CHAPTER 1

INTRODUCTION

Gestures are the actions that commonly originate from any bodily motion, but usually they can be originated from the face or the hand. These gestures are helpful in communicating meaningful messages by the user. A hand gesture recognition system can be developed in order to capture the hand gestures that are being performed by the user, analyse them further to control the system based on the information gained from the analysis.

The hand gesture recognition system for human computer interaction uses OpenCV for gesture recognition and CNN for training the model in use. It takes in gestures such as the palm for zooming in, the fist for zooming out, a swing of the hand turns the pages towards the right and the peace sign turns the pages towards the left.

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. It contains various modules like video analysis for motion estimation, object tracking and background subtraction, object detection for detecting pre-defined objects like cars or faces etc, HighGUI for an easy to use interface with simple UI capabilities and a calib3D module that includes algorithms for object elimination and elements of 3D reconstruction. In our project OpenCV is used to detect the hand in order to recognize the gesture made.

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition of faces etc., are some of the areas where CNNs are widely used. CNN image classifications take an input image, process it and classify it under certain categories. In our project CNN is used to train the model built.

1.1 Problem Statement

In order to control a system, here a PDF or book, the user has to manually interact with a hardware such as mouse or keyboard to press a button in order to perform the actions such as turning the page, zoom in or zoom out. This can be overcome using a Human Computer Interaction System that detects the user's gestures and controls them. In the literature survey, many implementations have certain strict constraints like having uniform background, having certain lightning conditions, wearing special gloves for making the gestures, etc. Our system doesn't include such constraints. It only requires a clear background that would not interfere in capturing and detecting the hand gesture.

1.2 Motivation

Communication plays an important role in today's life, whether it could be communication between two people or between a user and the system. For one such use, hand gestures would be a great help in facilitating intelligent human and computer interaction. Coming across the various aspects of life in today's world there are many situations where it is difficult to achieve a proper communication due to some reasons, our major motivation for this project came from one such need.

Among us, there are some people who do not have the voice to express, and sometimes, the capability to hear. The deaf and dumb sign language was developed to aid the communication for such people. Our idea was to create a device which could help them communicate, to give them a voice. Therefore, by making use of hand gestures we develop a hand gesture recognition system that helps in human computer interaction.

1.3 Objectives

The main objective of the system is to eliminate the drawbacks like wearing special gloves, having certain lighting conditions, having uniform background etc that are observed in the literature survey. The following points state the objectives of the system built in detail.

- Hand detection – OpenCv is used to detect the hand gesture made by the user.
- Training the model – Convolution Neural Networks are used to train the model so as to recognize the gesture made.
- Classifying the data – it is used to categorize the various gestures made. This is again achieved using CNN.

1.4 Existing System

A hand gesture recognition system was developed to capture the hand gestures being performed by the user and to control a computer system by that incoming information which facilitates the human computer interaction. Many of such systems in literature have strict constraints like wearing special gloves, having uniform background, long-sleeved user arm, being in certain lightning conditions, using specified camera parameters etc. Some of the limitations of the existing systems are that they ruin the naturalness of a hand gesture recognition system and also correct detection rates and the performances of those systems are not well enough to work on a real time HCI system.

1.5 Proposed System

The system aims to design a vision-based hand gesture recognition system with a high correct detection rate along with a high-performance criterion, which can work in a real time HCI system without having any of the mentioned strict limitations (gloves, uniform background etc) as in the literature survey, on the user environment. It is composed of a human computer interaction system which uses hand gestures as input for communication.

System is initiated with acquiring an image from a web-cam or a pre-recorded video sequence. Skin color is determined by an adaptive algorithm in the first few frames. Once the skin color is fixed for the current user, lightning and camera parameter conditions, hand is localized with a histogram clustering method. Then a hand gesture recognition algorithm is applied in consecutive frames to distinguish the current gesture. Finally, the gesture is used as an input for a computer application to facilitate human computer interaction. The proposed system can be defined using a flowchart that contains three main steps. Fig 1.1 shows this.

- Learning
- Detection
- Recognition

1. Learning

It involves two aspects such as

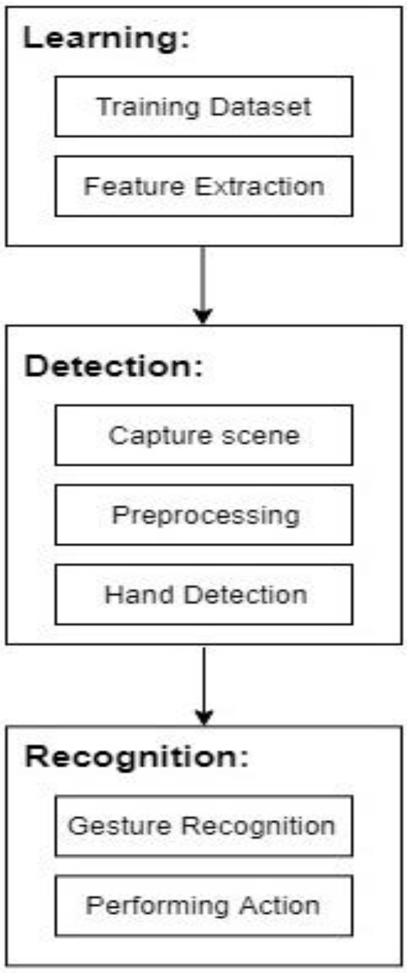
- Training dataset: This is the dataset that consists of different types of hand gestures that are used to train the system based on which the system performs the actions.

- Feature Extraction: It involves determining the centroid that divides the image into two halves at its geometric centre.

2. Detection

- Capture scene: Captures the images through a web camera, which is used as an input to the system.
- Pre-processing: Images that are captured through the webcam are compared with the dataset to recognize the valid hand movements that are needed to perform the required actions.
- Hand Detection: The requirements for hand detection involves the input image from the webcam should be fetched at a speed of 20 frames per second. The distance between hand and camera should be in the range of 30 to 100 cm. The video input is stored frame by frame into a matrix after pre-processing.

Fig 1.1: Flow chart of the proposed system



3.Recognition

- Gesture Recognition: The number of fingers present in the hand gesture is determined by making use of defect points present in the gesture. The resultant gesture obtained is fed through a 3Dimensional Convolutional Neural Network consecutively to recognize the current gesture.
- Performing action: The recognised gesture is used as an input to perform the actions required by the user.

CHAPTER 2

LITERATURE SURVEY

Many papers were reviewed for the literature survey. Every paper used a methodology that had its own advantages and disadvantages. Tabe 2.1 lists the various papers, the methodology used, positive and negative aspects of the methodology implemented.

Table 2.1: Literature Survey

SL. NO	AUTHOR, TITLE AND YEAR OF PUBLICATION	METHODOLOGY	ADVANTAGES	LIMITATIONS
1.	Meenakshi Panwar, Pawan Singh Mehra, **Hand Gesture Recognition for Human Computer Interaction.** **(**International Conference on Image Information Processing- 2011)	The implementation is divided into four main steps: 1.Image Enhancement and Segmentation 2.Orientation Detection 3.Feature Extraction 4.Classification	1.The proposed and implemented algorithm does not require any kind of training of sample data. 2.It takes a less computation time as compare to other approaches.	Its main limitation is that different light condition leads to change of colors very rapidly, which may cause error or even failures. For example, due to insufficient light condition, the existence of hand area is not detected but the non-skin regions are mistaken for the hand area because of same color.
2.	Rafiqul Zaman Khan and Noor Adnan Ibraheem, **Comparitive**	Three main steps for hand gesture recognition system: 1. Segmentation 2. Feature	1.Recognises gestures from both 2D and 3D images. 2.Different groups	1.The main drawback of the system is it does not consider gesture recognition of

	Study Of Hand Gesture Recognition System. (AIRCC Digital Library DOI:10.5121/csit. 2012.2320 - 2012)	Representation 3.Recognition Techniques Hand gesture recognition by modeling of the hand in spatial domain. Uses various 2D and 3D geometric and non-geometric models for modeling. It has used Fuzzy c-Means clustering algorithm which resulted in an accuracy of 85.83%.	of features are examined to decide the good performance group.	temporal space, i.e; motion of gestures. 2.It uses images to classify the gesture and is not in real time. 3.It is unable to classify images with complex background i.e; where there are other objects in the scene with the hand objects . 4.The computation time to recognize a gesture is high(2-4s). 5. Classification of same gesture in two different images with variation of hand position gives contradicting outputs.
3.	Arpita Ray Sarkar G. Sanyal , S. Majumder **Hand Gesture Recognition Systems: A Survey.** (International Journal of	Hand gesture recognition system has four different phases to find out the gesture. They are - 1.Data acquisition, 2.Hand segmentation and pre-processing, 3.Feature extraction and finally	This paper proves to be advantageous as it provides a brief description of all the gesture modelling algorithms that can be used to recognize a hand gesture. It displays	The paper presents some of the limitations regarding all the previously implemented systems: **1.Change in illumination:** When there is a change in the illumination condition, the system fails to recognize

	Computer Applications Volume 71– No.15, May 2013)	4.the recognition. 1.Suitable input device should be selected for the data acquisition. There are a number of input devices for data acquisition. Some of them are data gloves, marker, hand images (from webcam/ Kinect 3D sensor). 2.Gesture modeling has four different steps, viz. hand segmentation, filter/ noise removal, edge/ contour detection and lastly normalization. 3.Large number of features, such as, shape, textures, contour, motion, distance, centre of gravity etc. can be used for hand gesture recognition. 4.Hidden Markov Model (HMM) is a doubly stochastic model and appropriate for dealing with the stochastic properties	all the 4 phases of gesture recognition in a detailed and easy to understand manner. The various input and sensor devices available that help the system and the user to simplify the recognition process are mentioned in an appropriate manner.	properly **2.Rotation or orientation limitation:** An HGR system fails to recognize if the hand is orientated in a different angle **3.Scaling problem:** The problems of scaling arise due to different field of applications, hand size of the users, perspective. **4. Special hardware:** A number of special hardware, like Range camera , 3D depth sensor, Data gloves have been used.

		in gesture recognition.		
4.	Manjunath A E, Vijaya Kumar B P, Rajesh H **Comparative Study of Hand Gesture Recognition Algorithms.** (International Journal of Research in Computer and Communication Technology, Vol 3, Issue 4, April-2014)	This project is divided into 3 phases: - **1.Hand gesture recognition using kinect camera -** Kinect camera is a line motion input sensing device that is used to sense the hand gesture made by the user. **2.Algorithms for Hand Detection Recognition** **A.Edge Detection**: Using the boundary function in MATLAB, images are filled up and the boundary pixels are detected and are stored in a linear array. The fingertips are detected using segmentation algorithm or vectorization technique. **3.Hand Gesture recognition-**The system stores the orientation histogram	This system captures the gestured image and compares it with the predefined images in the database. On finding a match, it outputs the number which the gesture signifying. For example, if the gesture is just a forefinger, then the generated output is 1, etc.	The edge detection and segmentation algorithms used here are not very efficient when compared to neural networks. The dataset being considered here is very small and can be used to detect very few sign gestures. The histogram matching method is not very reliable and can give false output at times.

