2. Real-Time Processing: Implement a system that can process and recognize sign language gestures in real-time, allowing for immediate translation and interaction. This is crucial for facilitating natural conversations between sign language users and non-users.

3. High Recognition Accuracy: Achieve a high level of accuracy in gesture recognition, minimizing false positives and negatives. This involves optimizing the CNN architecture and training process to ensure reliable performance across diverse datasets.

4. Robustness to Variability: Ensure the model is robust to variations in signing styles, hand shapes, speeds, and environmental conditions (e.g., lighting, background). This includes training the model on a diverse dataset that captures a wide range of signing scenarios.

5. User Adaptability: Design the system to adapt to individual users' signing styles, allowing for personalized recognition that improves over time with user feedback and additional training data.

6. Integration with Natural Language Processing (NLP): Explore the integration of the sign language recognition system with NLP techniques to translate recognized gestures into text or spoken language, enabling seamless communication.

7. User -Friendly Interface: Create an intuitive and accessible user interface that allows both sign language users and non-signers to interact with the system easily. This may include visual feedback, audio output, and options for customization.

8. Evaluation and Benchmarking: Establish a framework for evaluating the performance of the sign language recognition system using standard metrics (e.g., accuracy, precision, recall) and benchmark it against existing solutions to identify areas for improvement.

9. Research Contribution: Contribute to the academic and practical understanding of gesture recognition by publishing findings, sharing datasets, and collaborating with other researchers in the field of computer vision and machine learning.

Chapter-2

System Analysis

2.1 Existing System and Disadvantages

Existing System:

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.

Disadvantages:

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.

2.2 Proposed System Description & Advantages

Proposed System:

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

Advantage:

1) High accuracy

2) High efficiency

2.3 Software Requirements & Specification

2.3.1 Hardware Requirements:

• System	: intel Core i3.
• Hard Disk	: 500 GB.
• RAM	: 4 GB.

2.3.2 Software Requirements:

• Operating System	: Windows 8
• IDE	**:** Visual Studio Code
• Software	: Python 3.7

2.4 Methodology

The first step of the proposed system is to collect data. Many researchers have used sensors or cameras to capture the hand movements. For our system, we make use of the web camera to shoot the hand gestures.The images undergo a series of processing operations whereby the backgrounds are detected and eliminated using the colour extraction algorithm HSV(Hue,Saturation,Value). Segmentation is then performed to detect the region of the skin tone. Using the morphological operations, a mask is applied on the images and a series of dilation and erosion using elliptical kernel are executed . With openCV, the images obtained are amended to the same size so there is no difference between images of different gestures . Our dataset has 2000 American sign gesture images out of which 1600 images are for training and the rest 400 are for testing purposes. It is in the ratio 80:20. Binary pixels are extracted from each frame, and Convolutional Neural Network is applied for training and classification. The model is then evaluated and the system would then be able to predict the alphabets.

2.5 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.

2.6 Algorithm CNN

Convolutional Neural Network is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as **h * w * d**, where h= height w= width and d= dimension. For example, An RGB image is **6 * 6 * 3** array of the matrix, and the grayscale image is **4 * 4 * 1** array of the matrix.

In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.

Chapter-3

Literature Survey

Literature review of our proposed system shows that there have been many explorations done to tackle the sign recognition in videos and images using several methods and algorithms.

Siming He[4] proposed a system having a dataset of 40 common words and 10,000 sign language images. To locate the hand regions in the video frame, Faster R-CNN with an embedded RPN module is used. It improves performance in terms of accuracy. Detection and template classification can be done at a higher speed as compared to single stage target detection algorithm such as YOLO. The detection accuracy of Faster R-CNN in the paper increases from 89.0% to 91.7% as compared to Fast-RCNN. A 3D CNN is used for feature extraction and a sign-language recognition framework consisting of long and short time memory (LSTM) coding and decoding network are built for the language image sequences.On the problem of RGB sign language image or video recognition in practical problems , the paper merges the hand locating network, 3D CNN feature extraction network and LSTM encoding and decoding to construct the algorithm for extraction. This paper has achieved a recognition of 99% in common vocabulary dataset.

Let's approach the research done by Rekha, J[5]. which made use of YCbCr skin model to detect and fragment the skin region of the hand gestures. Using Principal Curvature based Region Detector, the image features are extracted and classified with Multi class SVM, DTW and non-linear KNN. A dataset of 23 Indian Sign Language static alphabet signs were used for training and 25 videos for testing. The experimental result obtained were 94.4%for static and 86.4% for dynamic. In [6] , a low cost approach has been used for image processing . The capture of images was done with a green background so that during processing, the green colour can be easily subtracted from the RGB colourspace and the image gets converted to black and white. The sign gestures were in Sinhala language. The method that thy have proposed in the study is to map the signs using centroid method. It can map the input gesture with a database irrespective of the hands size and position. The prototype has correctly recognised 92% of the sign gestures.

The paper by M. Geetha and U. C. Manjusha[7], make use of 50 specimens of every alphabets and digits in a vision based recognition of Indian Sign Language characters and numerals using B-Spine approximations. The region of interest of the sign gesture is analysed and the boundary is removed . The boundary obtained is further transformed to a B-spline curve by using the Maximum Curvature Points(MCPs) as the Control points. The B-spline curve undergoes a series of smoothening process so features can be extracted. Support vector machine is used to classify the images and the accuracy is 90.00%.

In [8], Pigou used CLAP14 as his dataset [9]. It consists of 20 Italian sign gestures.After preprocessing the images , he used a Convolutional Neural network model having 6 layers for training. It is to be noted that his model is not a 3D CNN and all the kernels are in 2D. He has used Rectified linear Units (ReLU) as activation functions. Feature extraction is performed by the CNN while classification uses ANN or fully connected layer. His work has achieved an accuracy of 91.70% with an error rate of 8.30%. A similar work was done by J Huang [10].He created his own dataset using Kinect and got a total a total of 25 vocabularies which are used in everyday lives. He then applied a 3D CNN in which all kernels are also in 3D. The input of his model consisted of 5 important channels which are colour-r, colour-b, colour-g, depth and body skeleton. He got an average accuracy of 94.2%.

Chapter-4

System Modules

Modules

Data Collection

Data Processing

Segmentation

Feature Extraction

Classification

1.Data Collection

Data collection is indelibly an essential part in this research as our result highly depends on it. We have therefore created our own dataset of ASL having 2000 images of 10 static alphabet signs. We have 10 classes of static alphabets which are A,B,C,D,K,N,O,T and Y. Two datasets have been made by 2 different signers. Each of them has performed one alphabetical gesture 200 times in alternate lighting conditions . The dataset folder of alphabetic sign gestures is further split into 2 more folders, one for training and the other for testing. Out of the 2000 images captured, 1600 images are used for training and the rest for testing. To get higher consistency, we have captured the photos in the same background with a webcam each time a command is given.The images obtained are saved in the png format .It is to be pinpointed that there is no loss in quality whenever an image in png format is opened ,closed and stored again.PNG is also good in handling high contrast and detailed image. The webcam will capture the images in the RGB colour space.

2.Data Processing

Since the images obtained are in RGB colourspaces, it becomes more difficult to segment the hand gesture based on the skin colour only. We therefore transform the images in HSV colourspace. It is a model which splits the colour of an image into 3 separate parts namely: Hue,Saturation and value. HSV is a powerful tool to improve stability of the images by setting apart brightness from the chromaticity [15]. The Hue element is unaffected by any kind of illumination, shadows and shadings[16] and can thus be considered for background removal. A track-bar having H ranging from 0 to 179, S ranging from 0-255 and V ranging from 0 to 255 is used to detect the hand gesture and set the background to black. The region of the hand gesture undergoes dilation and erosion operations with elliptical kernel.

3.Segmentation

The first image is then transformed to grayscale. As much as this process will result in the loss of colour in the region of the skin gesture, it will also enhance the robustness of our system to changes in lighting or illumination. Non-black pixels in the transformed image are binarised while the others remain unchanged, therefore black.The hand gesture is segmented firstly by taking out all the joined components in the image and secondly by letting only the part which is immensely connected, in our case is the hand gesture. The frame is resized to a size of 64 by 64 pixel. At the end of the segmentation process, binary images of size 64 by 64 are obtained where the area in white represents the hand gesture, and the black coloured area is the rest.

4.Feature Extraction

One of the most crucial part in image processing is to select and extract important features from an image. Images when captured and stored as a dataset usually take up a whole lot of space as they are comprised of a huge amount of data. Feature extraction helps us solve this problem by reducing the data after having extracted the important features automatically.It also contributes in maintaining the accuracy of the classifier and simplifies its complexity. In our case, the features found to be crucial are the binary pixels of the images. Scaling the images to 64 pixels has led us to get sufficient features to effectively classify the American Sign Language gestures . In total, we have 4096 number of features, obtained after multiplying 64 by 64 pixels

5.Classification

In our proposed system, we apply a 2D CNN model with a tensor flow library. The convolution layers scan the images with a filter of size 3 by 3. The dot product between the frame pixel and the weights of the filter are calculated. This particular step extracts important features from the input image to pass on further. The pooling layers are then applied after each convolution layer. One pooling layer decrements the activation map of the previous layer. It merges all the features that were learned in the previous layers' activation maps. This helps to reduce overfitting of the training data and generalises the features represented by the network. In our case, the input layer of the convolutional neural network has 32 feature maps of size 3 by 3, and the activation function is a Rectified Linear Unit. The max pool layer has a size of 2×2. The dropout is set to 50 percent and the layer is flattened. The last layer of the network is a fully connected output layer with ten units, and the activation function is Softmax. Then we compile the model by using category cross-entropy as the loss function and Adam as the optimiser.

Chapter-5

Overview of The Concepts

5.1 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 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

5.1.1 Advantages of Python

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. 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

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.

5.1.2 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.

5.1.3 Disadvantages of Python

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**.**

2.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.

5.2 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.

5.3 Python project modules

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 Python. 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.

Chapter-6

System Design

6.1 System Architecture

The system architecture for sign language recognition using Convolutional Neural Networks (CNNs) typically consists of several key components that work together to process input data, recognize gestures, and provide output.

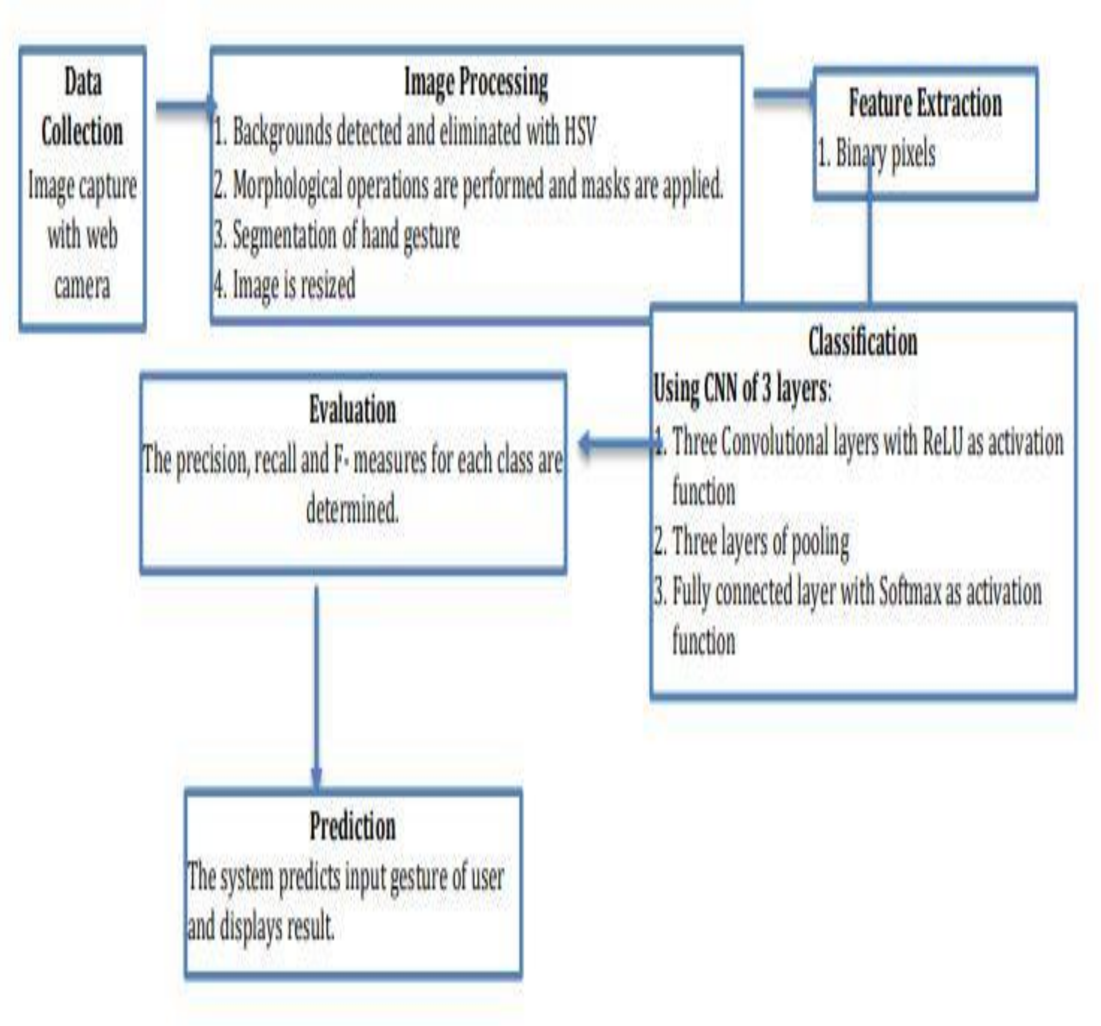

Data Collection
Image capture with web camera
Image Processing
1. Backgrounds detected and eliminated with HSV
2. Morphological operations are performed and masks are applied.
3. Segmentation of hand gesture
4. Image is resized
Feature Extraction
1. Binary pixels
Classification
Using CNN of 3 layers:
1. Three Convolutional layers with ReLU as activation function
2. Three layers of pooling
3. Fully connected layer with Softmax as activation function
Evaluation
The precision, recall and F- measures for each class are determined.
Prediction
The system predicts input gesture of user and displays result.

