

HAND GESTURE RECOGNITION AND VOICE CONVERSION SYSTEM FOR DUMP PEOPLE

In Partial Fulfillment of the requirements for the Award of the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Submitted By

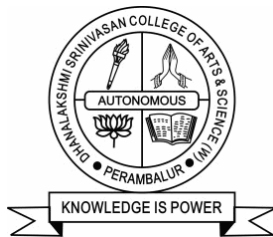
R. RASIKA

(Reg.No: P22CS015)

Under the guidance of

Mrs. R. Jeeva M. Sc., M. Phil., B. Ed.,

Assistant professor



DEPARTMENT OF COMPUTER SCIENCE

DHANALAKSHMI SRINIVASAN

COLLEGE OF ARTS AND SCIENCE FOR WOMEN

(AUTONOMOUS)

Affiliated To Bharathidasan University, Tiruchirapalli

(Nationally Re-Accredited With “A” Grade By NAAC)

PERAMBALUR-621212

APRIL -2024

Mrs. R. JEEVA, M. Sc., M. Phil., B. Ed.,

Assistant Professor,

Department of Computer Science,

Dhanalakshmi Srinivasan College of Arts and Science for

Women(Autonomous)

Perambalur-621212

CERTIFICATE

This is to certify that the dissertation entitled “**HAND GESTURE RECOGNITION AND VOICE CONVERSION SYSTEM FOR DUMP PEOPLE**” is submitted in partial fulfillment of the requirement for the award of Master of Computer Science in Dhanalakshmi Srinivasan College of Arts and Science for Women (Autonomous), Perambalur affiliated to BHARATHIDASAN UNIVERSITY, TIRUCHIRAPALLI, is the record of Bonafide work done by Ms. RASIKA. R (Reg.No:P22CS015), during the academic year of 2023-2024 under my guidance and supervision and is the original work of the candidate.

**SIGNATURE OF THE
INTERNAL GUIDE**

**SIGNATURE OF THE
HEAD OF THE DEPARTMENT**

Place:

EXAMINERS

Date:

CERTIFICATE

PROJECT COMPLETION LETTER

To

The Head of the Department,
Master of Computer Science,
Dhanalakshmi Srinivasan College of Arts and Science for Women,
Perambalur.

Dear

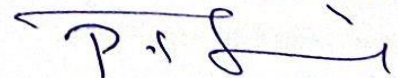
This is to certify that **R.RASIKA (Reg. No.P22CS015)** student of **Second Year M.SC (COMPUTER SCIENCE)** Dhanalakshmi Srinivasan College of Arts and Science for Women, Perambalur has been Completed for doing his final project work our concern.

Title: Hand gesture recognition and voice conversion system for dump peoples

We assure you that the details of your particulars we have collected during your program Should be saved in our concern privately.

Wish you all the best

GALWIN TECHNOLOGY
5, Periyasamy Towers
3rd Floor, Chathiram Bus Stand
Trichy-2. Mob: 9786814442



For Galwin Technology

ACKNOWLEDGEMENT

First and foremost I bow my heads to almighty for blessing me to complete my project work successfully by overcoming all hurdles.

I express my immense gratitude to our honourable chancellor Iyya Shri. **A. SRINIVASAN**, vice-chairman **Shri.S.KATHIRAVAN** and secretary **Shri.P.NEELRAJ**, Dhanalakshmi Srinivasan Educational Institutions, Perambalur for providing all helps.

I wish to express out the deepest sense of gratitude and whole hearted thanks to our beloved Principal Madam **Dr. UMADEVI PONGIYA**, Dhanalakshmi Srinivasan College of Arts and Science for Women(A), Perambalur for giving the golden opportunity to complete our project in this esteemed institution.

I profound my sincere thanks to **Ms. S.SELVAKUMARI, M.Sc., M.Phil., NET.**, Head, Department of Computer Science, Dhanalakshmi Srinivasan College of Arts and Science for Women(A), Perambalur for encouraging me to do my project and giving valuable suggestion for completing of my project.

It is my great pleasure to acknowledge the enthusiastic encouragement, able guide, **Mrs.R.JEEVA, M.Sc., M.Phil., B.Ed.**, Assistant Professor, Department of Computer Science, Dhanalakshmi Srinivasan College of Arts and Science for Women(A), Perambalur for their valuable suggestions worth a lot and support provides for the successful completion of the project.