		corresponding to the input. It then tracks the histogram of the input to the histograms of the gestures stored in the database. If any of the histograms match, then the corresponding gesture will be shown.		
5.	Dnyanada R Jadhav, L. M. R. J Lobo, **Navigation of PowerPoint Using Hand Gestures.** (International Journal of Science and Research (IJSR) - 2015)	The System architecture consists of: 1.Image acquisition 2.Segmentation of hand region. 3.Distance transform method for gesture recognition.	1.Distance transform method is better in performance than circular profiling method. 2. The presented gesture recognition system recognizes both static and dynamic gestures.	The limitations are: 1.The number of gestures that are recognized are less. 2.The gestures where not used to control any applications.
6.	Ruchi Manish Gurav Premanand K. Kadbe, **Real time Finger Tracking and Contour**	There are three main algorithms that are used: 1.Viola and jones Algorithm. 2.Convex Hull Algorithm.	The work was accomplished by training a set of feature set which is local contour sequence. The main	The limitations of this system are that it requires two sets of images for classification. One is the positive set that contains the

	Detection for Gesture Recognition using OpenCV. **(**International Conference on Industrial Instrumentation and Control (ICIC) – 2015)	3. The AdaBoost based learning Algorithm.	advantage of Local contour sequence is that it is invariant to rotation, translation and scaling so it is a good feature to train the learning machine. We have achieved 92% accuracy with Convex Hull and 70% accuracy with AdaBoost.	required images. The other is the negative set that contains contradicting images.
7.	Pei Xu, **A Real-Time Hand Gesture Recognition and Human Computer Interaction System.** (Arxiv Journal Repository-Cornell University Library – April 2017)	The system consists of three components: 1.Hand detection 2.Gesture recognition 3. Human-Computer Interaction (HCI) It has implemented the following methodology: 1.The input image is preprocessed and the hand detector tries to filter out the hand from the input image 2.A CNN classifier is employed to recognize gestures	1.The system uses Convolutional Neural Networks (CNN) which reach an accuracy rate of 99% rather than other approaches such as Hidden Markov Model (HMM), Orientation Histogram which are less accurate. 2.It uses only one monocular camera to capture the image.	1.The system recognizes only static images. 2.The CNN used is not robust and reliable since the number of images and used to train and test the classifier is less. 3.It only recognizes a gesture and not a motion of gestures to control the mouse actions

		from the processed image, while a Kalman Filter is used to estimate the position of the mouse cursor. 3.The recognition and estimation results are submitted to a control centre which decides the action to be taken.		(Gesture one for dragging the mouse, Gesture two for clicking the mouse, etc).
8.	P.Suganya, R.Sathya, K.Vijayalakshmi **Detection and Recognition of Hand Gestures To Control The System Applications By Neural Networks.** (International Journal of Pure and Applied Mathematics, Volume 118 No. 10-2018)	This project focuses on detection of hand gestures using java and neural networks. It is divided into two phases: - 1. Detection module using java where in the hand is detected using background subtraction and conversion of video feed into HSB video feed thus detecting skin pixels, 2. The second module is the prediction module; a convolutional neural network is used. The	1. Optimised Recognition of Gestures Aided by Neural Networks (ORGAN) is used to control the mouse actions such as mouse press, mouse release, cursor movement etc. using hand gestures. 2. ORGAN had proved to predict hand gestures using an optimized detection and prediction mechanism. It had shown its	There a few limitations regarding this project: 1. Socket programming is required in order to connect the java and python modules. 2. This socket programming is unreliable sometimes and requires constant internet connection. 3. The training dataset used is small. 4. In order to control the cursor movement robot class in java is being used and that again increases the

		neural network is written with the Theano library. A NumPy array is used to collect the dataset images. The input feed image is gained from Java. The input image is fed into the neural network and is analysed with respect to the dataset images.	effectiveness without the use of any specialized hardware for boosting its performance and efficiency.	complexity of the system. 5. The number of gestures that are trained and classified are less since the dataset used is small.

This literature survey throws light on the various techniques that can be used to implement Hand gesture recognition. Each method used has a few limitations that were overcome in the methodologies that were proposed later on. All the methodologies implemented were successful in the main task of recognizing the gesture made by the user.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Software Requirement Specification

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.

The SRS fully minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. The common understanding between the user and developer is captured in requirements document.

The SRS discusses the product but not the project that developed it; hence the SRS serves as basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued product evaluation.

SRS includes the functional and non-functional requirements for the model to be developed, they are as follows:

- **FUNCTIONAL REQUIREMENTS:**

Functional requirements are the necessary requirements for functioning of a software or a system. Functional requirements required for this system are –

1. Hand Detection
2. Background Subtraction
3. Hand Segmentation
4. Gesture Recognition and Prediction
5. Performing actions according to the recognised gesture on the pdf.

• NON – FUNCTIONAL REQUIREMENTS:

Non – Functional requirements support the functioning of the system. They have a check on the entire system as a whole. Non – functional requirements of the system are:

1. **Reliability:** How reliable is the system to deliver the system its service.
2. **Safety:** The ability of the system to perform without failure.
3. **Availability:** The ability of the system to render its services at all times whenever requested by the user.
4. **Dependability:** This is a property of a software / system that establishes a confidence about the system to the user.
5. **Security:** The system is secure such that no outside attacker can intercept the system.

• HARDWARE REQUIREMENTS:

1. **Web Camera:** 10 Megapixels or higher
2. **Processor:** Intel Core i5 or higher
3. **Graphical Processing Unit (GPU):** 4 GB
4. **Memory:** 25 GB Disk Space

• SOFTWARE REQUIREMENTS:

1. **Programming Language :** Python 3.6
2. **Operating System :** Windows / Linux / Mac OS.
3. **Anaconda :** Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.
4. **Integrated Development Environment (IDE) :** Spyder 3.3.4
5. **OpenCV :** A real time computer vision library.
6. **NumPy :** Python package for scientific computing.
7. **Scikit - learn**: Python library for machine learning.
8. **TensorFlow :** is a free and open-source software machine learning library.
9. **TFlearn** : is a modular and transparent deep learning library built on top of Tensorflow.
10. **PyAutoGUI :** PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

CHAPTER 4

SYSTEM ANALYSIS

4.1 Overview

System analysis is the process of observing systems for troubleshooting or development purposes. It is applied to information technology, where computer-based systems require defined analysis according to their mark-up and design.

The terms analysis and synthesis a term from Greek, meaning to take apart and to put together, respectively. These terms are used in many scientific disciplines, from mathematics and logic to economics and psychology, to denote similar investigative procedures. Analysis is defined as the procedure by which we break down an intellectual or substantial whole into parts, while synthesis means the procedure by which we combine separate elements or components in order to form a coherent whole. System Analysis researches apply methodology to the system involved, forming an overall picture. System analysis is used in every field where something is developed. Analysis can also be a series of components that perform organic functions together, such as system engineering. System engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed. It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

In IT, systems analysis can include looking at end-user implementation of a software package or product; looking in-depth at source code to define the methodologies used in building the system; or taking feasibility studies and other types of research to support the use and production of a software product, among other things.

Systems analysis professionals are often called upon to look critically at systems, and redesign or recommend changes as necessary. Inside and outside of the business world, systems analysts help to evaluate whether a system is viable or efficient within the context of its overall architecture and help to uncover the options available to the employing business or other party.

System analysts are different than systems administrators, who maintain systems day to day, and their roles generally involve a top-level view of a system to determine its overall effectiveness according to its design.

The development of an accident detection and alert system includes a system analysis phase. This helps produce the data model, a precursor to creating or enhancing accident detection. The system analysis would constitute the following steps:

- The development of a feasibility study: determining whether a project is economically, socially, technologically and organizationally feasible.
- Fact: finding measures, designed to ascertain the requirements of the system's end-users (typically involving visual observations of work on the existing system).
- Gauging how the end-users would operate the system (in terms of general experience) what the system would be used for and so on.

4.2 Mapping of Requirements:

The functional requirements of this system are given in Table 4.1. They state the requirements of each module that is built.

Table 4.1 Functional Requirements

SL. NO	Actors	Requirements	Module Mapping
1.	Hand Detector	• An input image from the web camera is fetched at the rate of 20 frames per second. • The distance between the hand and web-camera must be within the range of 30-100cm.	• The TensorFlow object detector along with OpenCV detects the hand by creating a boundary around it.
2.	Motion and Gesture Detector	• A processed matrix that consists of the movement of the detected hand. • Collection of frames of the desired gesture is captured and given as input to the 3D CNN.	• Recognition of the hand movement (left or right) and the gesture performed (zooming in and zooming out). • It interacts with the dataset for verification.
3.	Communication and Interaction `	• The detected hand movement or the gesture from the trained CNN.	• Performs the required action on the PDF/PPT through PyAutoGUI or System Calls.

The Non-functional requirements such as safety, efficiency and accuracy of the system built are stated in Table 4.2

Table 4.2 Non Functional Requirements

SL. NO	Requirement	Description
1.	Reliability and Safety	• The services provided by the system must be reliable. • The system must perform without any failures.
2.	Accuracy	• The system must be able the detect only the hand in a given environment. • Must not detect any other object of the same colour. • The system must perform the correct action for the given gesture.
3.	Efficiency	• Expecting an efficiency of 80% and above in all backgrounds.
4.	Operability	• It works on any Operating System – Windows or Linux, that has a web-camera of 5MP or higher.
5.	Usage	• Repeated usage will produce better results as it trains the 3D CNN for motion detection.

4.3 Functionalities of the Sub-System:

The system is mainly divided into the following subsystems – Hand and Motion detection, Datasets and 3D Convolution Neural Network.

Hand and Motion Detection:

- The Web-camera captures the hand movement and provides it as input to OpenCV and TensorFlow Object detector.
- Edge detection and skin detection are performed to obtain the boundary of the hand.
- This is then sent to the 3D CNN.

Dataset:

- It is used for training the 3D CNN.
- Two types of datasets are being used – one for the hand detection and the other for the motion or gesture detection
- Hand detection uses EGO dataset.
- Motion or Gesture Recognition uses Jester dataset.

3D CNN:

- CNN's are a class of deep learning neural networks used for analysing videos and images. It consists of several layers – input layer, hidden layers and output layer.
- It performs back propagation for better accuracy and efficiency.
- It performs training and verification of the recognised gestures and human computer interactions take place – turning of the pages, zooming in and zooming out.
- The interactions with the computer take place with the help of PyAutoGUI or System Calls.