Fig. 6.1 System Architecture

6.2 UML Diagrams

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

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

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

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

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

Goals:

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

1.Provide users a ready-to-use, expressive visual modeling 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 modeling 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.

6.2.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral 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.

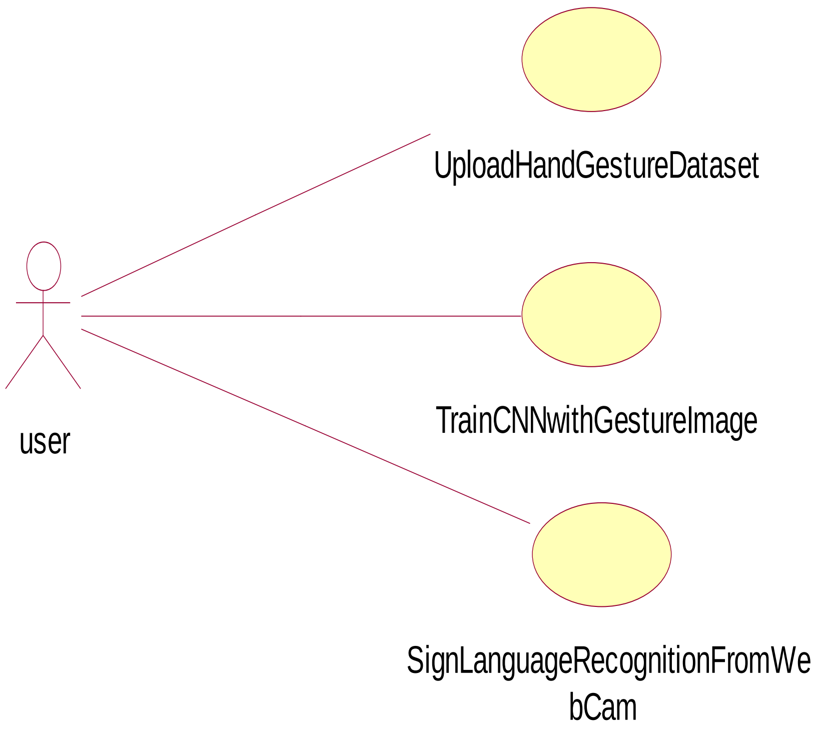


Fig. 6.2.1 Use Case Diagram for user

-

6.2.2 Class Diagram

In software engineering, a class diagram in the Unified Modeling 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 the classes. It explains which class contains information.

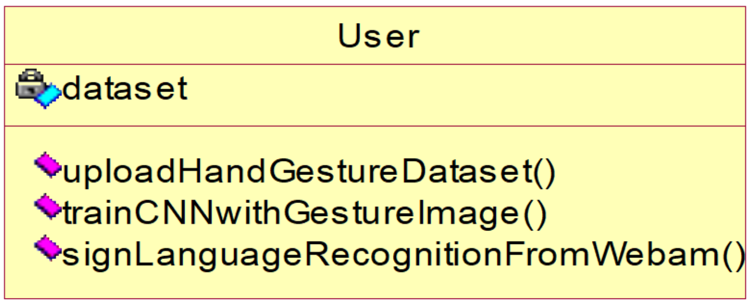

User
dataset
uploadHandGestureDataset()
trainCNNwithGestureImage()
signLanguageRecognitionFromWebam()

Fig 6.2.2 Class Diagram for user

6.2.3 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

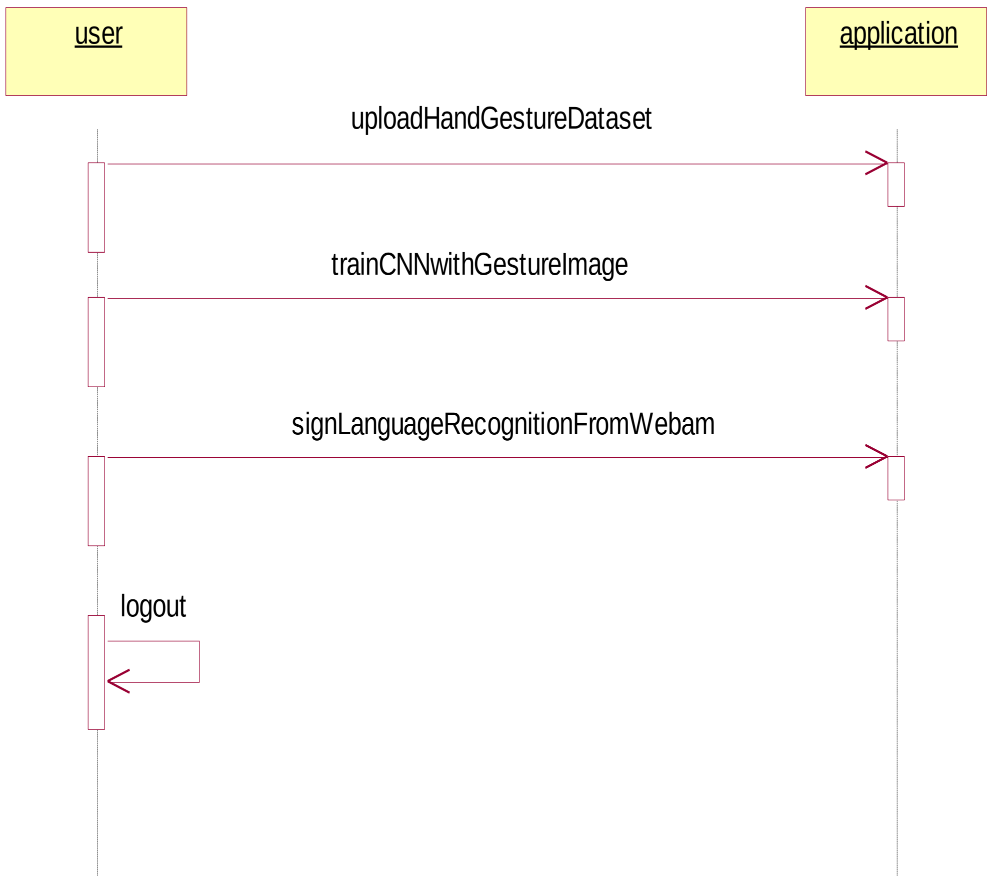


Fig 6.2.3 Sequence Diagram for user

6.2.4 Collaboration Diagram

A Collaboration Diagram (also known as a Communication Diagram) is a type of UML diagram used to show the interactions between objects in a system. It focuses on how objects interact through messages in a specific use case or scenario.

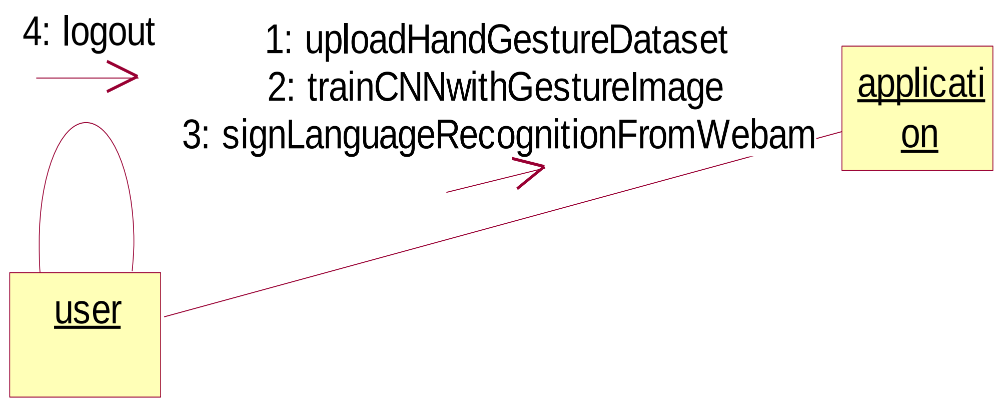

4: logout
1: uploadHandGestureDataset
2: trainCNNwithGestureImage
3: signLanguageRecognitionFromWebam
application
user

Fig 6.2.4 Collaboration Diagram for user

6.3 Database Dictionary

A data dictionary for a sign language recognition system using Convolutional Neural Networks (CNNs) outlines the key data elements, their types, and descriptions that are used throughout the system. This can help in understanding the structure of the data, its purpose, and how it interacts with various components of the system.

Dataset Overview

- Name: Sign Language Gesture Dataset
- Description: A collection of images or video frames of hand gestures representing signs from a specific sign language (e.g., ASL, BSL, or ISL).
- Source: Publicly available datasets, custom-captured data, or synthesized datasets.
- Format: Image files (JPEG, PNG) or video files (MP4, AVI).

6.3.1 Image/Video Data

Field Name	Date Type	Description
image_id	String	Unbique identifier for each image or video frame.
file_path	String	File path to the image or video frame.
image_format	String	Format of the image (e.g., JPEG, PNG).
dimensions	Tuple	Dimensions of the image (e.g., (128, 128, 3) for RGB images).
timestamp	Date/Time	Timestamp for video frames (if applicable).

Fig 6.3.1 Image/Video Data

6.3.2 Labels

Field Name	Date Type	Description
gesture_label	String	The gesture's corresponding sign language label (e.g., "A", "B", "Hello").
class_id	Integer	Numeric representation of the gesture class.
language	String	Language of the sign (e.g., "ASL" for American Sign Language).

Fig 6.3.2 Lables

6.3.3 Metadata

Field Name	Date Type	Description
background_type	String	Type of background in the image (e.g., plain, cluttered).
lighting	String	Lighting conditions during image capture (e.g., bright, dim, natural).
hand_pose	String	Description of the hand pose (e.g., "open palm", "closed fist").
skin_tone	String	Approximate skin tone of the hand in the image.

Fig 6.3.3 Metadata

6.3.4 Preprocessing Fields

Field Name	Date Type	Description
normalized	Boolean	Indicates whether the image has been normalized (True/False).
augmented	Boolean	Indicates whether the image is augmented (True/False).
rotation_angle	Float	Rotation angle applied during augmentation (if any).
scale_factor	Float	Scaling factor applied during augmentation (if any).

Fig 6.3.4 Preprocessing Fields

6.3.5 Model Specific Data

Field Name	Date Type	Description
train_split	Boolean	Indicates whether the image belongs to the training dataset (True/False).
validation_split	Boolean	Indicates whether the image belongs to the validation dataset (True/False).
test_split	Boolean	Indicates whether the image belongs to the testing dataset (True/False).

Fig 6.3.5 Model Specific Data

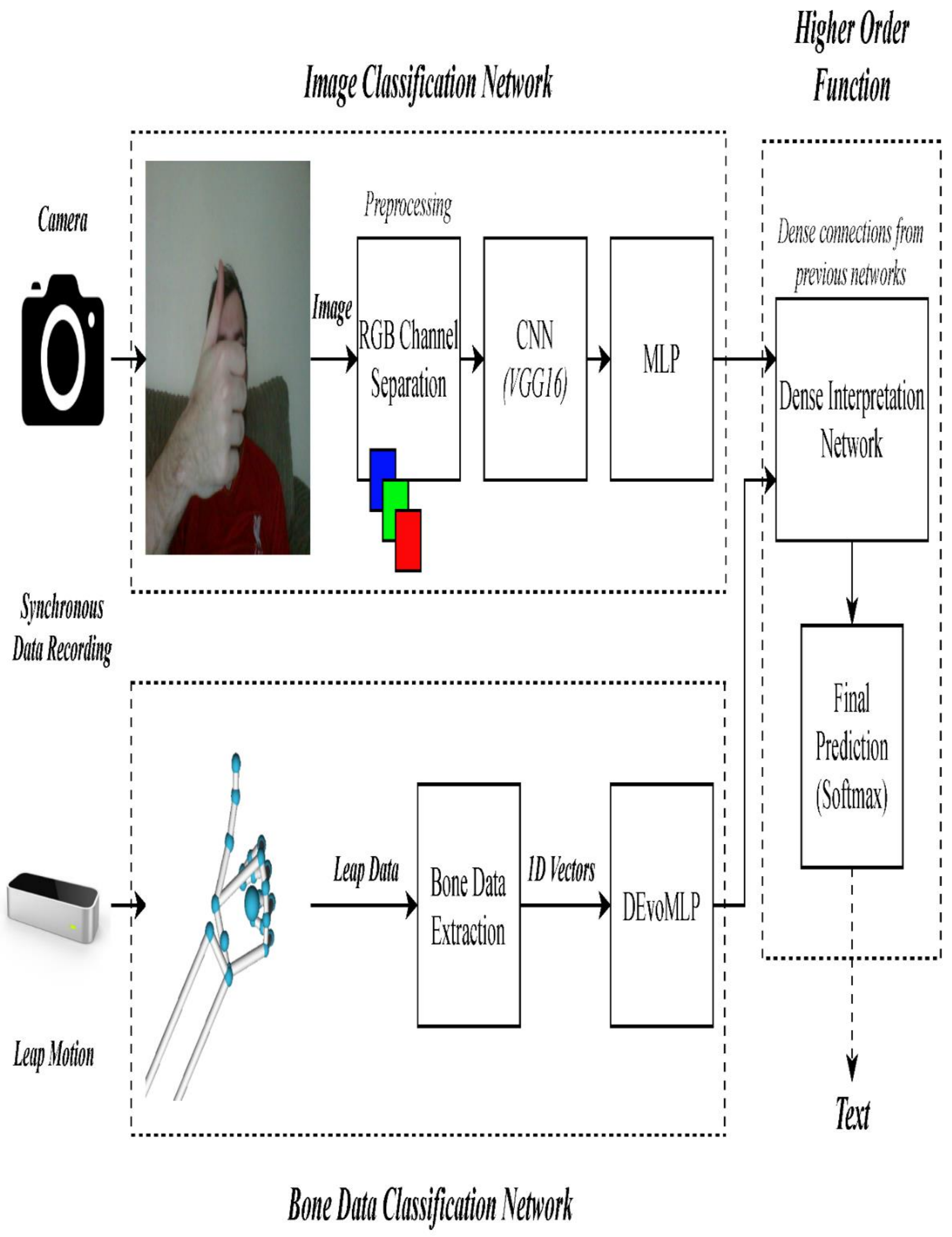
6.3.6 Example Entries

Image_id	File_path	Gesture_label	Class_id	Language	Background_type
IMG_001	data/A/IMG_001.jpg	“A”	1	“ASL”	Plain
IMG_456	data/B/IMG_456.jpg	“B”	2	“ASL”	Cluttered

Fig 6.3.6 Example Entries

6.4 Data Flow Diagram

A Data Flow Diagram (DFD) visually represents the flow of data within a system. For a sign language recognition system using Convolutional Neural Networks (CNN), the DFD can be broken down into several key components, including data sources, processes, data storage, and outputs.