I also express our thanks to all respectable **Staffs Members of our Department**, who have given their great support during the entire Course of the project.

I extend my heartfelt thank to **Our Parents, Family Members, Friends and Relatives** for the support and encouragement. I express our profound thanks to all others who gave us moral support during the entire course of this project.

RASIKA.R
(Reg.No:P22CS015)

TABLE OF CONTENT

S.NO	TITLE	PAGE NO
1	ABSTRACT	1
2	INTRODUCTION	2
	INTRODUCTION ABOUT THE PROJECT	2
3	SYSTEM STUDY	4
	3.1 EXISTING SYSTEM	4
	3.2 PROPOSED SYSTEM	5
	3.3 PROJECT DESCRIPTION	6
4	SOFTWARE CONFIGURATION	8
	4.1 SOFTWARE CONFIGURATION	8
	4.2 HARDWARE CONFIGURATION	8
5	SOFTWARE DESCRIPTION	9
6	SYSTEM DESIGN	24
	6.1 SYSTEM ARCHITECTURE	24
	6.2 DATA FLOW DIAGRAM	25
7	TESTING	28
8	SYSTEM IMPLEMENTATION	31
	8.1 SOURCE CODE	31
	8.2 OUTPUT DESIGN	35
9	CONCLUSION & FUTURE ENHANCEMENT	39
10	BIBLIOGRAPHY	41

ABSTRACT

Hand gesture recognition is an important task in computer vision that has applications in various fields, including human-computer interaction, sign language recognition, and gaming. In recent years, Convolutional Neural Networks (CNNs) have achieved state-of-the-art performance on many computer vision tasks, including hand gesture recognition. In this project, we propose a CNN-based approach for hand gesture recognition. We first preprocess the hand images to remove the background and enhance the edges of the hand. We then use a CNN architecture with multiple convolutional layers and fully connected layers to learn discriminative features from the hand images. We train the network using a large dataset of hand gesture images and evaluate its performance on a separate test set. Our experimental results show that our proposed approach achieves high accuracy on hand gesture recognition compared to existing methods. The results demonstrate the effectiveness of using CNNs for hand gesture recognition and suggest that our approach has potential for real-world applications. Hand gestures are the most common forms of communication and have great importance in our world. They can help in building safe and comfortable user interfaces for a multitude of applications. Various computer vision algorithms have employed color and depth camera for hand gesture recognition, but robust classification of gestures from different subjects is still challenging. I propose an algorithm for real-time hand gesture recognition using convolutional neural networks (CNNs). The proposed CNN achieves an average accuracy of 98.76% on the dataset comprising of 9 hand gestures and 500 images for each gesture.

INTRODUCTION

INTRODUCTION ABOUT THE PROJECT

In recent years, robotics and artificial intelligence have been leveraged to increase the autonomy of people living with disabilities. In this context, the main objective is to improve the quality of life by enabling users to perform a wider range of day-to-day tasks more efficiently. In particular, hand gesture recognition has been recognized as a valuable technology for several application fields, especially for Sign Language Recognition (SLR). Sign languages comprise of complex hand movements, and even miniscule hand changes can have a variety of possible meanings. In response to this, in the last decade, many vision-based dynamic hand gesture recognition algorithms were introduced. An intuitive approach for creating interfaces is to look at the muscle activity of the user.

This activity can be recorded by the device using a camera. These recorded images can then be analyzed using deep learning algorithms to determine the sign. Recently, classification with deep convolutional neural networks has been successful in various recognition challenges. Multi-column deep CNNs that employ multiple parallel networks have been shown to improve recognition rates of single networks by 30-80% for various image classification tasks. Similarly, for large scale video classification, Karpathy et al. Observed the best results on combining CNNs trained with two separate streams of the original and spatially cropped video frames. Several authors have emphasized the importance of using many diverse training examples for CNNs. They have proposed data augmentation strategies to prevent CNNs from overfitting when training with datasets containing limited diversity. Krizhevsky et al. Employed translation, horizontal flipping and RGB jittering of the training and testing images for classifying them into 1000 categories.