CHAPTER 5

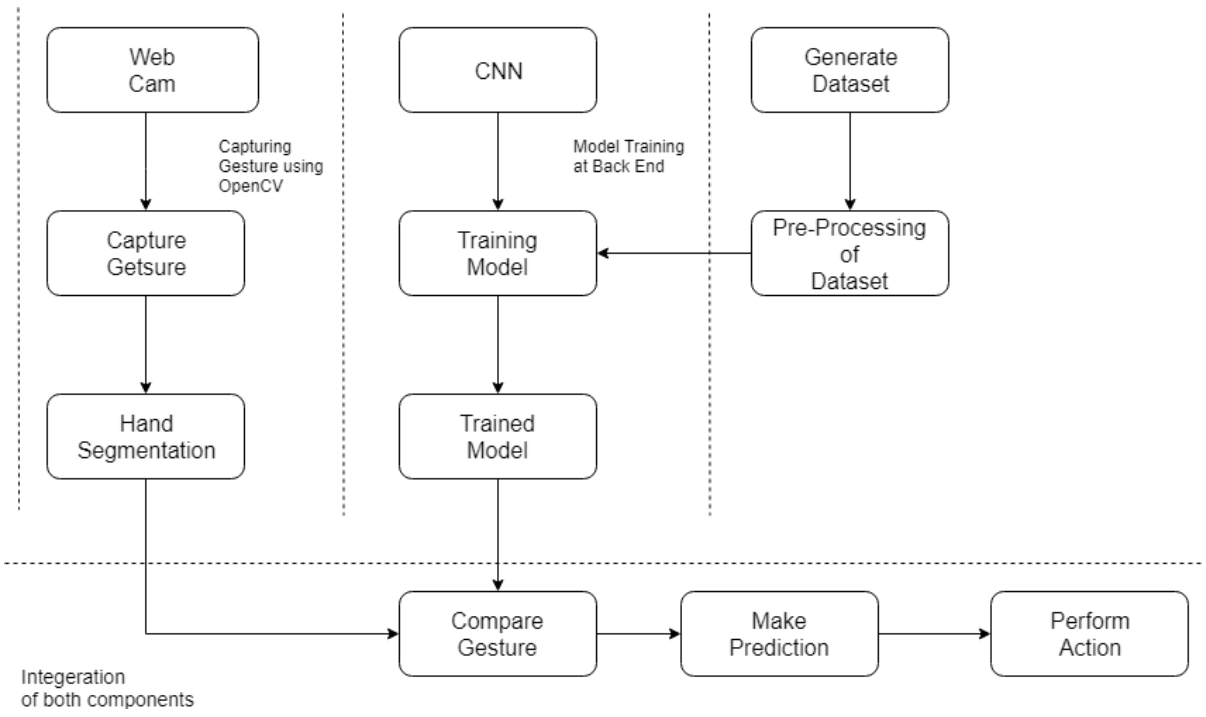
SYSTEM DESIGN

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of the systems analysis, systems architecture and system engineering. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

5.1 Architectural Design

The architectural design is shown in Fig 5.1.1

Figure 5.1.1: Architectural Design



There are 2 essential components involved in this project: the OpenCV part to detect the hand gesture and the neural network(CNN) to train the model. Before training the model, the dataset should be pre-processed by carrying out actions such as background subtraction and resizing. Finally the hand is segmented from the captured gesture and the prediction is made using the trained model. Once the prediction is made, the necessary action is performed at real time.

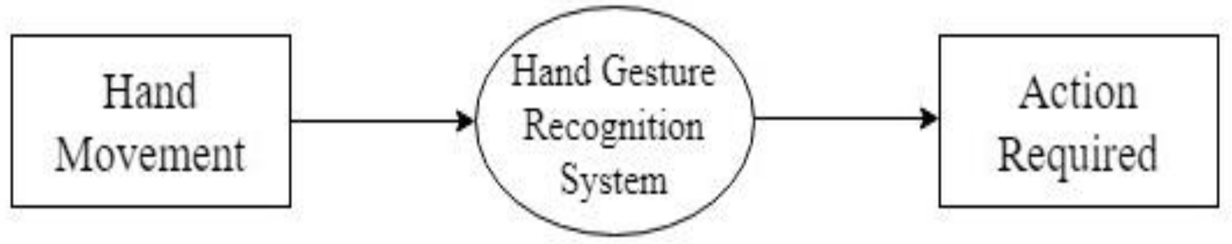
5.2 Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. DFD's can also be used for the visualization of data processing (structure design). In a DFD, data flows from an external source or an internal data source or an external data sink via an internal process. DFD provides no information about the timing of processes or whether the processes will operate in sequence or in parallel. The various levels of DFD's are shown in Figures (numbers).

LEVEL 0 DFD

The Level 0 DFD is a basic diagram that shows main the system in the model developed, the inputs and outputs that are required and expected.

Figure 5.2.1: LEVEL 0 DFD



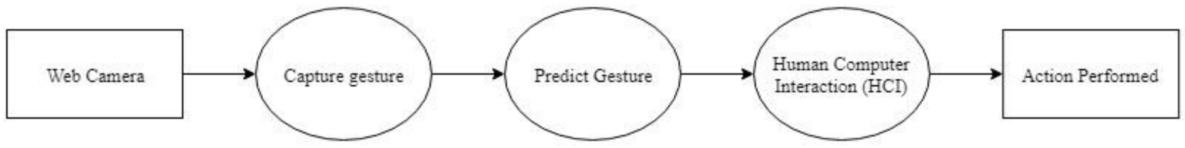
The DFD shows the system used – Hand Gesture Recognition System, the input – Hand movement and the output – Action performed (scrolling up, scrolling down, zooming in, zooming out, rotating 90 and 180 degrees).

LEVEL 1 DFD

The level 1 DFD shows the various modules in the system developed. There are two main modules in our system – Hand detector and the Human computer interaction. The Hand detector is used to obtain a real time hand gesture. The Human Computer Interaction is used to generate commands that will control the PDF or Book used.

Figure 4.3 Shows the Level 1 Data Flow diagram of the Hand Gesture Recognition and Human Computer Interaction System developed.

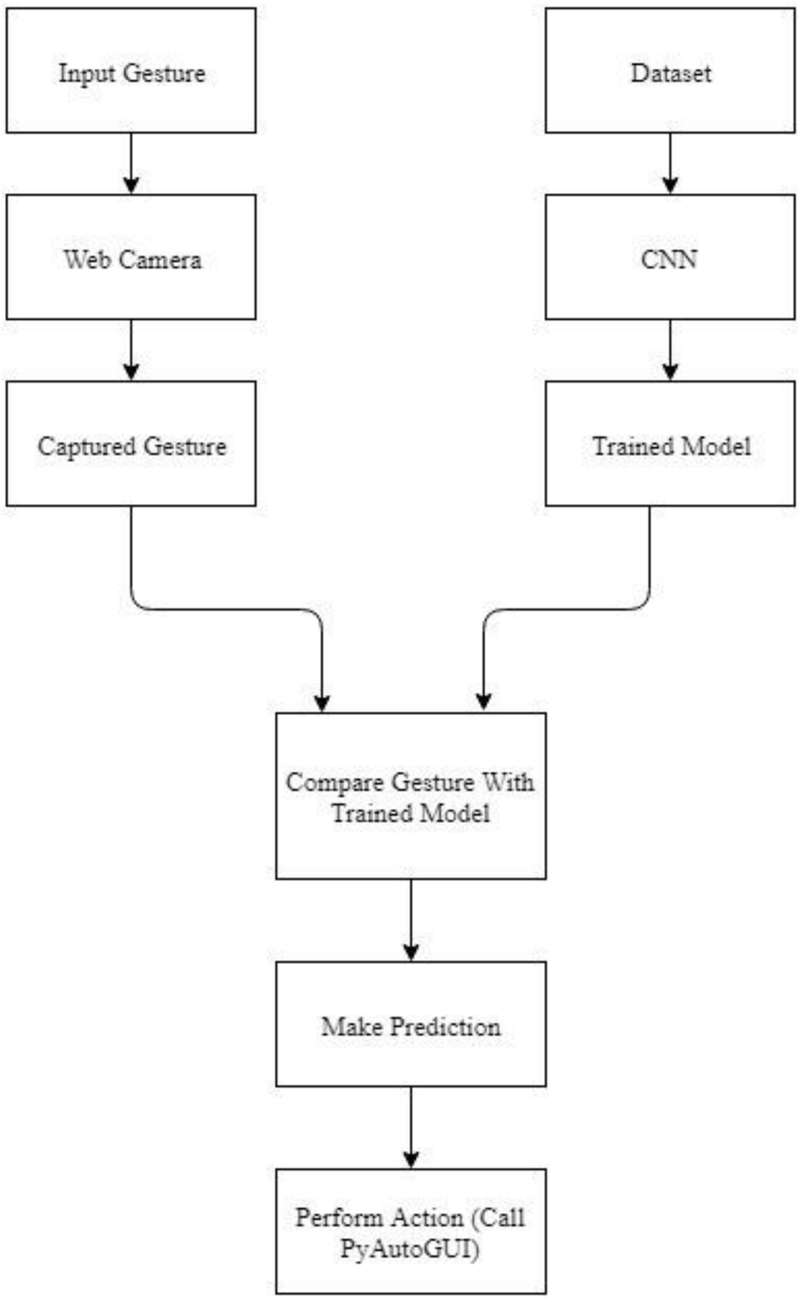
Figure 5.2.2: LEVEL 1 DFD



5.3 Overall Process

Fig. 5.3 demonstrates the flow of modules in the project. The captured gesture and the trained model are compared and the prediction is made. This prediction is used for performing actions at real time.

Fig 5.3 Flow Diagram



5.4 Behavioural Design

A Sequence Diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate, in which order and when. It is similar to a Message Sequence Chart. Sequence Diagrams are sometimes known as event diagram, event scenarios and timing diagrams. The Sequence Diagram for the system is as shown.

Figure 5.4: Sequence Diagram

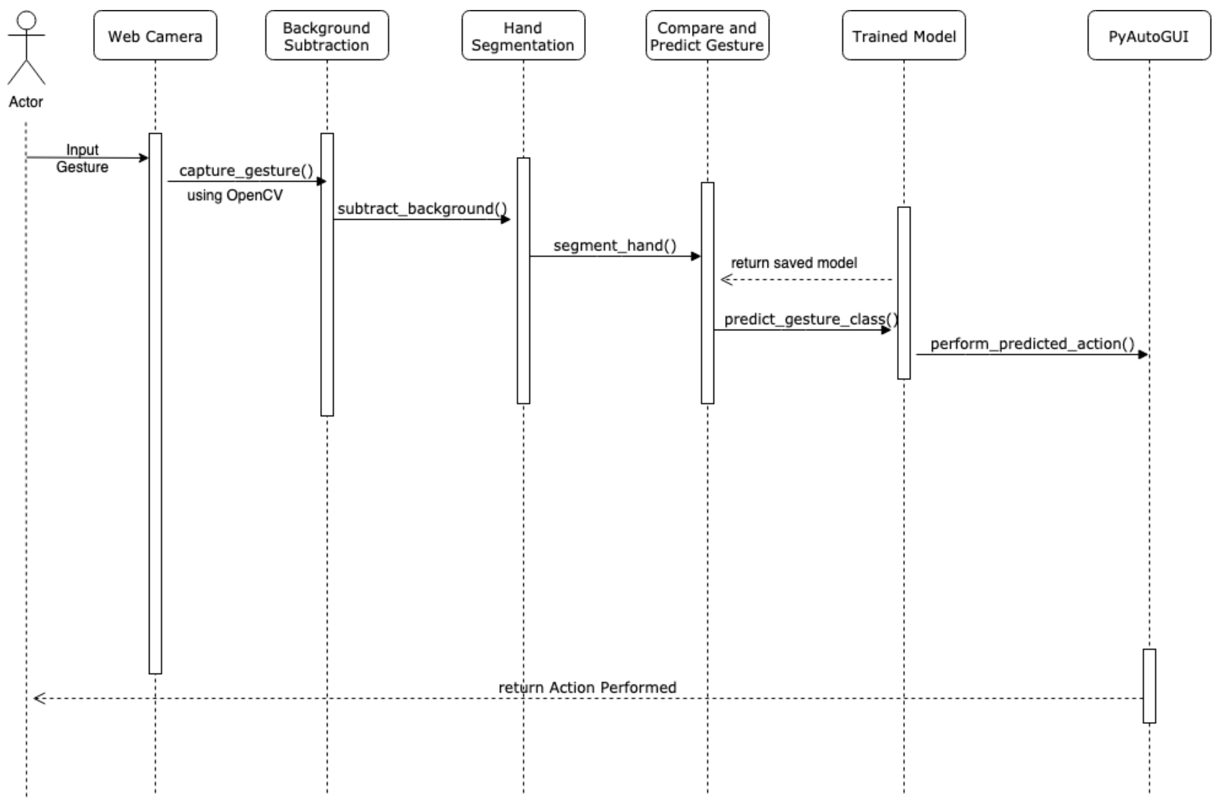


Figure 5.4 shows the behavioural design of the system. The actor (the user) performs some action which will then be detected by OpenCV. Then background subtraction is applied to the captured frame and from that the hand is segmented. The detected hand gesture is then compare with the trained model which is generated on running the model trainer. The prediction is made and finally the action is performed by integrating PyAutoGUI.