Higher Order Function
Image Classification Network
Camera
Preprocessing
Image
RGB Channel Separation
CNN (VGG16)
MLP
Dense connections from previous networks
Dense Interpretation Network
Synchronous Data Recording
Final Prediction (Softmax)
Leap Data
Bone Data Extraction
1D Vectors
DEvoMLP
Leap Motion
Text
Bone Data Classification Network

Chapter-7

Implementation

7.1 Installation Procedure

How to Install Python on Windows and Mac :

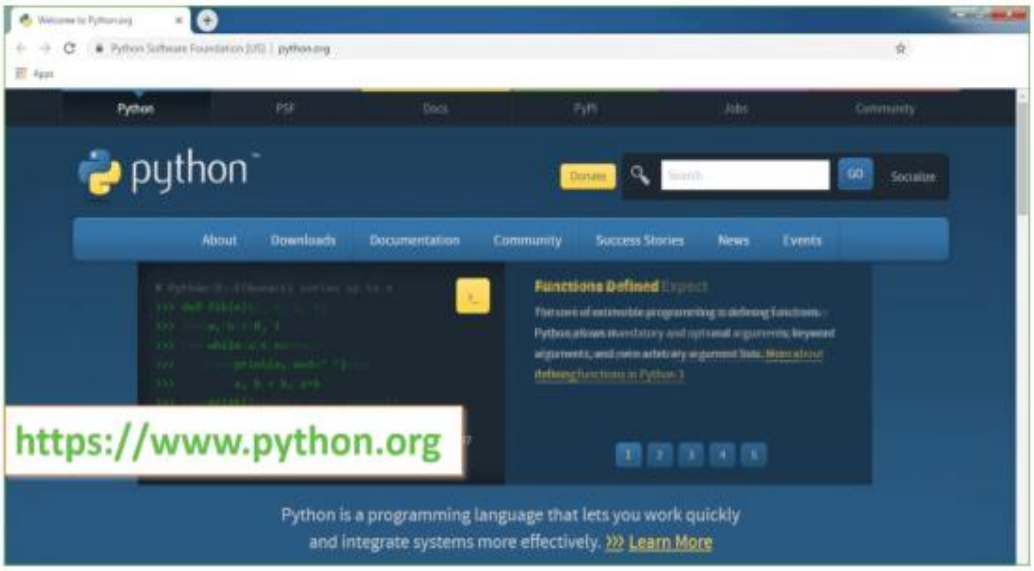
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other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

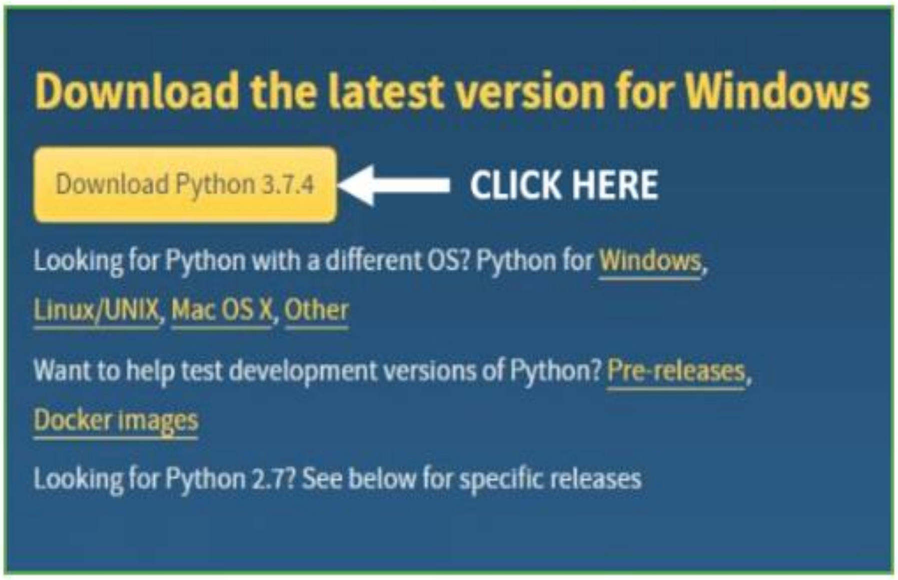
Download the Correct version into the system

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.



Download the latest version for Windows

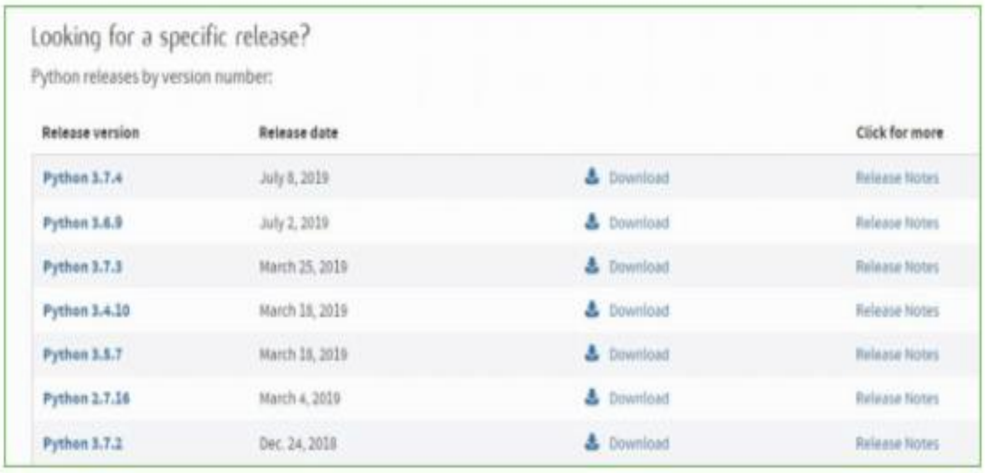
Download Python 3.7.4 CLICK HERE

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

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



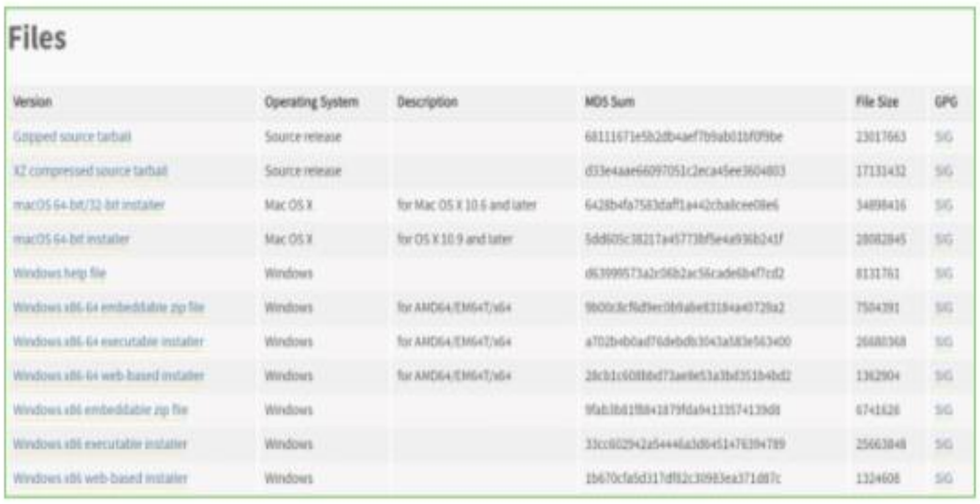
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

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		68111671e5b2db4aef7b9ab01bf0f9be	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		d63999573a2c56b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b6abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	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.

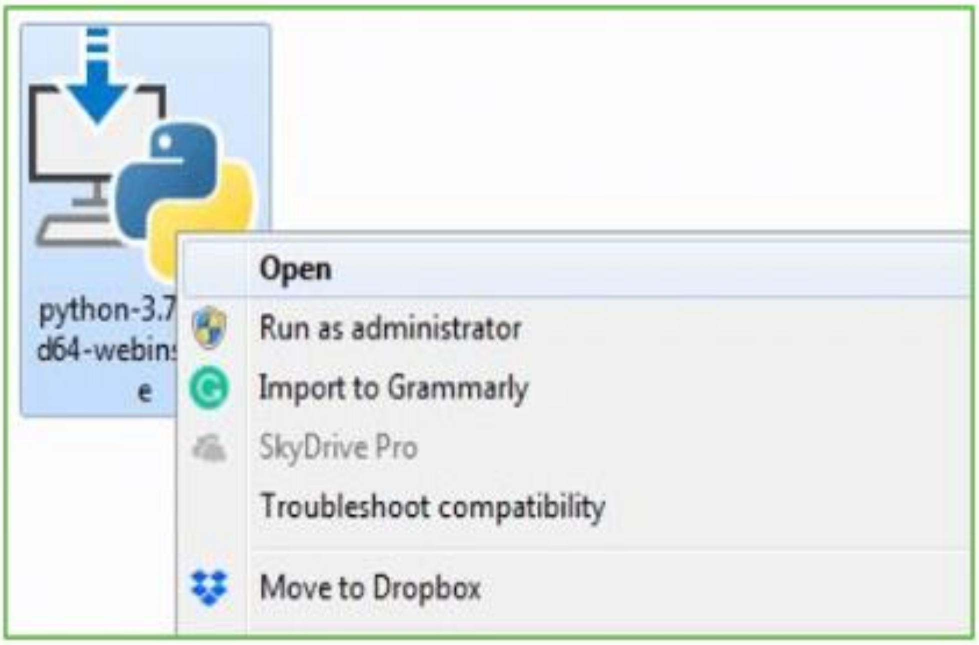
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

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

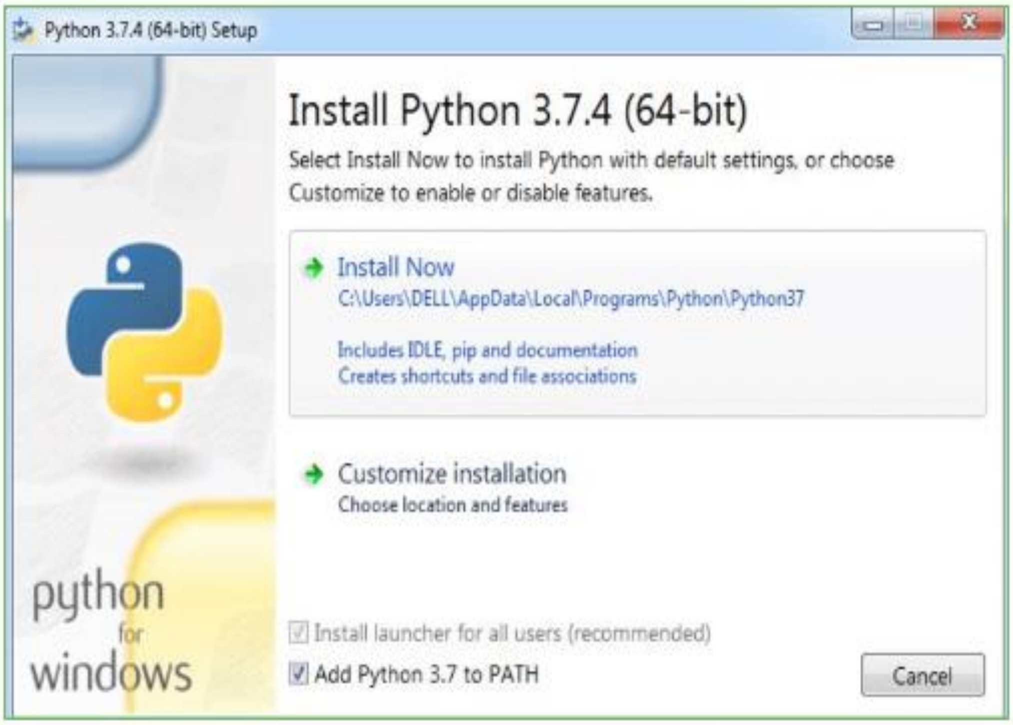
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

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.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