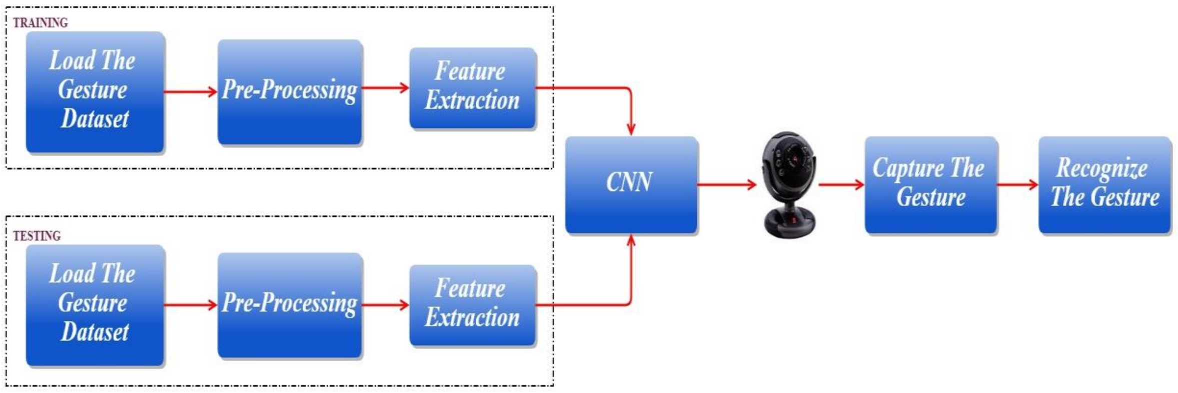
CHAPTER 6

IMPLEMENTATION

The implementation phase of any project development is the most important phase as it yields the final solution, which solves the problem at hand. The implementation phase involves the actual materialization of the ideas, which are expressed in the analysis document and developed in the design phase.

The flow of the implementation is depicted in the figure below:

Figure 6.1: Flow of Modules



1. Loading the training and testing dataset:
   The dataset is loaded and trained to create a model file. This model file will be used at run time to predict the gesture being shown.
2. Convolutional Neural Network:
   This is the neural network into which the dataset is loaded into. After running for a certain number of iterations, the model will be trained for the hand gesture datasets.
3. OpenCV:
   Using a webcam, the gesture will be captured. A certain number of pre-processing actions like background subtraction will be applied to the frame and then considered for prediction of gesture.

6.1 Integrated Development Environment (IDE)

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda*. The Anaconda distribution is used by over 13 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPyand Cython, as well as other open source software. It is released under the MIT license.

Some of the features of Spyder are:

- An editor with syntax highlighting, introspection, code completion.
- Support for multiple IPython consoles.
- The ability to explore and edit variables from a GUI.
- A Help pane able to retrieve and render rich text documentation on functions, classes and methods automatically or on-demand.
- A debugger linked to IPdb, for step-by-step execution.

6.2 Programming Language, Tools and Data Structures

Python 3.6

Python is an interpreted, high-level, general purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python is commonly used in artificial intelligence projects with the help of various libraries. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

Tools:

OpenCV: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

NumPy: is the fundamental package for scientific computing in Python. It provides routines for fast operations on arrays, including mathematical, logical, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Scikit-Learn: Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language.It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

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.

TFLearn : TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

PyautoGUI: PyAutoGUI is cross-platform GUI automation module that works on Python 2 and 3. You can control the mouse and keyboard as well as perform basic image recognition to automate tasks on your computer.

Data Structures:

NumPy arrays: NumPy arrays have a fixed size at creation. Changing the size of an array will create a new array and delete the original. It facilitates advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.

Python dictionaries: in Python is an unordered collection of data values, used to store data values as a key value pair.

Python lists: is used to store multiple data at the same time. The list in Python are ordered and have a definite count. The elements in a list are indexed.

6.3 Algorithm

- The CNN is designed using tensorflow.
- The network contains **1** input layer, **7** hidden convolution layers with **Relu** as the activation function and **1** Fully connected layer.
- The model is trained using this network for four gestures.
- The gesture is captured using a webcam and is pre-processed.
- This gesture is compared to the trained model and a prediction is made.
- This prediction is used to perform actions using PyAutoGUI.

6.4 Functions

1. **resizeImage():**
   This function s used in order to resize the images so that it can be fed into the Convolution Neural Network designed using tensorflow. The network accepts 89 x 100 dimensional image.
2. **run_avg():**
   This function is used to initialize the background. The weighted average is computed, accumulated and the background is updated.
3. **segment():**

The absolute difference between the background and current frame is computed. This difference is used to calculate the threshold in order to get the foreground. The contours in this thresholded image are computed and the contour area is calculated. Based on the contour area, the maximum contour area is found and this is recognized as the hand.

4. **getPredictedClass():**

   This function is used to make the prediction based on the gesture captured in the frame. The prediction is made by using the model that is loaded from the ModelTrainer file. The predicted class and the confidence are returned as a tuple.

5. **showStatistics():**

   This function is used to display a window to show the predicted gesture along with the confidence with which the gesture is predicted as a percentage.

6. **main():**

   This is the main function which calls the other functions and runs the program.

6.5 Walkthrough of the project

i. **Generation of dataset:**

   The dataset is generated by running the PalmTrack.py file. On running this file, a window is opened for capturing the gesture being shown. The background is subtracted and the Region Of Interest (ROI) is converted to grayscale and blurred.

   On pressing ‘s’, the window starts capturing the images of the gesture being shown. For each gesture there are two folders being created i.e training-set and test-set. A set of 1000 images are captured for the training-set and a set of 100 images are captured for the test-set. The following images give a brief idea of how the dataset look like:

Fig 6.5.1: Fist Dataset

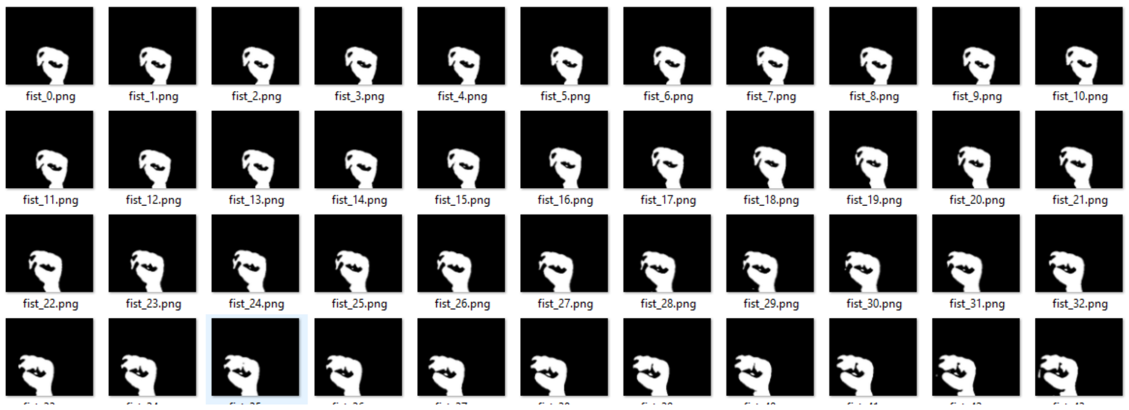


Fig 6.5.2: Palm Dataset

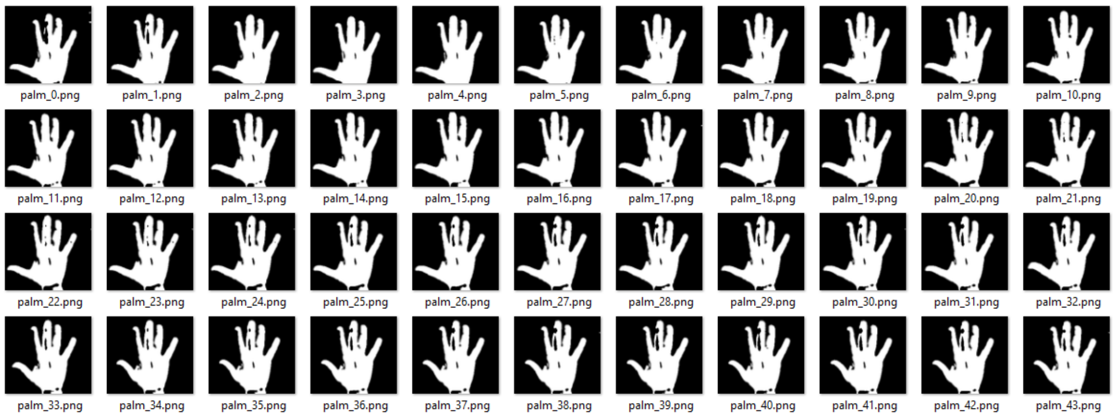


Fig 6.5.3: Swing Dataset

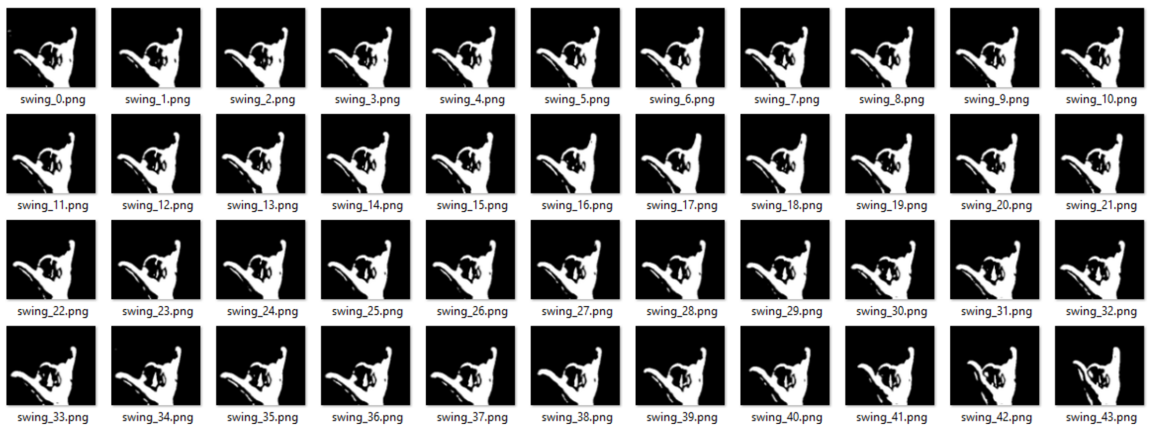
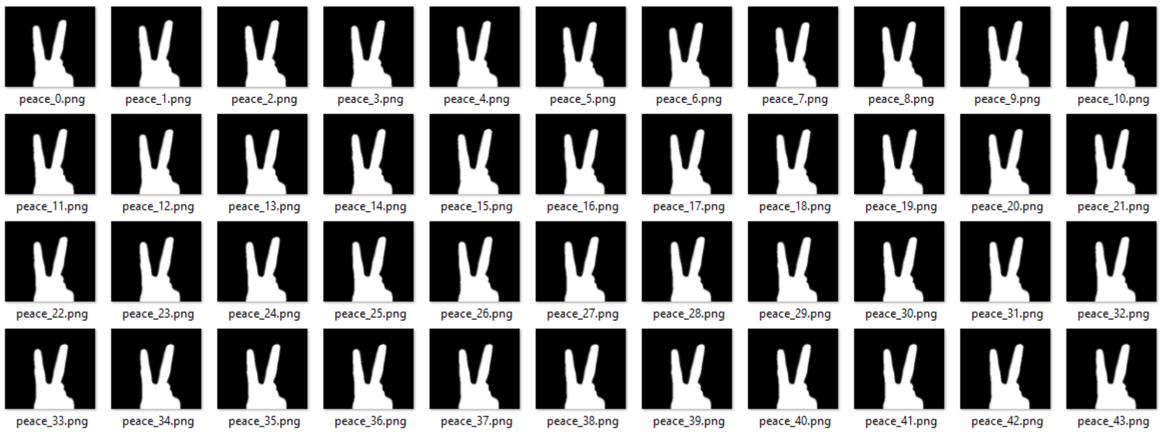


Fig 6.5.4: Peace Dataset



ii. **Resizing of Images:**

The resizeImage() function resizes the images in the dataset suitably in order to feed it into the neural network. The network accepts only images of size 89*100 pixels.

iii. **Training the model using Convolutional Neural Network (CNN):**

The CNN is designed using tensorflow. Once the dataset is prepared, it is fed into this network for training. The network contains **1** input layer, **7** hidden convolution layers with **Relu** as the activation function and **1** Fully connected layer. It is trained across **50** iterations with a batch size of **64**. The model achieves an accuracy of **99.94%** on the training dataset and **92**% on test dataset. The ratio of training set to validation set is **1000 : 100**. Once the model is trained, it is saved as a tfl file. This file is loaded at run time and used to predict the gesture.

iv. **Running the Project:**

Running the run.py file opens up your webcam and takes continuous frames of your hand image and then predicts the class of your hand gesture in realtime.

On running the project, there are 3 windows that open i.e the thresholded window displaying the captured frames after background subtraction, the video feed displaying the frame with a green rectangle indicating the region of interest (roi) and the Statistics window to display the name of the predicted gesture and the confidence level. The gestures indicate the following:

Palm is used for performing zoom in operation at runtime.

Fist is used for performing zoom out operation at runtime.

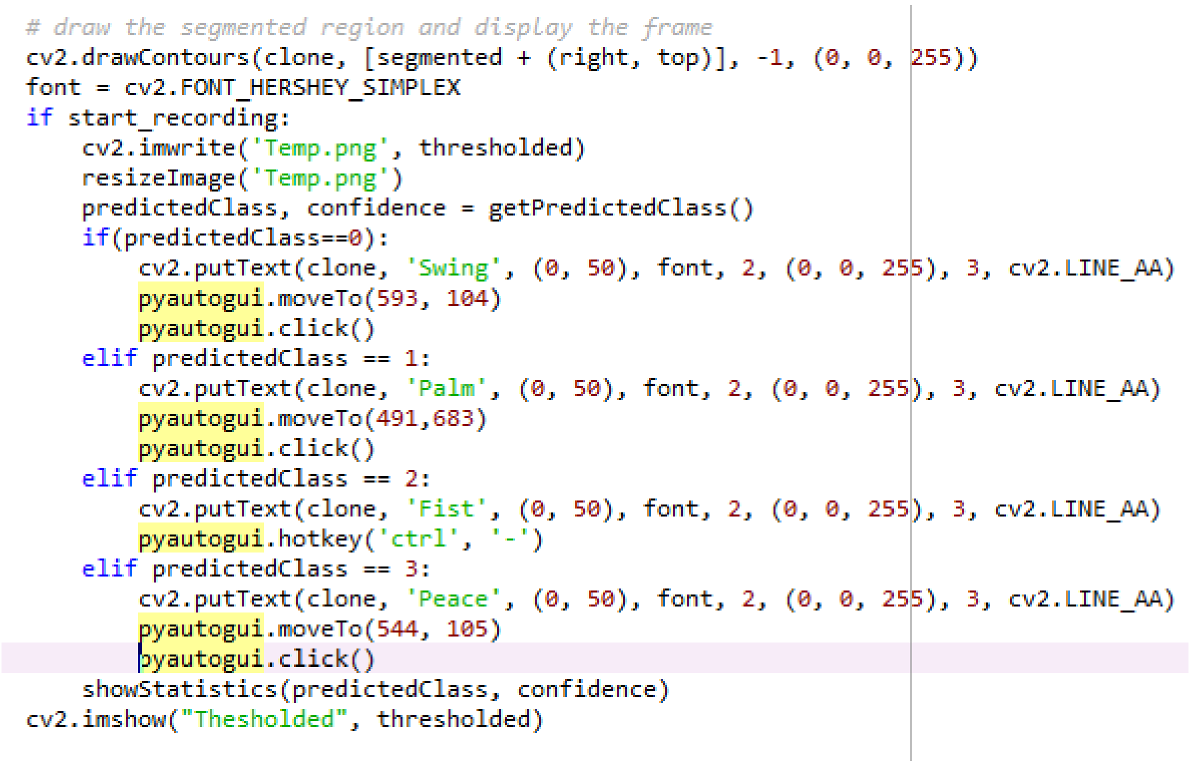
Swing is used for performing turning right operation at runtime.

Peace is used for performing turning left operation at runtime.

Now that the gestures are being detected, this output is used to perform certain real time applications. PyAutoGUI is linked here and thus actions such as zooming in , zooming out, turning the pages are performed on a PDF file.

The following image demonstrates the usage of PyAutoGUI in the code in order to perform the required actions.

Fig 6.5.5: Code for Integration of PyAutoGUI



```
# draw the segmented region and display the frame
cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
font = cv2.FONT_HERSHEY_SIMPLEX
if start_recording:
    cv2.imwrite('Temp.png', thresholded)
    resizeImage('Temp.png')
    predictedClass, confidence = getPredictedClass()
    if(predictedClass==0):
        cv2.putText(clone, 'Swing', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        pyautogui.moveTo(593, 104)
        pyautogui.click()
    elif predictedClass == 1:
        cv2.putText(clone, 'Palm', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        pyautogui.moveTo(491,683)
        pyautogui.click()
    elif predictedClass == 2:
        cv2.putText(clone, 'Fist', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        pyautogui.hotkey('ctrl', '-')
    elif predictedClass == 3:
        cv2.putText(clone, 'Peace', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        pyautogui.moveTo(544, 105)
        pyautogui.click()
    showStatistics(predictedClass, confidence)
cv2.imshow("Thesholded", thresholded)
```

CHAPTER 7

SYSTEM TESTING

Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use. When you test software, you execute a program using artificial data. You check the results of the test run for errors, anomalies, or information about the programs non-functional attributes. The testing process has two distinct goals:

1. To demonstrate to the developer and the customer that the software meets its requirements. For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
2. To discover situations in which the behaviour of the software is incorrect, undesirable, or does not conform to its specification. These are a consequence of software defects. Defect testing is concerned with rooting out undesirable system behaviour such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption.

The first goal leads to validation testing, where you expect the system to perform correctly using a given set of test cases that reflect the systems expected use. The second goal leads to defect testing, where the test cases are designed to expose defects. The test cases in defect testing can be deliberately obscure and need not reflect how the system is normally used. Of course, there is no definite boundary between these two approaches to testing. During validation testing, you will find defects in the system; during defect testing, some of the tests will show that the program meets its requirements.

Testing is part of a broader process of software verification and validation (V & V). Verification and validation are not the same thing, although they are often confused. Barry Boehm, a pioneer of software engineering, succinctly expressed the difference between them (Boehm, 1979):
‘Validation: Are we building the right product?’
‘Verification: Are we building the product right?’

Verification and validation processes are concerned with checking that software being developed meets its specification and delivers the functionality expected by the people paying for the software. These checking processes start as soon as requirements become available and continue through all stages of the development process.

The aim of verification is to check that the software meets its stated functional and non-functional requirements. Validation, however, is a more general process. The aim of validation is to ensure that the software meets the customer's expectations. It goes beyond simply checking conformance with the specification to demonstrating that the software does what the customer expects it to do. Validation is essential because, the specifications do not always reflect the real wishes or needs of system customers and users. The ultimate goal of verification and validation processes is to establish confidence that the software system is 'fit for purpose'. This means that the system must be good enough for its intended use.

7.1 Test Cases

7.1.1 Unit Testing

Unit testing is a level of software testing where individual units/components of a software are tested. The purpose is to validate that each unit of the software performs as required. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure etc.

There are various functions in our codes. They are tested individually and the following results are obtained.

1. Detect_hand - This function is tested and a boundary box detecting the hand is drawn.
2. Extract_defects - The empty space in the image is eliminated and only the hand is being detected. Blue defect points are drawn.
3. Get_Predicted_Class – Predicts to which class the gesture shown by the user belongs to.

Table 7.1: Unit Testing

Test Case Id	Description	Expected Result	Result
U_TC_001	Loading the dataset	Load the images and the labels of the training and test dataset into python lists and	Pass
U_TC_002	Training the convolutional neural network (CNN)	Generate a model on the dataset trained and save a model file.	Pass
U_TC_003	Loading the model	Load the trained model into the system for gesture recognition.	Pass
U_TC_004	Opening the web camera and display the camera frame	Start the web camera and render the frame captured by OpenCV.	Pass
U_TC_005	Drawing the region of interest (ROI)	Draw a rectangular box on the frame in the desired region of interest for the user to show the gesture.	Pass
U_TC_006	Gesture recognition and prediction	Capture the frame, recognise the gesture shown by the user in the region of user and predict it (right, left, zoom-in and zoom out)	Pass
U_TC_007	Human computer interaction using PyAutoGUI	Perform the mouse and keyboard controls through PyAutoGUI functions	Pass

7.1.2 Integration Testing

Integration testing (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Table 7.2 : Integration Testing

Test Case Id	Integration Units	Expected Result	Result
I_TC_001	Loading the dataset and Training the convolutional neural network (CNN)	The images loaded into the python lists to be used as input data to the convolutional neural network.	Pass
I_TC_002	Loading the trained model, opening the web camera and displaying the camera frame and drawing the region of interest.	Load the saved model and draw a bounding box in the region of interest on the frame.	Pass
I_TC_003	Gesture recognition and prediction and PyAutoGUI	The recognised and predicted gesture is sent to the PyAutoGUI module to perform the corresponding action	Pass

7.1.3 System Testing

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together (called assemblages). System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. The actual result is the behaviour produced or observed when a component or system is tested.