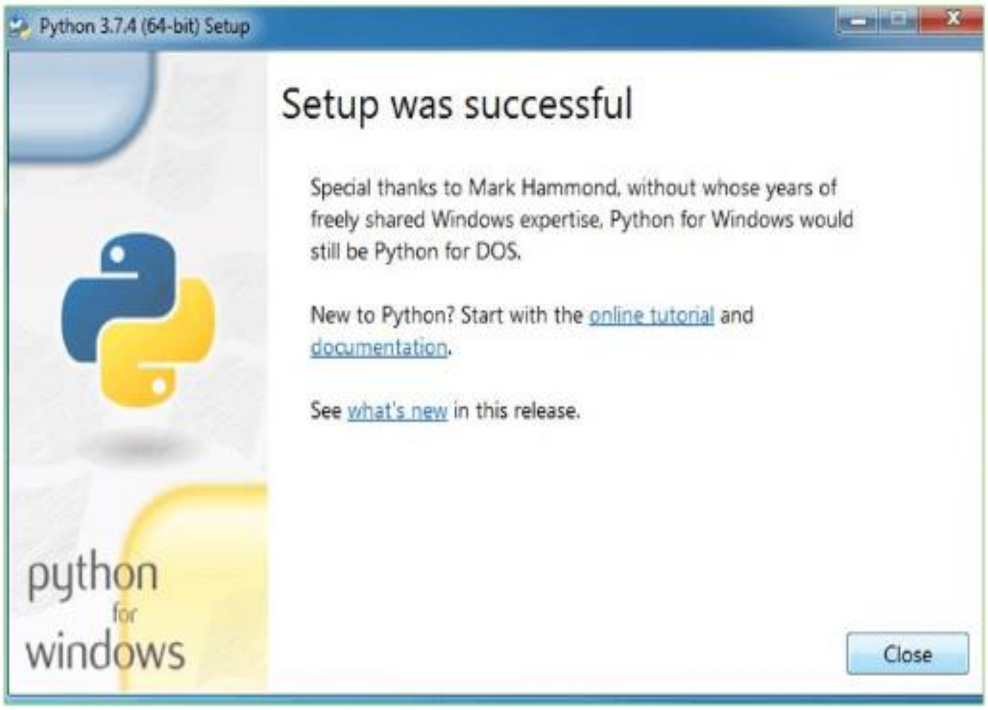
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

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


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

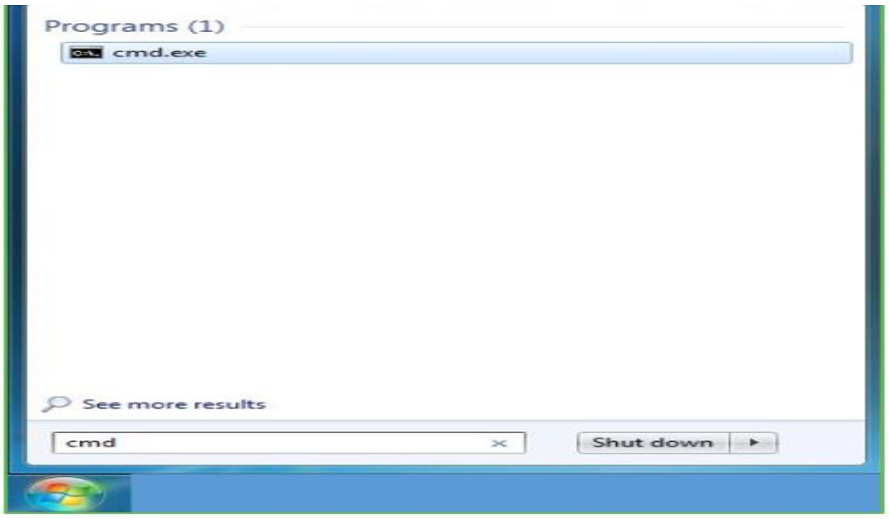
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

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

Verify the Python Installation

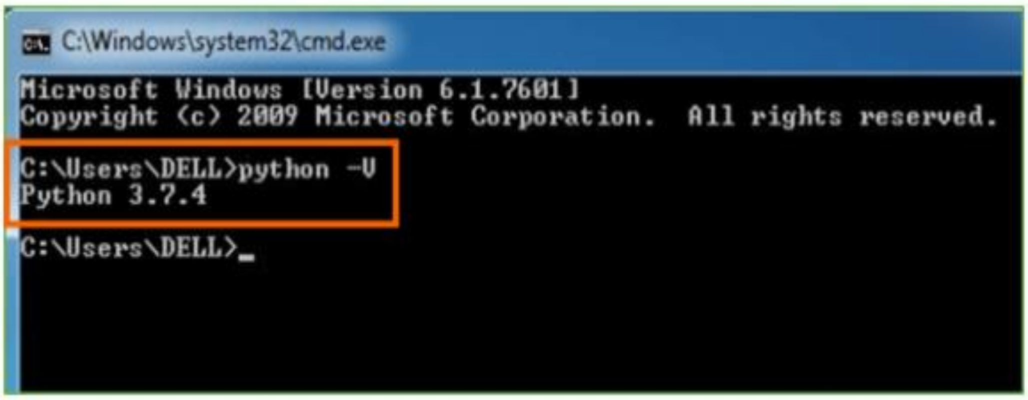
Step 1: Click on Start

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.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

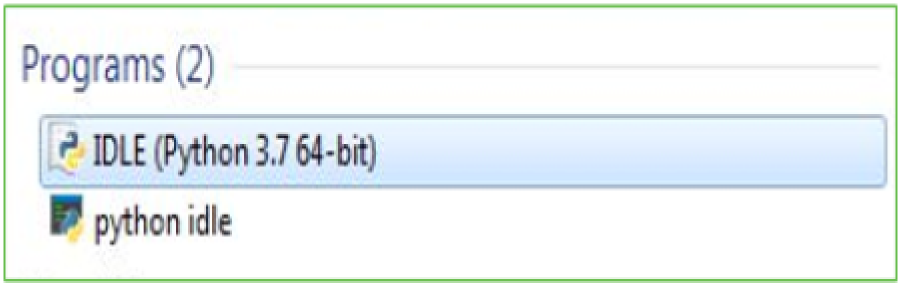
Step 5: You will get the answer as 3.7.4

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

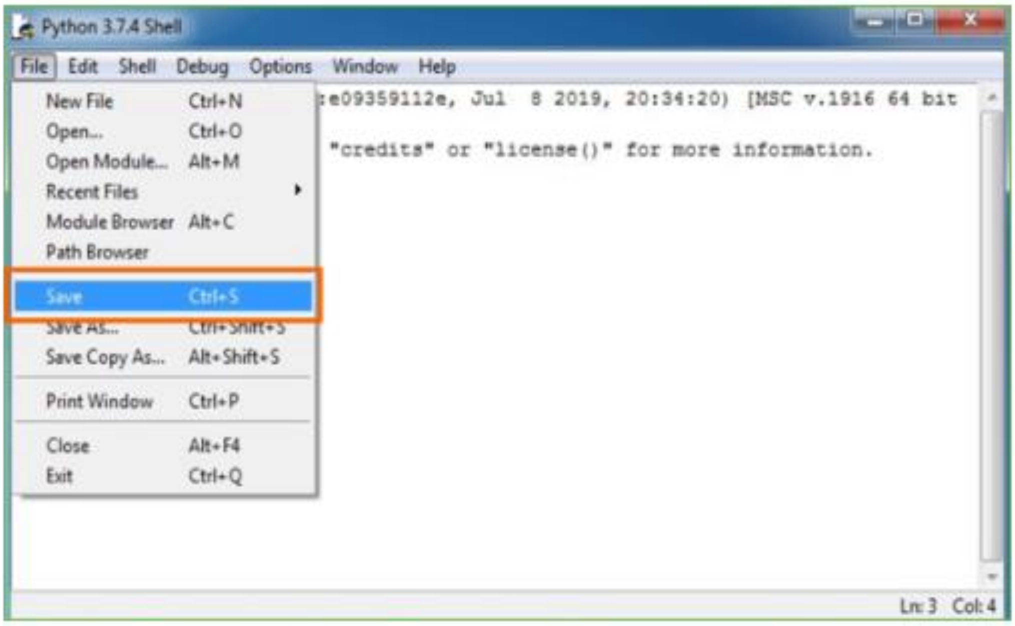
Step 1: Click on Start

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**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 "credits" or "license()" for more information.
```

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**

7.1 Code Description

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

import imutils

from gtts import gTTS

from playsound import playsound

import os

from threading import Thread

main = tkinter.Tk()

main.title("Sign Language Recognition to Text & Voice using CNN Advance")

main.geometry("1300x1200")

global filename

global classifier

bg = None

playcount = 0

#names = ['Palm','I','Fist','Fist Moved','Thumbs up','Index','OK','Palm Moved','C','Down']

names = ['C', 'Down', 'Fist', 'Five', 'I', 'LeftForward', 'Ok', 'Palm', 'Palmmoved', 'Peace', 'Rad', 'RightForward', 'RightReverse', 'Straight', 'Thumb']
```

```
def getID(name):
    index = 0
    for i in range(len(names)):
        if names[i] == name:
            index = i
            break
    return index
bgModel = cv2.createBackgroundSubtractorMOG2(0, 50)
def deleteDirectory():
    filelist = [ f for f in os.listdir('play') if f.endswith(".mp3") ]
    for f in filelist:
        os.remove(os.path.join('play', f))
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')
    print(Y_train.shape)
    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[9] * 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[9] * 100
        text.insert(END,"CNN Hand Gesture Training Model Prediction Accuracy =
"+str(accuracy))
def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)
def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
        ( cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)
def webcamPredict():
    global playcount
    oldresult = 'none'
    count = 0
    fgbg2 = cv2.createBackgroundSubtractorKNN();
    aWeight = 0.5
    camera = cv2.VideoCapture(0)
    top, right, bottom, left = 10, 350, 325, 690
    num_frames = 0
    while(True):
        (grabbed, frame) = camera.read()
```

```
frame = imutils.resize(frame, width=700)

frame = cv2.flip(frame, 1)

clone = frame.copy()

(height, width) = frame.shape[:2]

roi = frame[top:bottom, right:left]

gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (41, 41), 0)

if num_frames < 30:

    run_avg(gray, aWeight)

else:

    temp = gray

    hand = segment(gray)

    if hand is not None:

        (thresholded, segmented) = hand

        cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))

        #cv2.imwrite("test.jpg",temp)

        #cv2.imshow("Thesholded", temp)

            #ret, thresh = cv2.threshold(temp, 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)

        roi = frame[top:bottom, right:left]
        roi = fgbg2.apply(roi);

        cv2.imwrite("test.jpg",roi)

        '''

        cv2.imwrite("newDataset/Straight/"+str(count)+".png",roi)

        count = count + 1

        print(count)

        if count > 400:

            break

        '''

        img = cv2.imread("test.jpg")

        img = cv2.resize(img, (32, 32))

        img = img.reshape(1, 32, 32, 3)

        img = np.array(img, dtype='float32')

        img /= 255

        predict = classifier.predict(img)

        value = np.amax(predict)

        cl = np.argmax(predict)

        result = names[np.argmax(predict)]

        if value >= 0.99:

            print(str(value)+" "+str(result))

                     cv2.putText(clone, '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(clone, '', (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.5, (0,
255, 255), 2)
                cv2.imshow("video frame", roi)
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)
        num_frames += 1
        cv2.imshow("Video Feed", clone)
        keypress = cv2.waitKey(1) & 0xFF
        if keypress == ord("q"):
            break
    camera.release()
    cv2.destroyAllWindows()
font = ('times', 16, 'bold')
title = Label(main, text='Sign Language Recognition to Text & Voice using CNN
Advance',anchor=W, justify=CENTER)
title.config(bg='blue', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

upload = Button(main, text="Upload Hand Gesture Dataset",
command=uploadDataset)

upload.place(x=50,y=100)

upload.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='yellow4', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=50,y=150)

markovButton = Button(main, text="Train CNN with Gesture Images",
command=trainCNN)

markovButton.place(x=50,y=200)

markovButton.config(font=font1)

predictButton = Button(main, text="Sign Language Recognition from Webcam",
command=webcamPredict)

predictButton.place(x=50,y=250)

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=100)

text.config(font=font1)
```

```
deleteDirectory()

main.config(bg='purple')

main.mainloop()
```

Chapter-8

Testing

8.1 Test

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.

8.2 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.

8.3 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.

8.4 System Tests

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.

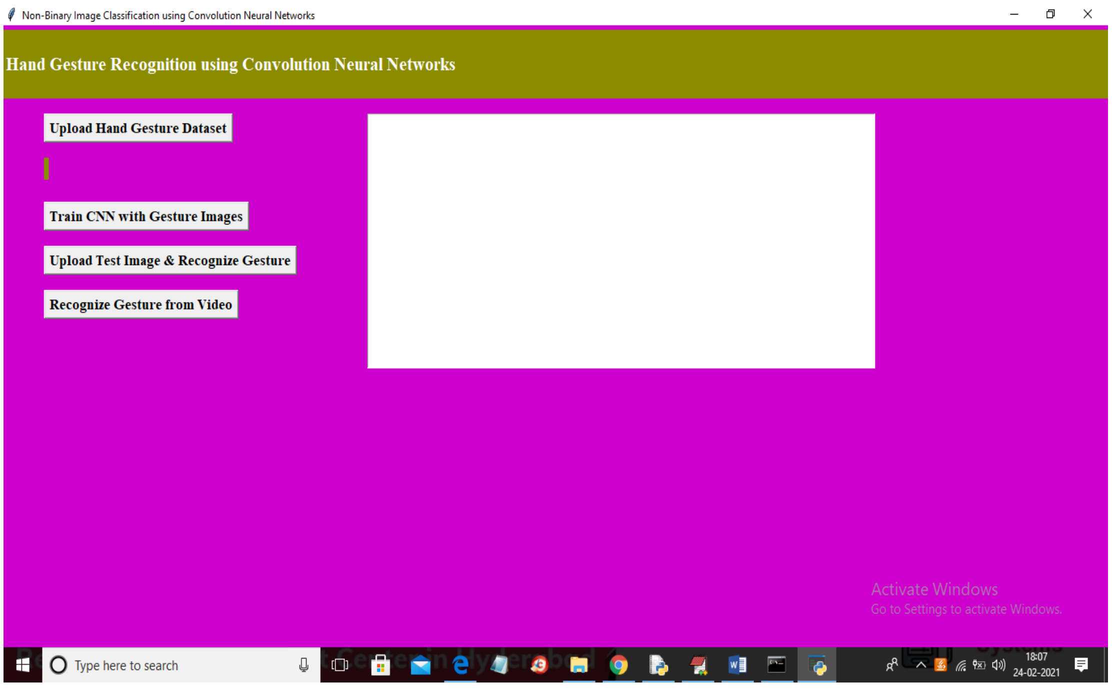
Acceptance Testing

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.

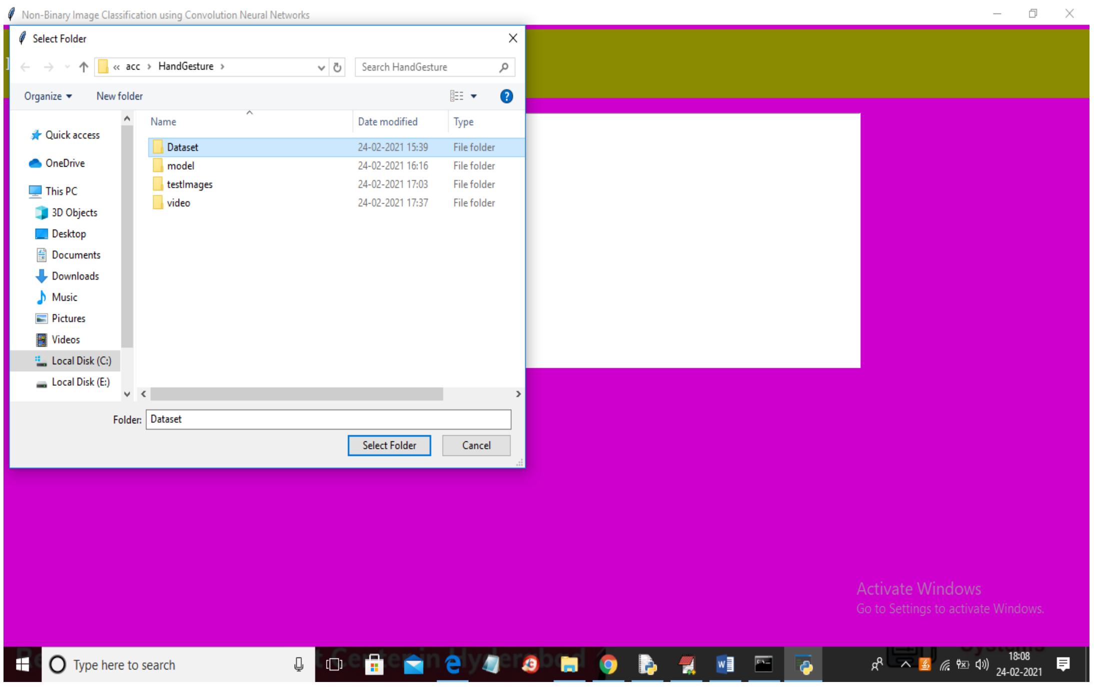
Chapter-9

Output Screens

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

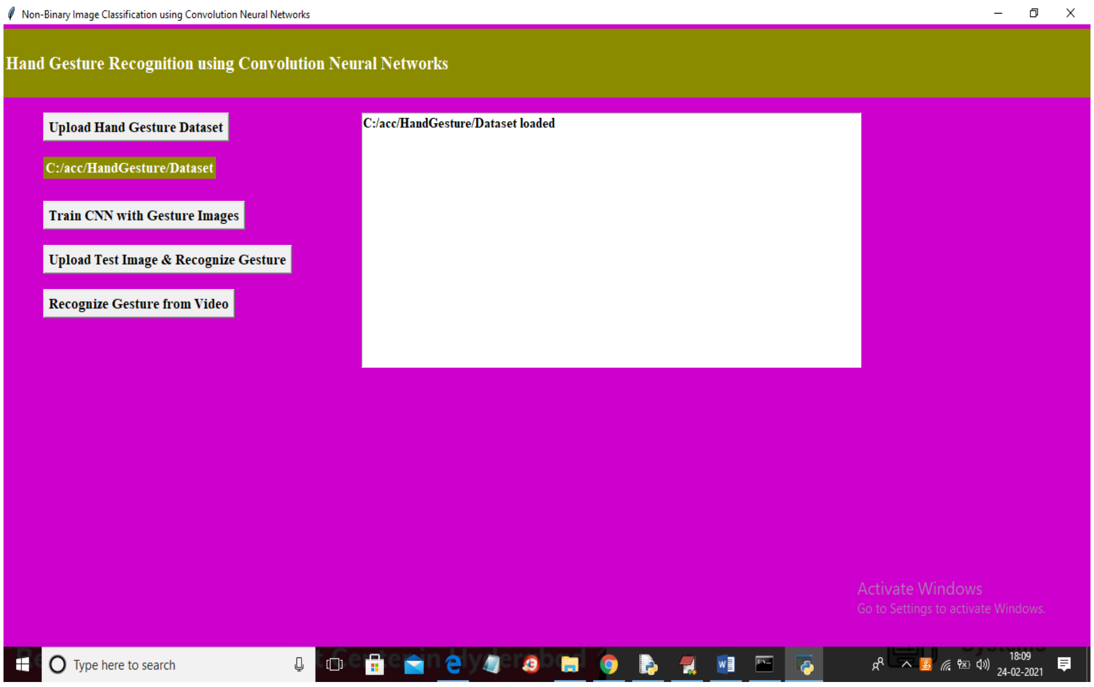

Non-Binary Image Classification using Convolution Neural Networks
Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video
Activate Windows
Go to Settings to activate Windows.
Type here to search
18:07
24-02-2021

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



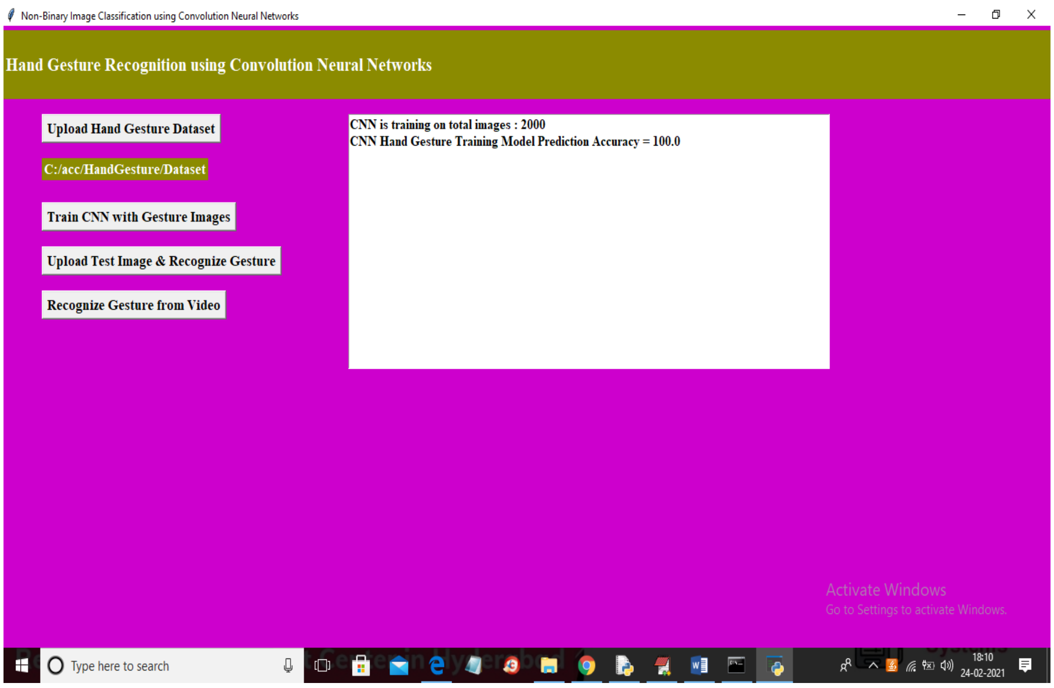
Name	Date modified	Type
Dataset	24-02-2021 15:39	File folder
model	24-02-2021 16:16	File folder
testImages	24-02-2021 17:03	File folder
video	24-02-2021 17:37	File folder

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


Non-Binary Image Classification using Convolution Neural Networks
Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
C:/acc/HandGesture/Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video
Activate Windows
Go to Settings to activate Windows.
Type here to search
18:09
24-02-2021

C:/acc/HandGesture/Dataset loaded

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

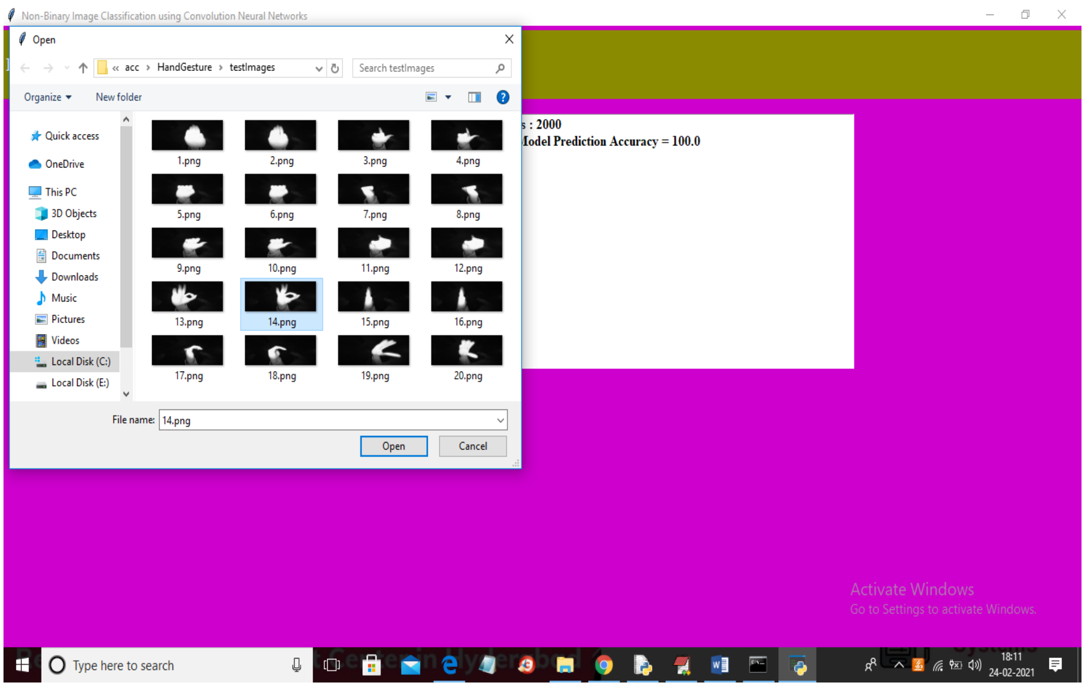

Non-Binary Image Classification using Convolution Neural Networks

Hand Gesture Recognition using Convolution Neural Networks

```
CNN is training on total images : 2000
CNN Hand Gesture Training Model Prediction Accuracy = 100.0
```

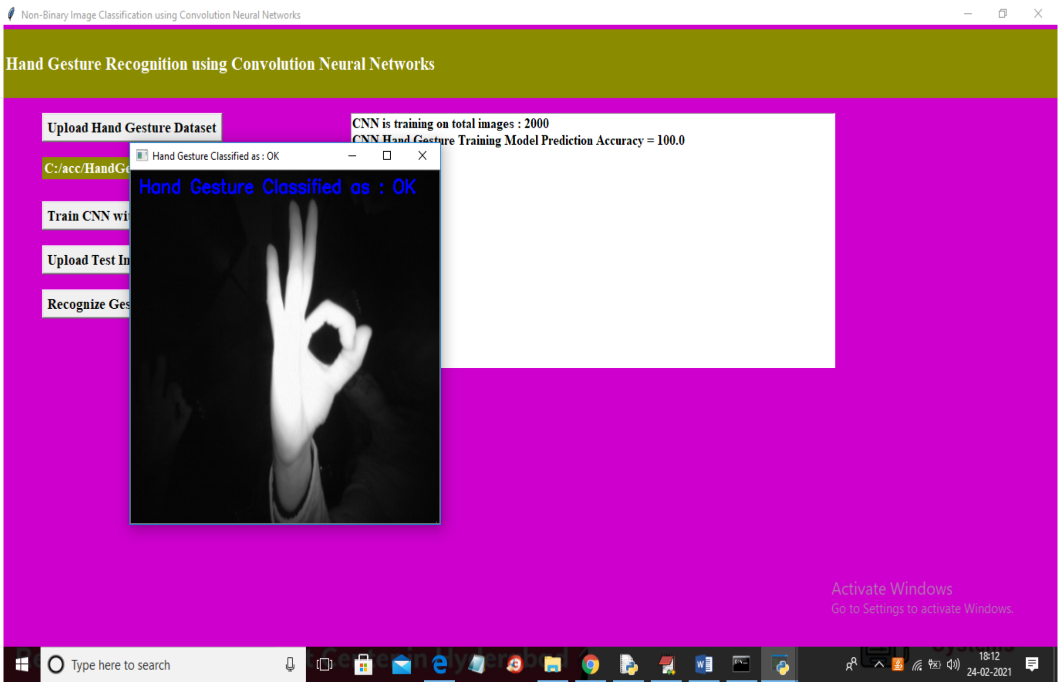
Upload Hand Gesture Dataset
C:/acc/HandGesture/Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video

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.