Table 7.3 : System Testing

Test Case Id	Gesture	Expected Result	Result
S_TC_001	Zoom – in (Palm)	Zoom in of the page	Pass
S_TC_002	Zoom – out (Fist)	Zoom out of the page.	Pass
S_TC_003	Right (Swing)	Turn the page right or switch to the next page.	Pass
S_TC_004	Left (Peace)	Turn the page left or switch to the previous page.	Pass
S_TC_005	Any random gesture	No action should be performed	Fail

CHAPTER 8

CONCLUSION, FUTURE ENHANCEMENT AND SNAPSHOTS

8.1 Conclusion

This project demonstrates the usage of convolutional neural networks in hand gesture recognition using OpenCV. It depicts the efficiency and usage of neural networks in a systematic manner and gives the reader a detailed understanding of CNN and its applications. The project can be useful in various areas such as

- Internet of Things (IOT)
- Automated homes,
- Self driving cars
- Making presentations
- Gaming
- Switching T.V channels

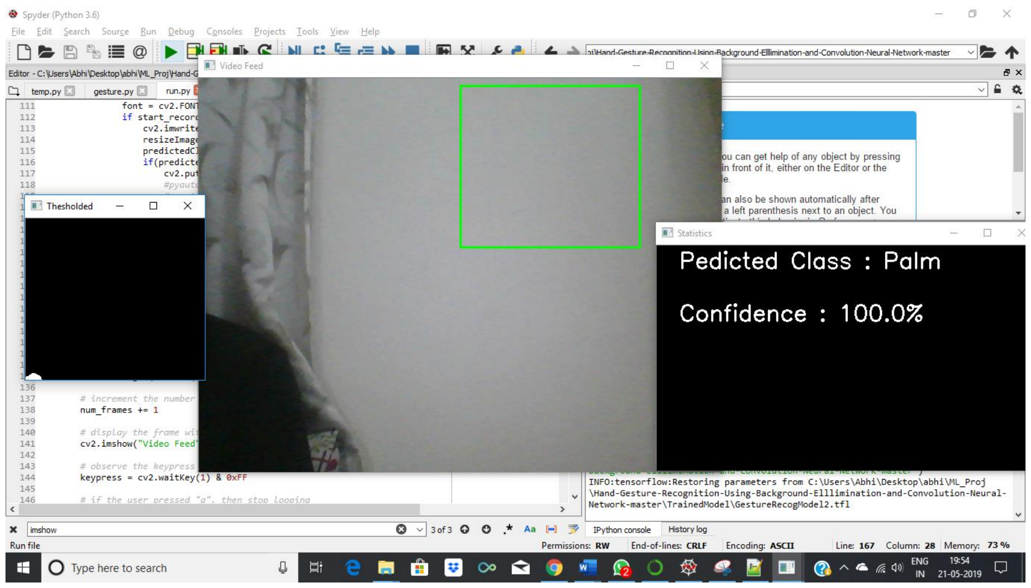
OpenCV is widely used library used in computer vision. This project gives an exposure into this library and the application for which it is used is gesture recognition. The model achieves an accuracy of **99.94%** on the training dataset and **92**% on test dataset.

8.2 Future Enhancement

1. More number of gestures
2. Extending to dynamic gestures such as swiping left and right
3. Better accuracy
4. Increased efficiency

8.3 Results and Snapshots

Fig 8.3.1: Display of Windows



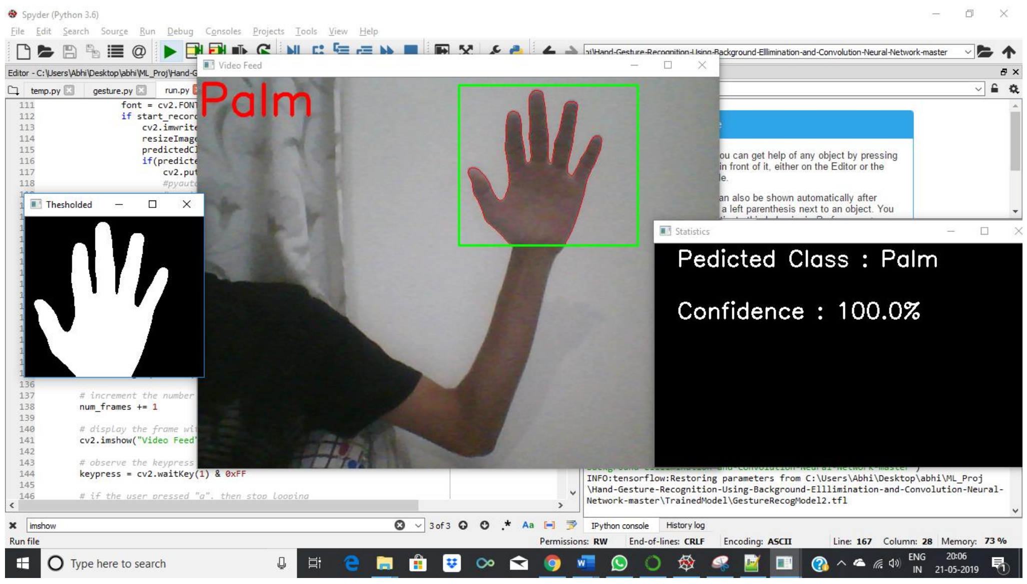
```
            font = cv2.FONT
            if start_recor
                cv2.imwrite
                resizeImage
                predictedC
                if(predicte
                    cv2.put
                    #pyaut

        # increment the number
        num_frames += 1

        # display the frame wi
        cv2.imshow("Video Feed"

        # observe the keypress
        keypress = cv2.waitKey(1) & 0xFF

        # if the user pressed "q", then stop looping
```

```
Pedicted Class : Palm

Confidence : 100.0%
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Abhi\Desktop\abhi\ML_Proj
\Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-
Network-master\TrainedModel\GestureRecogModel2.tfl
```

Run file   Permissions: RW   End-of-lines: CRLF   Encoding: ASCII   Line: 167   Column: 28   Memory: 73 %

Fig 8.3.2: Prediction of Palm Gesture (Zoom In)



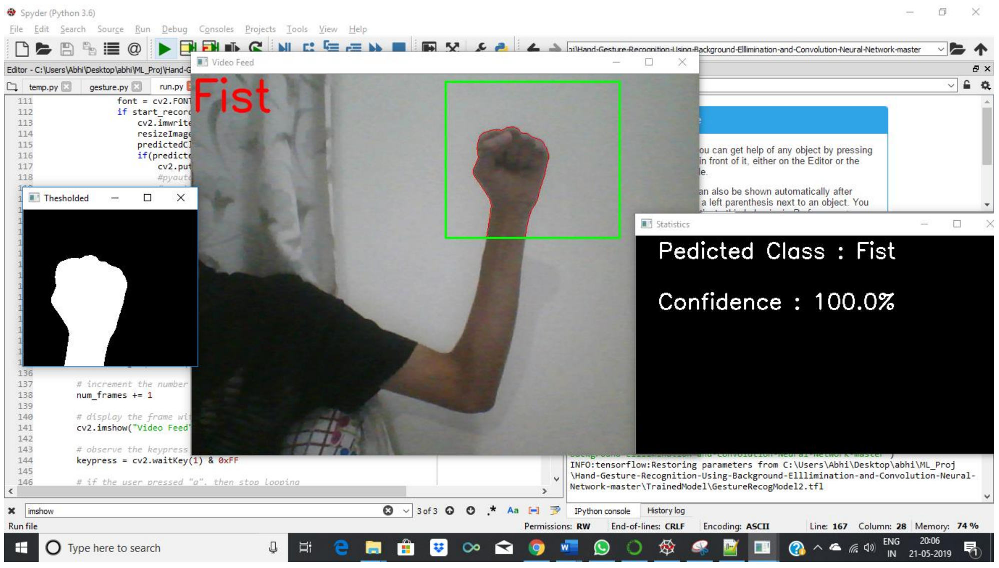
```
            font = cv2.FONT
            if start_recor
                cv2.imwrite
                resizeImage
                predictedC
                if(predicte
                    cv2.put
                    #pyaut

        # increment the number
        num_frames += 1

        # display the frame wi
        cv2.imshow("Video Feed"

        # observe the keypress
        keypress = cv2.waitKey(1) & 0xFF

        # if the user pressed "q", then stop looping
```

```
Pedicted Class : Palm

Confidence : 100.0%
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Abhi\Desktop\abhi\ML_Proj
\Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-
Network-master\TrainedModel\GestureRecogModel2.tfl
```

Run file   Permissions: RW   End-of-lines: CRLF   Encoding: ASCII   Line: 167   Column: 28   Memory: 73 %

Fig 8.3.3: Prediction of Fist Gesture (Zoom Out)



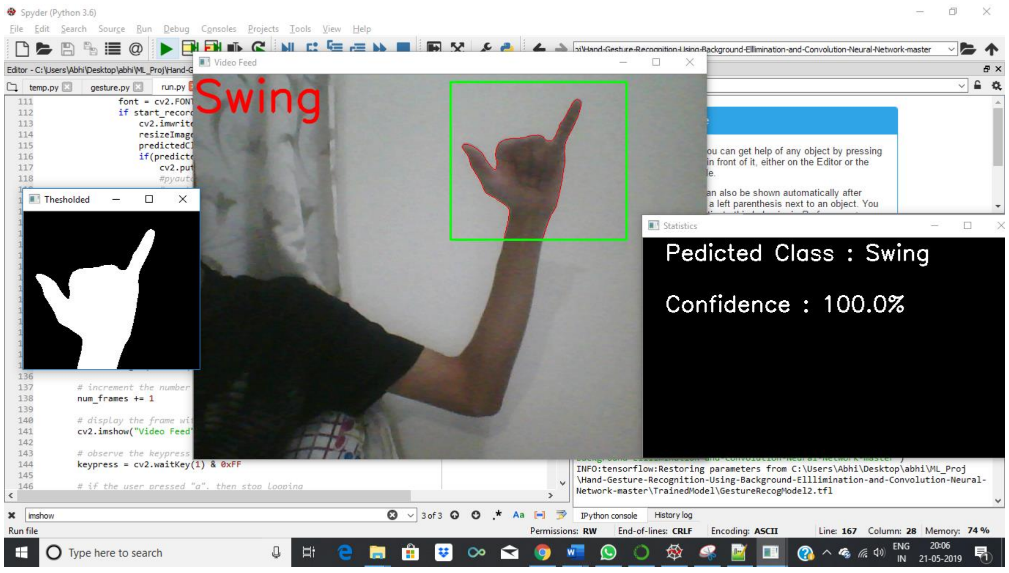
```
                font = cv2.FONT
                if start_record
                    cv2.imwrite
                    resizeImage
                    predictedCl
                    if(predicte
                        cv2.put
                        #pyauto

            # increment the number
            num_frames += 1

            # display the frame wi
            cv2.imshow("Video Feed"

            # observe the keypress
            keypress = cv2.waitKey(1) & 0xFF

            # if the user pressed "a", then stop looping
```

```
Pedicted Class : Fist

Confidence : 100.0%
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Abhi\Desktop\abhi\ML_Proj
\Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-
Network-master\TrainedModel\GestureRecogModel2.tfl
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 167 Column: 28 Memory: 74 %

Fig 8.3.4: Prediction of Swing Gesture (Right)



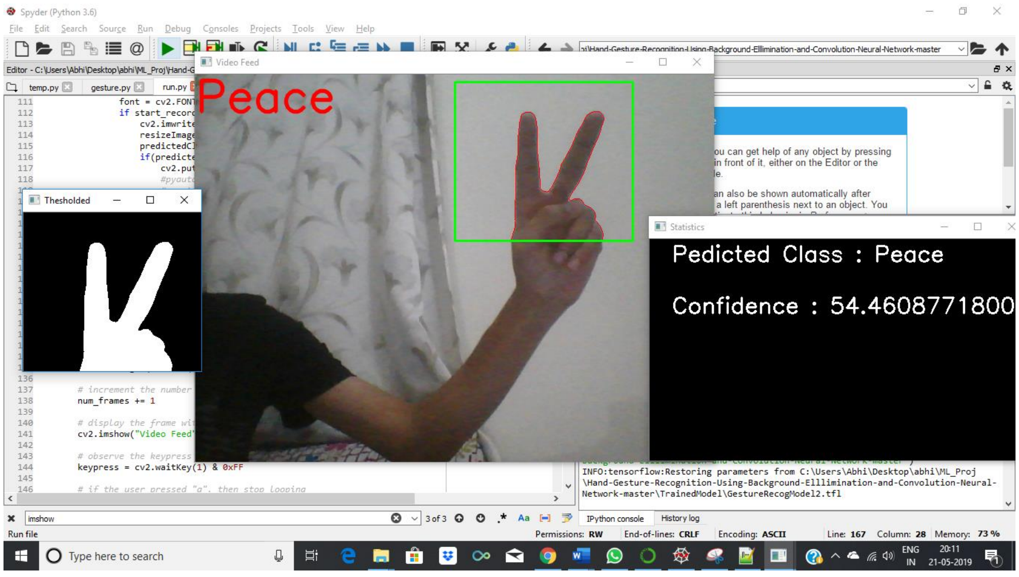
```
                font = cv2.FONT
                if start_record
                    cv2.imwrite
                    resizeImage
                    predictedCl
                    if(predicte
                        cv2.put
                        #pyauto

            # increment the number
            num_frames += 1

            # display the frame wi
            cv2.imshow("Video Feed"

            # observe the keypress
            keypress = cv2.waitKey(1) & 0xFF

            # if the user pressed "a", then stop looping
```

```
Pedicted Class : Swing

Confidence : 100.0%
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Abhi\Desktop\abhi\ML_Proj
\Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-
Network-master\TrainedModel\GestureRecogModel2.tfl
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 167 Column: 28 Memory: 74 %

Fig 8.3.5: Prediction of Peace Gesture (Left)



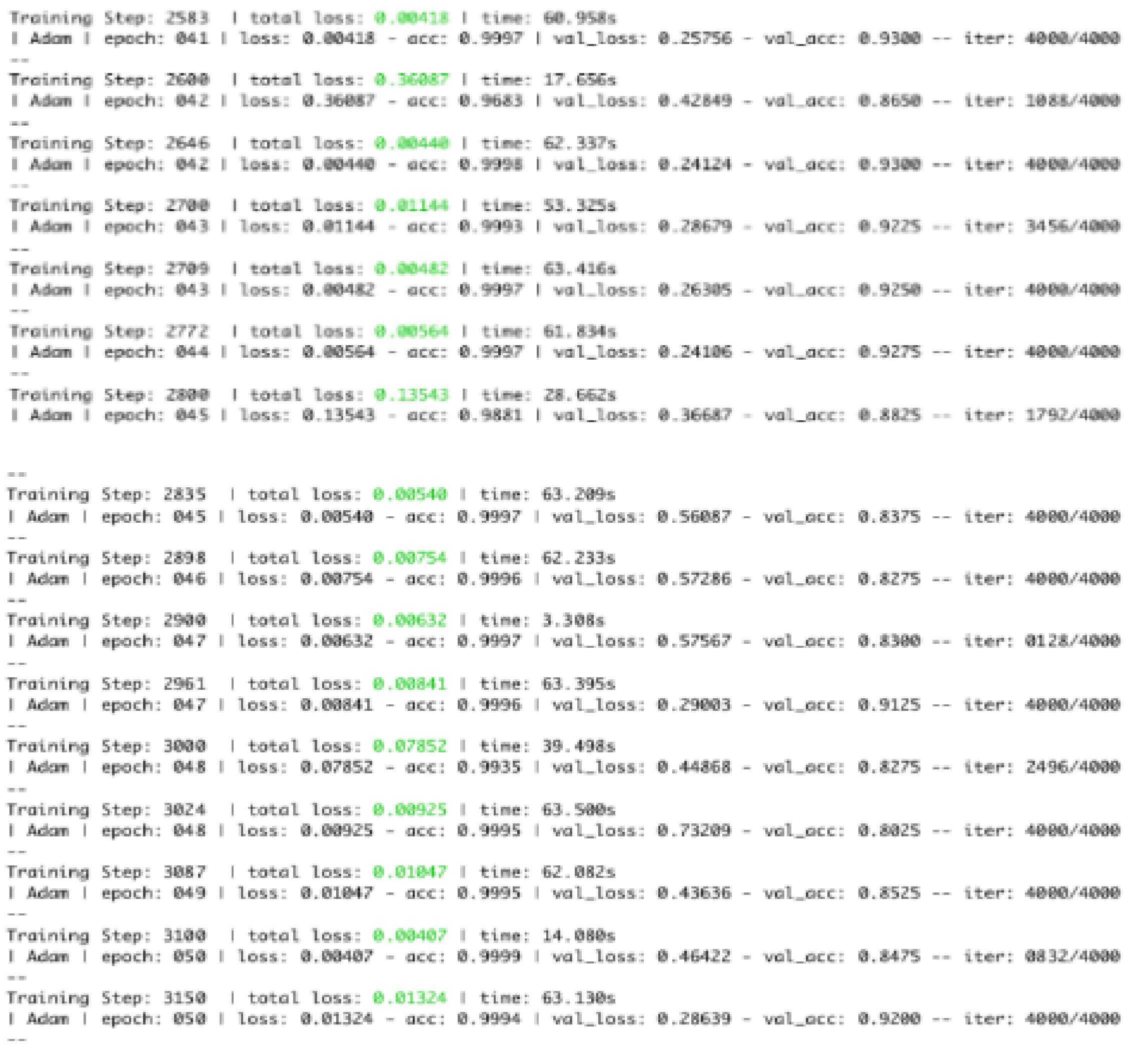
```
Pedicted Class : Peace

Confidence : 54.4608771800
```

```
        num_frames += 1

        cv2.imshow("Video Feed"

        keypress = cv2.waitKey(1) & 0xFF
```

```
INFO:tensorflow:Restoring parameters from C:\Users\Abhi\Desktop\abhi\ML_Proj
\Hand-Gesture-Recognition-Using-Background-Ellimination-and-Convolution-Neural-
Network-master\TrainedModel\GestureRecogModel2.tfl
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 167 Column: 28 Memory: 73 %

Fig 8.3.6: Training the model



```
Training Step: 2583  | total loss: 0.00418 | time: 60.958s
| Adam | epoch: 041 | loss: 0.00418 - acc: 0.9997 | val_loss: 0.25756 - val_acc: 0.9300 -- iter: 4000/4000
--
Training Step: 2600  | total loss: 0.36087 | time: 17.656s
| Adam | epoch: 042 | loss: 0.36087 - acc: 0.9683 | val_loss: 0.42849 - val_acc: 0.8650 -- iter: 1088/4000
--
Training Step: 2646  | total loss: 0.00440 | time: 62.337s
| Adam | epoch: 042 | loss: 0.00440 - acc: 0.9998 | val_loss: 0.24124 - val_acc: 0.9300 -- iter: 4000/4000
--
Training Step: 2700  | total loss: 0.01144 | time: 53.325s
| Adam | epoch: 043 | loss: 0.01144 - acc: 0.9993 | val_loss: 0.28679 - val_acc: 0.9225 -- iter: 3456/4000
--
Training Step: 2709  | total loss: 0.00482 | time: 63.416s
| Adam | epoch: 043 | loss: 0.00482 - acc: 0.9997 | val_loss: 0.26305 - val_acc: 0.9250 -- iter: 4000/4000
--
Training Step: 2772  | total loss: 0.00564 | time: 61.834s
| Adam | epoch: 044 | loss: 0.00564 - acc: 0.9997 | val_loss: 0.24106 - val_acc: 0.9275 -- iter: 4000/4000
--
Training Step: 2800  | total loss: 0.13543 | time: 28.662s
| Adam | epoch: 045 | loss: 0.13543 - acc: 0.9881 | val_loss: 0.36687 - val_acc: 0.8825 -- iter: 1792/4000


--
Training Step: 2835  | total loss: 0.00540 | time: 63.209s
| Adam | epoch: 045 | loss: 0.00540 - acc: 0.9997 | val_loss: 0.56087 - val_acc: 0.8375 -- iter: 4000/4000
--
Training Step: 2898  | total loss: 0.00754 | time: 62.233s
| Adam | epoch: 046 | loss: 0.00754 - acc: 0.9996 | val_loss: 0.57286 - val_acc: 0.8275 -- iter: 4000/4000
--
Training Step: 2900  | total loss: 0.00632 | time: 3.308s
| Adam | epoch: 047 | loss: 0.00632 - acc: 0.9997 | val_loss: 0.57567 - val_acc: 0.8300 -- iter: 0128/4000
--
Training Step: 2961  | total loss: 0.00841 | time: 63.395s
| Adam | epoch: 047 | loss: 0.00841 - acc: 0.9996 | val_loss: 0.29003 - val_acc: 0.9125 -- iter: 4000/4000
--
Training Step: 3000  | total loss: 0.07852 | time: 39.498s
| Adam | epoch: 048 | loss: 0.07852 - acc: 0.9935 | val_loss: 0.44868 - val_acc: 0.8275 -- iter: 2496/4000
--
Training Step: 3024  | total loss: 0.00925 | time: 63.500s
| Adam | epoch: 048 | loss: 0.00925 - acc: 0.9995 | val_loss: 0.73209 - val_acc: 0.8025 -- iter: 4000/4000
--
Training Step: 3087  | total loss: 0.01047 | time: 62.082s
| Adam | epoch: 049 | loss: 0.01047 - acc: 0.9995 | val_loss: 0.43636 - val_acc: 0.8525 -- iter: 4000/4000
--
Training Step: 3100  | total loss: 0.00407 | time: 14.080s
| Adam | epoch: 050 | loss: 0.00407 - acc: 0.9999 | val_loss: 0.46422 - val_acc: 0.8475 -- iter: 0832/4000
--
Training Step: 3150  | total loss: 0.01324 | time: 63.130s
| Adam | epoch: 050 | loss: 0.01324 - acc: 0.9994 | val_loss: 0.28639 - val_acc: 0.9200 -- iter: 4000/4000
--
```

APPENDIX A

CODE

Block 1: Training CNN using TensorFlow

```
import tensorflow as tf
import tflearn
from tflearn.layers.conv import conv_2d,max_pool_2d
from tflearn.layers.core import input_data,dropout,fully_connected
from tflearn.layers.estimator import regression
import numpy as np
import cv2
from sklearn.utils import shuffle
```