Non-Binary Image Classification using Convolution Neural Networks

```
s : 2000
Model Prediction Accuracy = 100.0
```

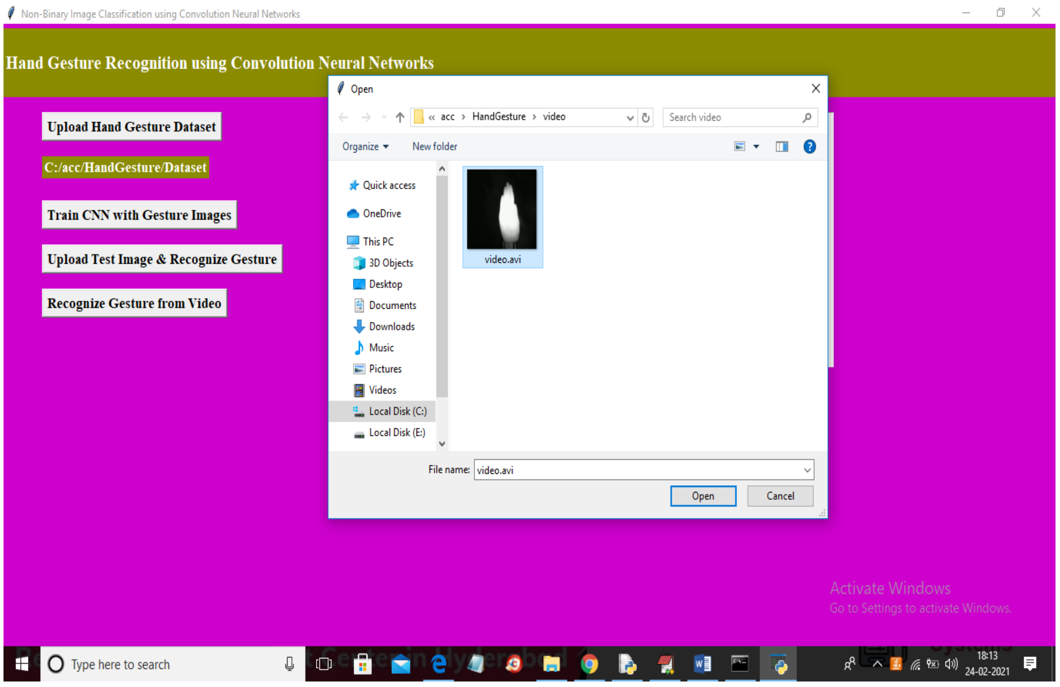
In above screen selecting and uploading '14.png' file and then click Open button to get below result.


Non-Binary Image Classification using Convolution Neural Networks
Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
C:/acc/HandG
Train CNN wi
Upload Test In
Recognize Ges
Hand Gesture Classified as : OK
Activate Windows
Go to Settings to activate Windows.
Type here to search
18:12
24-02-2021

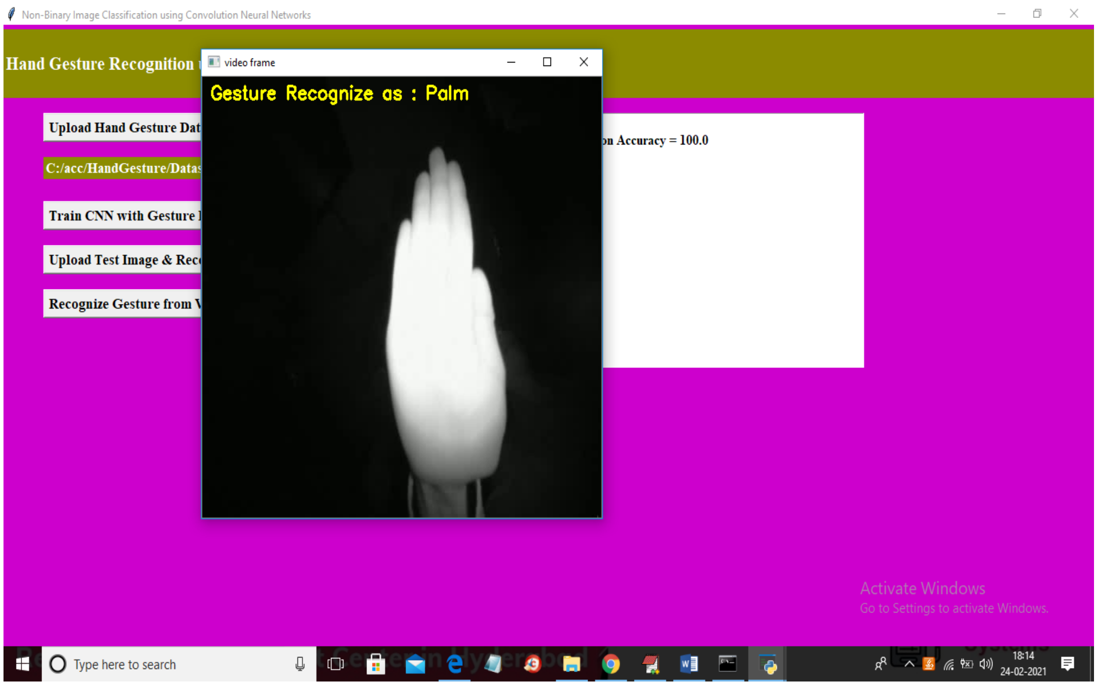
```
CNN is training on total images : 2000
CNN Hand Gesture Training Model Prediction Accuracy = 100.0
```

Hand Gesture Classified as : OK

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.


Non-Binary Image Classification using Convolution Neural Networks
Hand Gesture Recognition using Convolution Neural Networks
Upload Hand Gesture Dataset
C:/acc/HandGesture/Dataset
Train CNN with Gesture Images
Upload Test Image & Recognize Gesture
Recognize Gesture from Video
Open
« acc › HandGesture › video
Search video
Organize
New folder
Quick access
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (E:)
video.avi
File name: video.avi
Open
Cancel
Activate Windows
Go to Settings to activate Windows.
Type here to search
18:13
24-02-2021