```
#Load Images from Swing
loadedImages = []
for i in range(0, 1000):
    image = cv2.imread('Dataset/SwingImages/swing_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From Palm
for i in range(0, 1000):
    image = cv2.imread('Dataset/PalmImages/palm_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From Fist
for i in range(0, 1000):
    image = cv2.imread('Dataset/FistImages/fist_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From Peace
for i in range(0, 1000):
    image = cv2.imread('Dataset/PeaceImages/peace_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

# Create OutputVector
outputVectors = []
for i in range(0, 1000):
    outputVectors.append([1, 0, 0, 0])

for i in range(0, 1000):
    outputVectors.append([0, 1, 0, 0])

for i in range(0, 1000):
    outputVectors.append([0, 0, 1, 0])

for i in range(0, 1000):
    outputVectors.append([0, 0, 0, 1])

testImages = []

#Load Images for swing
for i in range(0, 100):
    image = cv2.imread('Dataset/SwingTest/swing_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

#Load Images for Palm
for i in range(0, 100):
    image = cv2.imread('Dataset/PalmTest/palm_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))
#Load Images for Fist
for i in range(0, 100):
    image = cv2.imread('Dataset/FistTest/fist_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

#Load Images for Peace
for i in range(0, 100):
    image = cv2.imread('Dataset/PeaceTest/peace_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

testLabels = []
for i in range(0, 100):
    testLabels.append([1, 0, 0, 0])
```

```
for i in range(0, 100):
    testLabels.append([0, 1, 0, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 1, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 0, 1])

# Define the CNN Model
tf.reset_default_graph()
convnet=input_data(shape=[None,89,100,1],name='input')
convnet=conv_2d(convnet,32,2,activation='relu')
convnet=max_pool_2d(convnet,2)
convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=fully_connected(convnet,1000,activation='relu')
convnet=dropout(convnet,0.75)
```

```
convnet=fully_connected(convnet,4,activation='softmax')

convnet=regression(convnet,optimizer='adam',learning_rate=0.001,loss='categorical_crossent
ropy',name='regression')

model=tflearn.DNN(convnet,tensorboard_verbose=0)

# Shuffle Training Data
loadedImages, outputVectors = shuffle(loadedImages, outputVectors, random_state=0)

# Train model
model.fit(loadedImages, outputVectors, n_epoch=50,
        validation_set = (testImages, testLabels),
        snapshot_step=100, show_metric=True, run_id='convnet_coursera')

model.save("TrainedModel/GestureRecogModel.tfl")
```

Block 2: Gesture Prediction using OpenCV

```
import tensorflow as tf
import tflearn
from tflearn.layers.conv import conv_2d,max_pool_2d
from tflearn.layers.core import input_data,dropout,fully_connected
from tflearn.layers.estimator import regression
import numpy as np
from PIL import Image
import cv2
import imutils
import pyautogui

# global variables
bg = None
```

```
def resizeImage(imageName):
    basewidth = 100
    img = Image.open(imageName)
    wpercent = (basewidth/float(img.size[0]))
    hsize = int((float(img.size[1])*float(wpercent)))
    img = img.resize((basewidth,hsize), Image.ANTIALIAS)
    img.save(imageName)

def run_avg(image, aWeight):
    global bg
    # initialize the background
    if bg is None:
        bg = image.copy().astype("float")
        return

    # compute weighted average, accumulate it and update the background
    cv2.accumulateWeighted(image, bg, aWeight)

def segment(image, threshold=25):
    global bg
    # find the absolute difference between background and current frame
    diff = cv2.absdiff(bg.astype("uint8"), image)

    # threshold the diff image so that we get the foreground
    thresholded = cv2.threshold(diff,
                                threshold,
                                255,
                                cv2.THRESH_BINARY)[1]

    # get the contours in the thresholded image
    (_, cnts, _) = cv2.findContours(thresholded.copy(),
                                    cv2.RETR_EXTERNAL,
                                    cv2.CHAIN_APPROX_SIMPLE)
    # return None, if no contours detected
    if len(cnts) == 0:
        return
    else:
        # based on contour area, get the maximum contour which is the hand
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)

def main():
    # initialize weight for running average
    aWeight = 0.5

    # get the reference to the webcam
    camera = cv2.VideoCapture(0)

    # region of interest (ROI) coordinates
    top, right, bottom, left = 10, 350, 225, 590

    # initialize num of frames
    num_frames = 0
    start_recording = True

    # keep looping, until interrupted
    while(True):
        # get the current frame
        (grabbed, frame) = camera.read()

        # resize the frame
        frame = imutils.resize(frame, width = 700)

        # flip the frame so that it is not the mirror view
        frame = cv2.flip(frame, 1)
```

```
# clone the frame
clone = frame.copy()

# get the height and width of the frame
(height, width) = frame.shape[:2]

# get the ROI
roi = frame[top:bottom, right:left]

# convert the roi to grayscale and blur it
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (7, 7), 0)

# to get the background, keep looking till a threshold is reached
# so that our running average model gets calibrated
if num_frames < 30:
    run_avg(gray, aWeight)
else:
    # segment the hand region
    hand = segment(gray)

    # check whether hand region is segmented
    if hand is not None:
        # if yes, unpack the thresholded image and
        # segmented region
        (thresholded, segmented) = hand

        # draw the segmented region and display the frame
        cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
        font = cv2.FONT_HERSHEY_SIMPLEX
        if start_recording:
            cv2.imwrite('Temp.png', thresholded)
            resizeImage('Temp.png')
            predictedClass, confidence = getPredictedClass()
            if(predictedClass==0):
                cv2.putText(clone, 'Swing', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
                pyautogui.moveTo(593, 104)
                pyautogui.click()
            elif predictedClass == 1:
                cv2.putText(clone, 'Palm', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
                pyautogui.moveTo(491,683)
                pyautogui.click()
            elif predictedClass == 2:
                cv2.putText(clone, 'Fist', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
                pyautogui.hotkey('ctrl', '-')
            elif predictedClass == 3:
                cv2.putText(clone, 'Peace', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
                pyautogui.moveTo(544, 105)
                pyautogui.click()
            showStatistics(predictedClass, confidence)
        cv2.imshow("Thesholded", thresholded)

# draw the segmented hand
cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

# increment the number of frames
num_frames += 1

# display the frame with segmented hand
cv2.imshow("Video Feed", clone)

# observe the keypress by the user
keypress = cv2.waitKey(1) & 0xFF

# if the user pressed "q", then stop looping
if keypress == ord("q"):
    break
```

```
        if keypress == ord("s"):
            start_recording = True
    # free up memory
    camera.release()
    cv2.destroyAllWindows()

def getPredictedClass():
    # Predict
    image = cv2.imread('Temp.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    prediction = model.predict([gray_image.reshape(89, 100, 1)])
    return np.argmax(prediction), (np.amax(prediction) / (prediction[0][0] + prediction[0][1] +
prediction[0][2] + prediction[0][3]))

def showStatistics(predictedClass, confidence):

    textImage = np.zeros((300,512,3), np.uint8)
    className = ""
    if predictedClass == 0:
        className = "Swing"
    elif predictedClass == 1:
        className = "Palm"
    elif predictedClass == 2:
        className = "Fist"
    elif predictedClass == 3:
        className = "Peace"



    cv2.putText(textImage,"Pedicted Class : " + className,
    (30, 30),
    cv2.FONT_HERSHEY_SIMPLEX,
    1,
    (255, 255, 255),
    2)
    cv2.putText(textImage,"Confidence : " + str(confidence * 100) + '%',
    (30, 100),
    cv2.FONT_HERSHEY_SIMPLEX,
    1,
    (255, 255, 255),
    2)
    #print(confidence)
    cv2.imshow("Statistics", textImage)

# Model defined
tf.reset_default_graph()
convnet=input_data(shape=[None,89,100,1],name='input')
convnet=conv_2d(convnet,32,2,activation='relu')
convnet=max_pool_2d(convnet,2)
convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=fully_connected(convnet,1000,activation='relu')
```

```
convnet=dropout(convnet,0.75)

convnet=fully_connected(convnet,4,activation='softmax')

convnet=regression(convnet,optimizer='adam',learning_rate=0.001,loss='categorical_crossent
ropy',name='regression')

model=tflearn.DNN(convnet,tensorboard_verbose=0)

# Load Saved Model
model.load("TrainedModel/GestureRecogModel.tfl")

main()
```

REFERENCES

[1] Meenakshi Panwar, Pawan Singh Mehra**, Hand Gesture Recognition for Human Computer Interaction**, International Conference on Image Information Processing India (2011).

[2] Rafiqul Zaman Khan and Noor Adnan Ibraheem, **Comparitive Study of Hand Gesture Recognition System**, AIRCC Digital Library – 2012.

[3] Arpita Ray Sarkar, G. Sanyal, S. Majumder, **Hand Gesture Recognition Systems: A Survey**, International Journal of Computer Applications Volume 71– No.15, May 2013.

[4] Manjunath A E, Vijaya Kumar B P, Rajesh H, **Comparative Study of Hand Gesture Recognition Algorithms**, International Journal of Research in Computer and Communication Technology, Vol 3, Issue 4, April- 2014.

[5] Dnyanada R Jadhav, L. M. R. J Lobo, **Navigation of PowerPoint Using Hand Gestures, International** Journal of Science and Research (IJSR) 2015.

[6] Ruchi Manish Gurav, Premanand K. Kadbe, **Real time finger tracking and contour detection for gesture recognition using OpenCV**, IEEE Conference May 2015, Pune India.

[7] Pei Xu, Department of Electrical and Computer Engineering, University of Minnesota, **A Real-time Hand Gesture Recognition and Human-Computer Interaction System**, Research Paper April 2017.

[8] P.Suganya, R.Sathya, K.Vijayalakshmi , **Detection and Recognition of Gestures To Control The System Applications by Neural Networks**, International Journal of Pure and Applied Mathematics 2018.

[9] Xing-HanWu, Mu-Chun Su, and Pa-Chun Wang, **A Hand-Gesture-Based Control Interface for a Car-Robot**, The IEEE/RSJ International Conference on Intelligent Robots and Systems (2010).

[10] Utpal V. Solanki Nilesh H. Desai, **Hand gesture based remote control for home appliances: Handmote**, IEEE (2011).

[11] Xiaoyu Wu, Cheng Yang, Youwen Wang, Hui Li, Shengmiao Xu, **An Intelligent Interactive System Based on Hand Gesture Recognition Algorithm and Kinect**, Fifth International Symposium on Computational Intelligence and Design (2012).

[12] Prof. Yuvraj V. Parkale, **Gesture based operating system control**, Second International Conference on Advanced Computing & Communication Technologies (2012).

[13] Ganesh Choudhary B, Chethan Ram BV, **Real Time Robotic Arm Control Using Hand Gestures**, IEEE (2014).

[14] Rashmi Vashisth, Akshit Sharma, Shantanu Malhotra, Saurabh Deswal, Aman Budhraja, **Gesture Control Robot Using Accelerometer**, 4th IEEE International Conference on Signal Processing, Computing and Control (2017).

[15] Tsung-Han Tsai and Yih-Ru Tasi, **Design and Implementation of a 3D Hand Gesture Architecture System Under Complicated Environment**, IEEE (2017).

[16] Wang Zhi-heng, Cao Jiang-tao, Liu Jin-guo, Zhao Zi-qi, **Design of Human Computer Interaction Control System Based on Hand-Gesture Recognition**, IEEE (2017).