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


Non-Binary Image Classification using Convolution Neural Networks

Hand Gesture Recognition

Upload Hand Gesture Dat

C:/acc/HandGesture/Datas

Train CNN with Gesture I

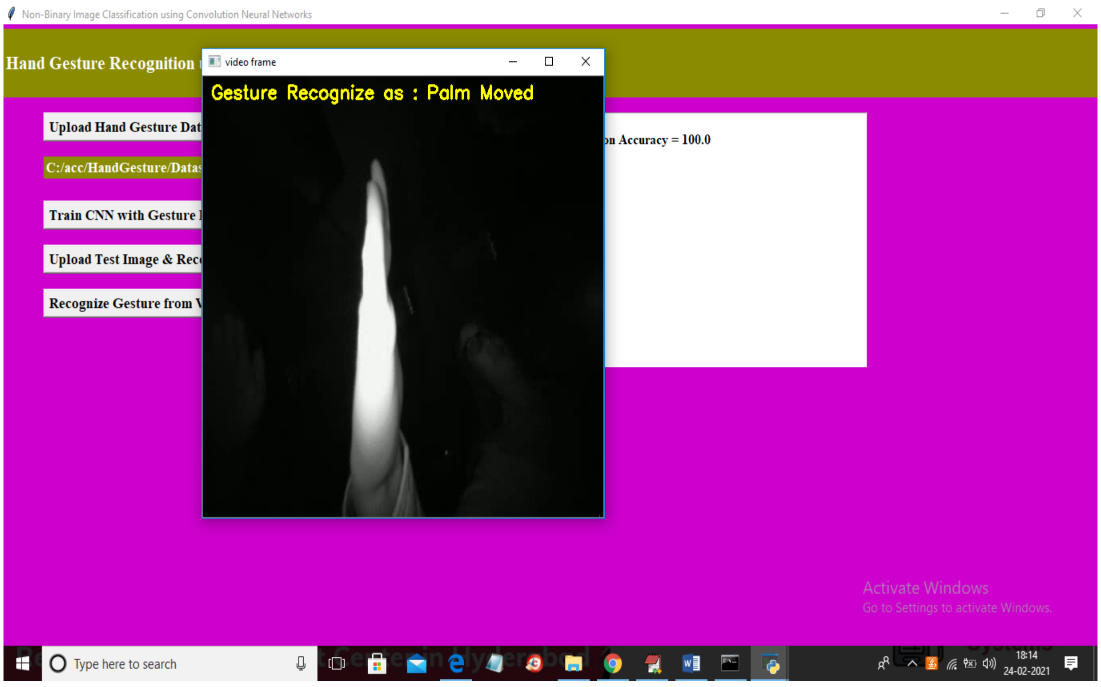
Upload Test Image & Rec

Recognize Gesture from V

video frame

Gesture Recognize as : Palm

on Accuracy = 100.0


Non-Binary Image Classification using Convolution Neural Networks

Hand Gesture Recognition

Upload Hand Gesture Dat

C:/acc/HandGesture/Datas

Train CNN with Gesture I

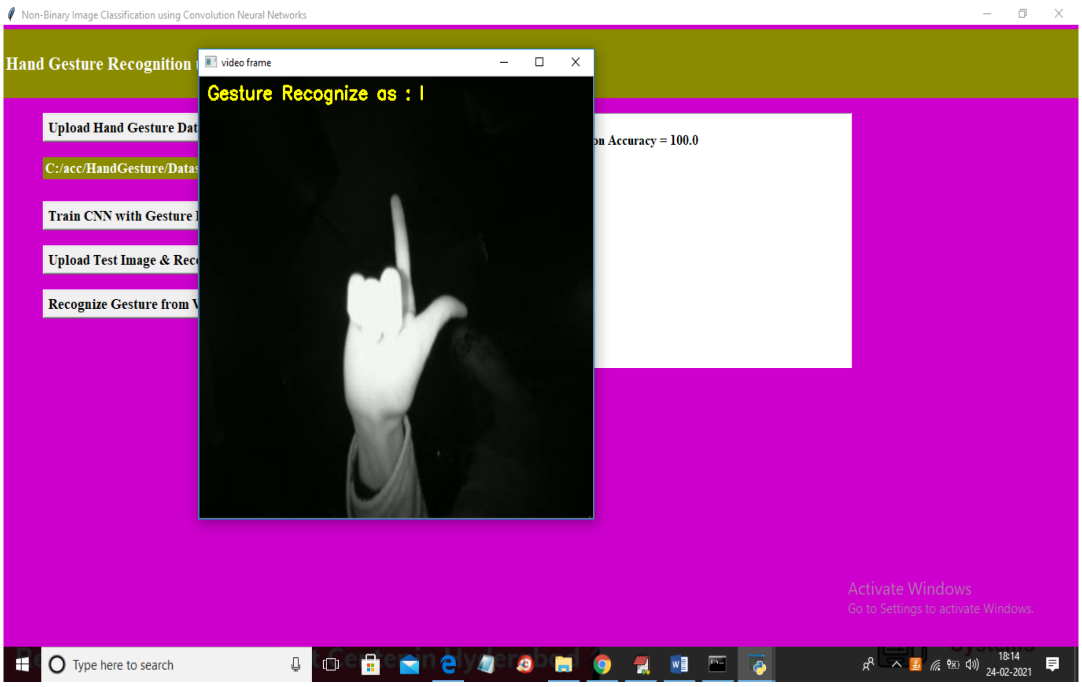
Upload Test Image & Rec

Recognize Gesture from V

video frame

Gesture Recognize as : Palm Moved

on Accuracy = 100.0

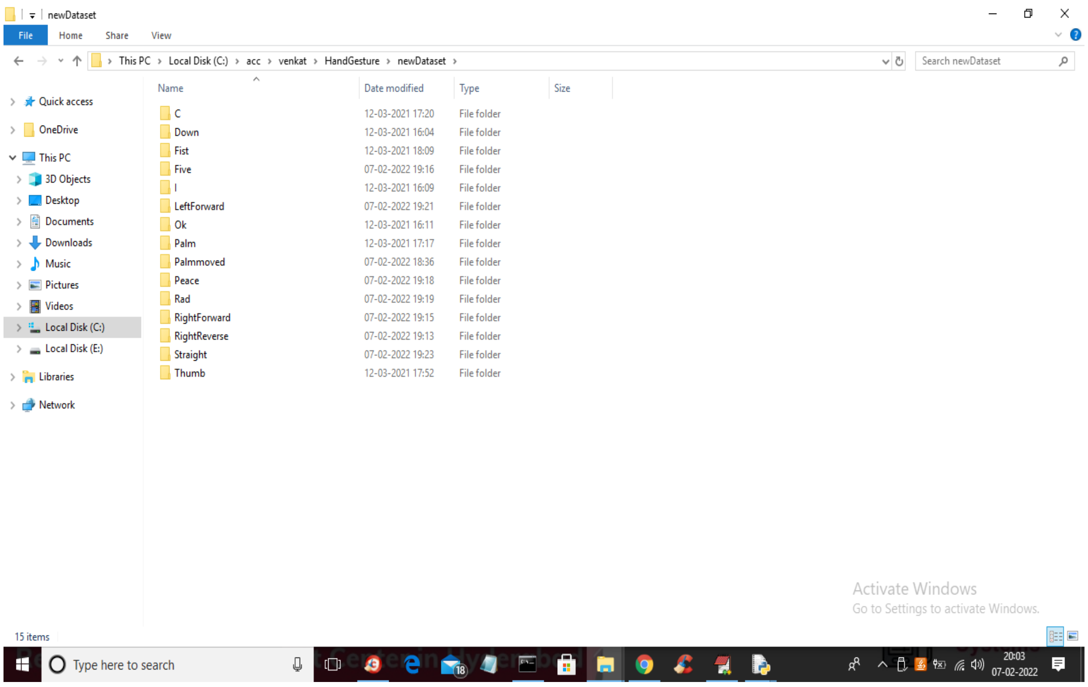


Gesture Recognize as : I

on Accuracy = 100.0

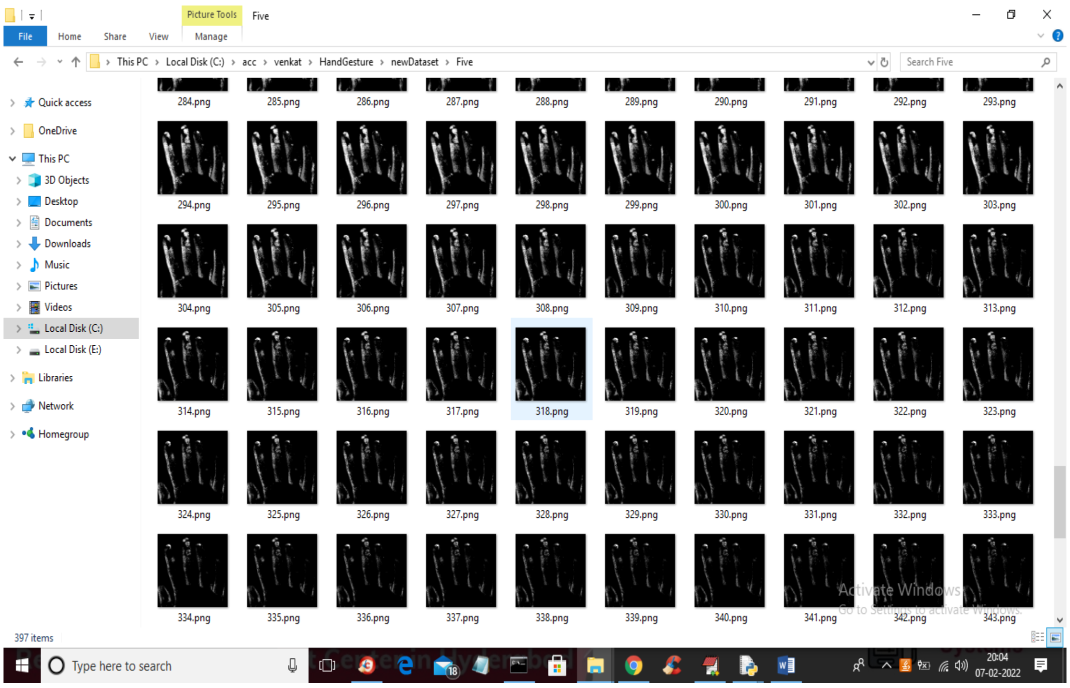
In above screen as video play then will get recognition result

Added new signs showing in below screen

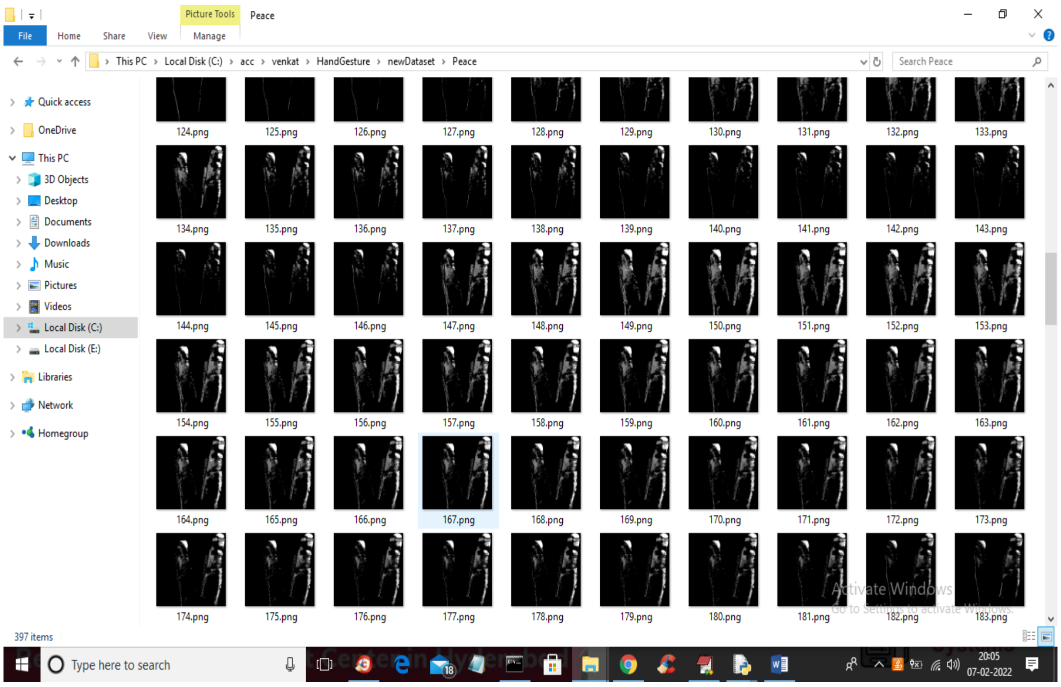


Name	Date modified	Type	Size
C	12-03-2021 17:20	File folder	
Down	12-03-2021 16:04	File folder	
Fist	12-03-2021 18:09	File folder	
Five	07-02-2022 19:16	File folder	
I	12-03-2021 16:09	File folder	
LeftForward	07-02-2022 19:21	File folder	
Ok	12-03-2021 16:11	File folder	
Palm	12-03-2021 17:17	File folder	
Palmmoved	07-02-2022 18:36	File folder	
Peace	07-02-2022 19:18	File folder	
Rad	07-02-2022 19:19	File folder	
RightForward	07-02-2022 19:15	File folder	
RightReverse	07-02-2022 19:13	File folder	
Straight	07-02-2022 19:23	File folder	
Thumb	12-03-2021 17:52	File folder	

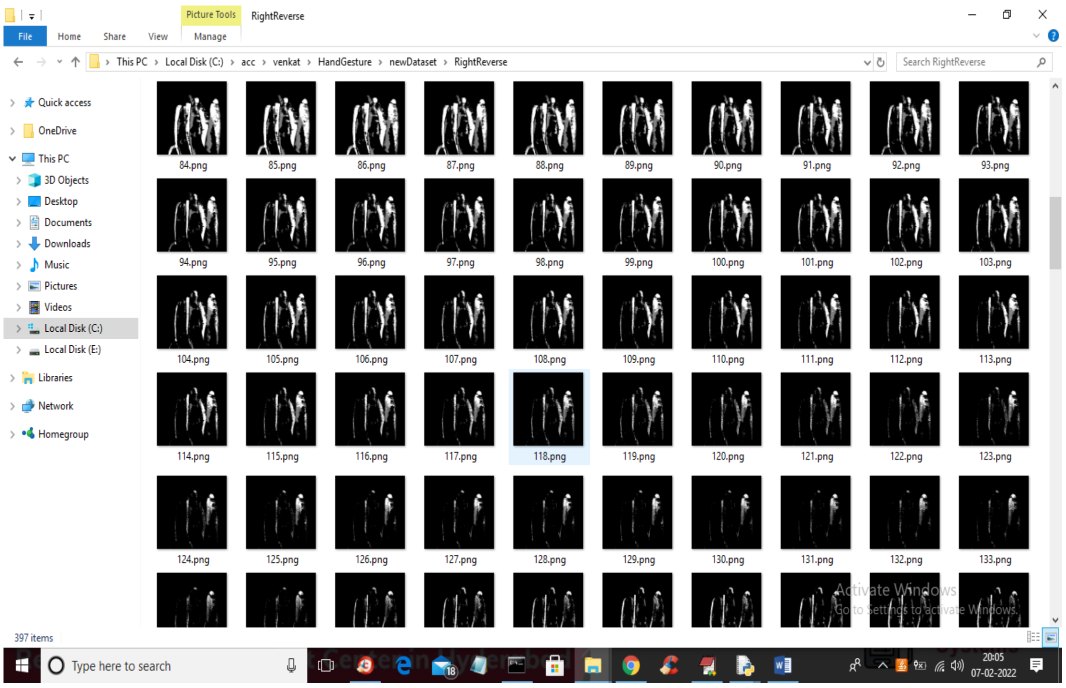
In above screen show all five fingers to get five recognition like below screen



Show two fingers to get peace sign like below screen

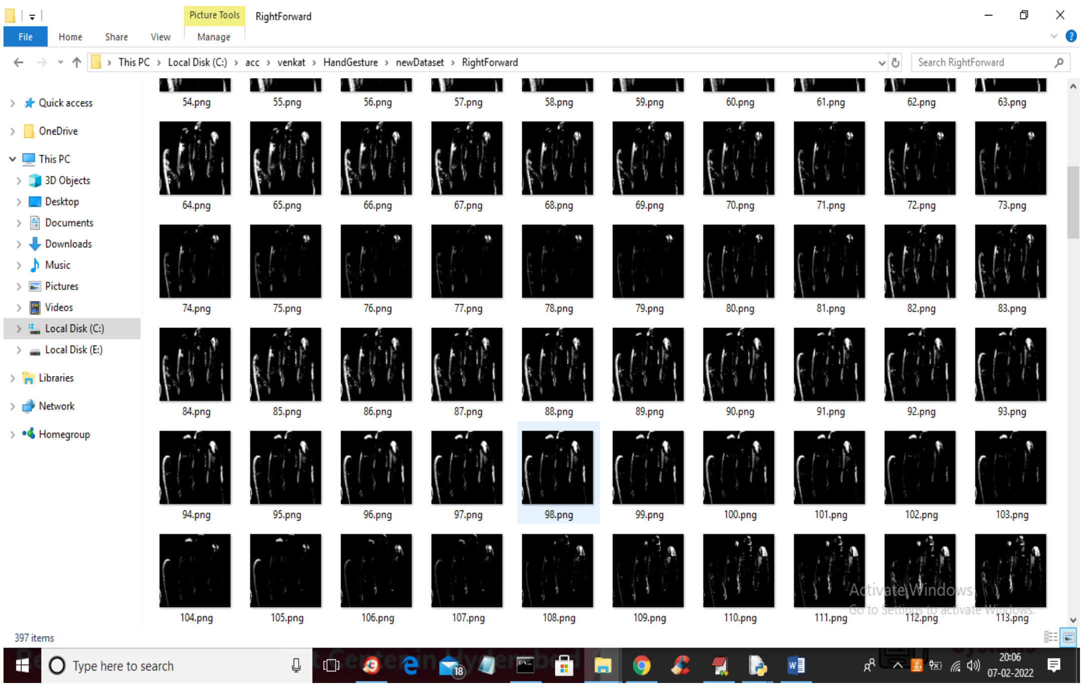


Revers hand two fingers should be shown and 3 fingers for "Right Reverse".



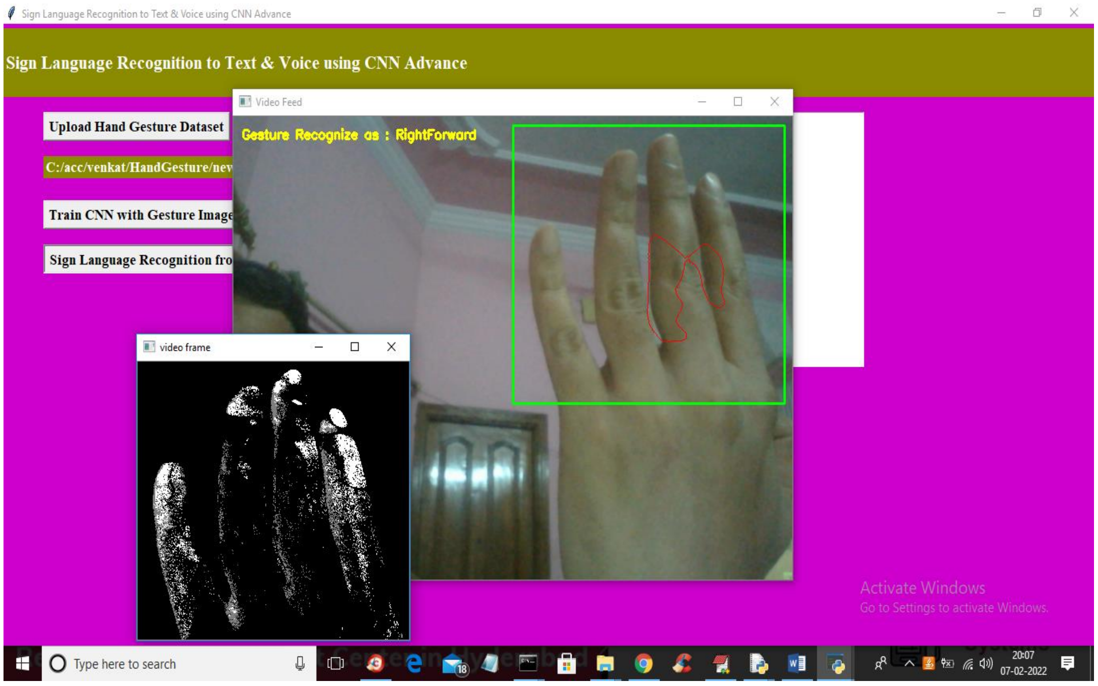
RightReverse
This PC › Local Disk (C:) › acc › venkat › HandGesture › newDataset › RightReverse
397 items

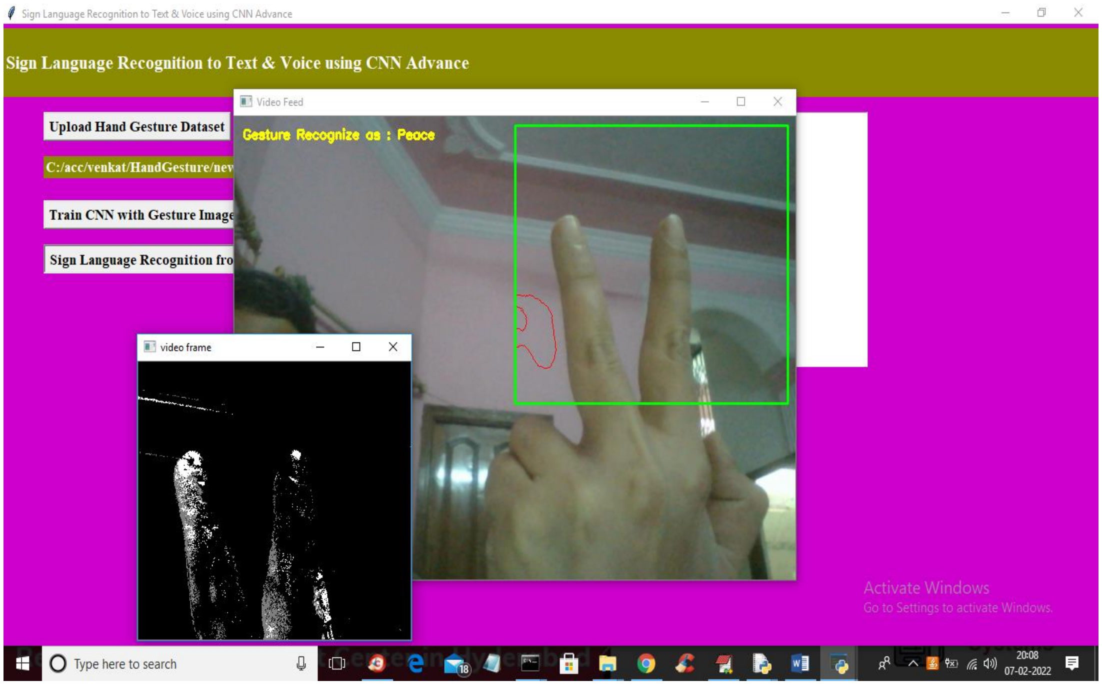
Show 4 fingers for “Right Forward” like below screen.

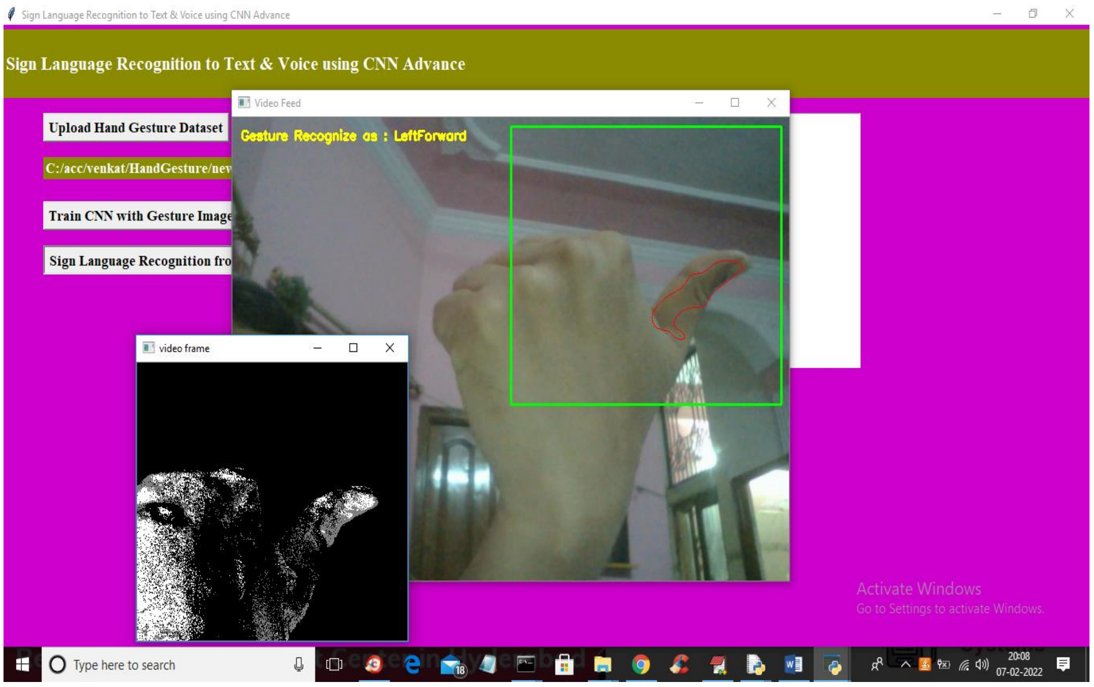


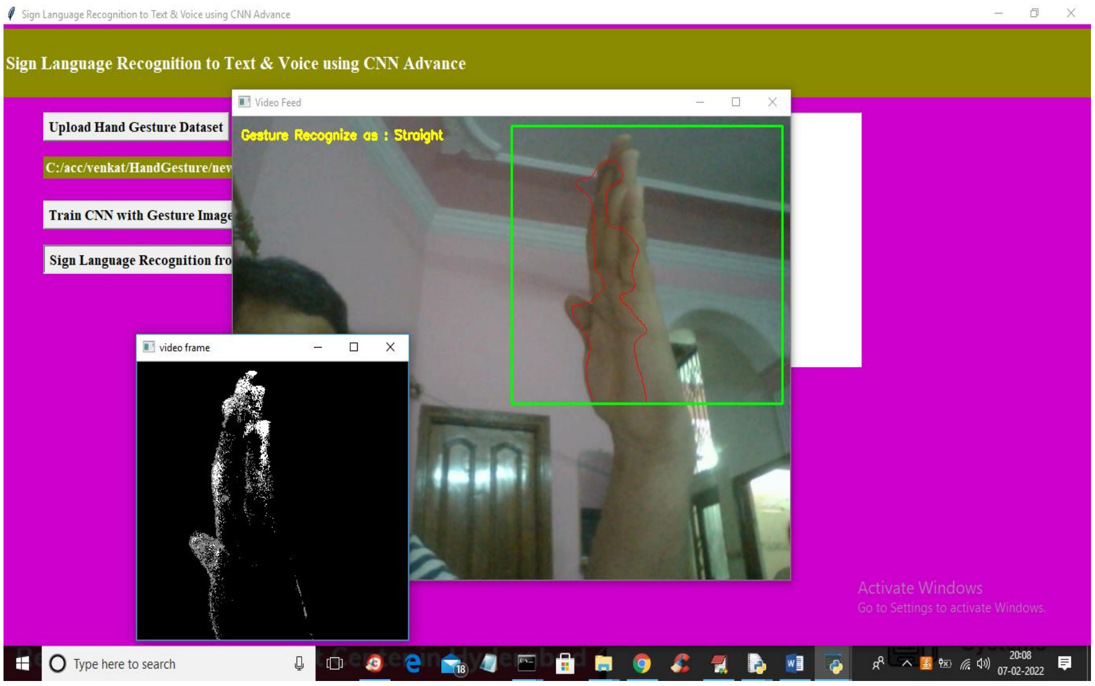
RightForward
This PC › Local Disk (C:) › acc › venkat › HandGesture › newDataset › RightForward
397 items

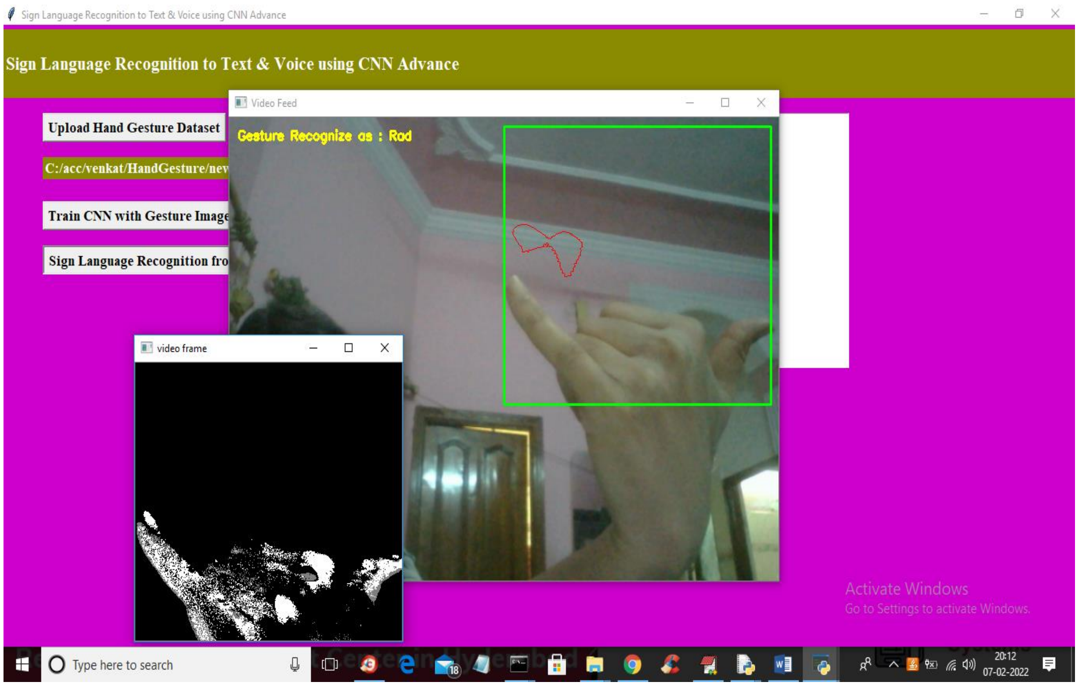
Similarly in new dataset folder whatever signs showing you can show those signs to camera to get output.

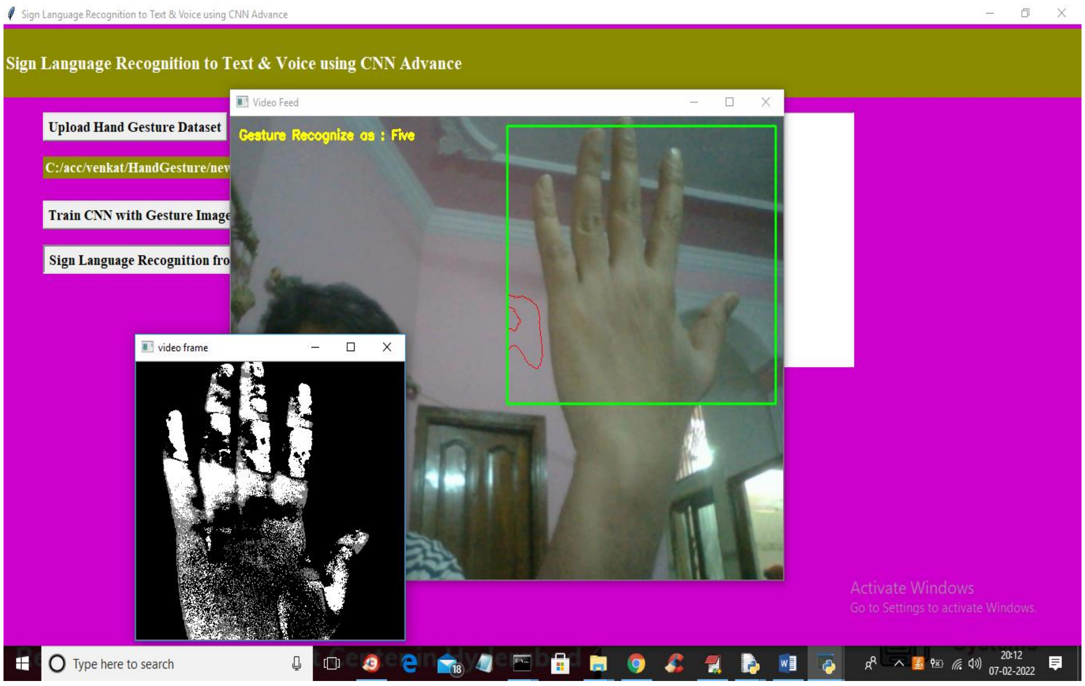

Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : RightForward
video frame


Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : Peace
video frame


Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : LeftForward
video frame


Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : Straight
video frame


Sign Language Recognition to Text & Voice using CNN Advance
Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : Rod
video frame
Type here to search


Sign Language Recognition to Text & Voice using CNN Advance
Sign Language Recognition to Text & Voice using CNN Advance
Upload Hand Gesture Dataset
C:/acc/venkat/HandGesture/nev
Train CNN with Gesture Image
Sign Language Recognition fro
Video Feed
Gesture Recognize as : Five
video frame
Type here to search

Chapter-10

Conclusion

Sign languages are very broad and differ from country to country in terms of gestures, body language and face expressions. The grammars and structure of a sentence also varies a lot. In our study, learning and capturing the gestures was quite a challenge for us since the movement of hands had to be precise and on point . Some gestures are difficult to reproduce. And it was hard to keep our hands in exact same position when creating our dataset.

References

[1] https://peda.net/id/08f8c4a8511

[2] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.

[3] CABRERA, MARIA & BOGADO, JUAN & Fermín, Leonardo & Acuña, Raul & RALEV, DIMITAR. (2012). GLOVE-BASED GESTURE RECOGNITION SYSTEM. 10.1142/9789814415958_0095.

[4] He, Siming. (2019). Research of a Sign Language Translation System Based on Deep Learning. 392-396. 10.1109/AIAM48774.2019.00083.

[5] International Conference on Trendz in Information Sciences and Computing (TISC). : 30-35, 2012.

[6] Herath, H.C.M. & W.A.L.V.Kumari, & Senevirathne, W.A.P.B & Dissanayake, Maheshi. (2013). IMAGE BASED SIGN LANGUAGE RECOGNITION SYSTEM FOR SINHALA SIGN LANGUAGE