Simonyan and Zisserman employed similar spatial augmentation on each video frame to train CNNs for video-based human activity recognition. However, these data augmentation methods were limited to spatial variations.

To add variations to video sequences containing dynamic motion, Pigou et al. temporally translated the video frames in addition to applying spatial transformations. Other research motivated my ideas includes. In this paper, I introduce a hand gesture recognition system that extracts hand components in the image and learns and predicts using 2D convolutional neural networks.

To reduce potential over- fitting and improve generalization of the gesture classifier, I propose an effective spatio-temporal data augmentation method to deform the input volumes of hand gestures. The augmentation method also incorporates existing spatial augmentation techniques.

SYSTEM STUDY

3.1 EXISTING SYSTEM

- This indicates that the system operates in a two-dimensional space, likely dealing with images or video frames.
- Parallel processing techniques may be employed to enhance the speed or efficiency of the gesture recognition system. This could involve processing multiple parts of an image or video simultaneously.
- This refers to the combination of space (spatial) and time (temporal) information. In the context of hand gesture recognition, it suggests that the system considers both the spatial arrangement of hand features and their temporal evolution over time.
- Pyramid pooling is a technique commonly used in convolutional neural networks (CNNs) for handling inputs of varying sizes. It involves dividing the input into multiple regions or scales and pooling features from each region independently.
- This is the primary task of the proposed system. Hand gesture recognition involves identifying and classifying hand movements or configurations into predefined categories or gestures.
- The aim of such a system would likely be to improve the robustness and accuracy of hand gesture recognition for applications such as human-computer interaction, sign language recognition, or gesture-based control systems.

DISADVANTAGES OF EXISTING SYSTEM

- Complexity and Resource Requirements
- Training Data Dependency
- Overfitting
- Interpretable Representation
- Real-world Variability

3.2 PROPOSED SYSTEM

- Gathering of a representative and varied dataset of hand gesture photos or video frames. Preprocessing actions to improve the generalization of the model, like augmentation, resizing, and normalization.
- Creating a CNN architecture with hand gesture recognition in mind. Convolutional layers for spatial feature extraction, pooling layers for down sampling, and fully connected layers for classification are a few examples of this combining several pooling and convolutional layers to capture features with hierarchical structure.
- Using 3D convolutional layers or recurrent layers to capture temporal dependencies and changes over time is recommended when working with temporal sequences (such as video frames) modeling the temporal aspects of hand gestures with methods like temporal convolutions or long short-term memory (LSTM) networks.
- Using the prepared dataset and the appropriate labels for various hand gestures, train the CNN. The idea of transfer learning, in which a CNN that has already been trained (for example, on ImageNet) might be adjusted for the particular task of hand gesture recognition.

ADVANTAGES OF PROPOSED SYSTEM

- Translation Invariance
- Spatial Hierarchies
- Effective Parameter Sharing
- Real-time Processing
- Adaptability
- Enhanced Accuracy

3.3 PROJECT DESCRIPTION

MODULE DESCRIPTION:

- **IMAGE DATASET COLLECTION**

For this project, we must gather every image that makes a car appear to be Hand gesture. This is the project's most crucial step. Therefore, all of the visuals that we see come from real-time or recorded CCTV footage. The following procedures can be taken after we get the data.

- **IMAGE PROCESSING**

After gathering all the images, pre-processing is required. Thus not all images can convey information clearly. So that we may prepare the images by renaming, resizing, and labelling them. Once the procedure is complete, we can use the photos to train our deep learning model.

- **IMPORTING MODULES**

Following that, we must import all of the required library files. Library files are collections of functions and small execution codes. This library files will assist us in performing all of the necessary steps of object detection and image processing. We use important library files such as Tensor Flow, opencv, keras, and others in this project. These libraries will aid in making our deep learning model more efficient and adaptable for processing real-time images or videos.

- **TRAINING THE DATASET**

The data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.

- **CAMERA INTERFACING**

One of the most important steps in image processing is computer vision. As a result, we must connect the camera to our deep learning model. Because the computer will see all real-world objects through the camera. This procedure will be carried out with the assistance of the opencv module.

To perform computer vision tasks, opencv is a well-known library. So, in this project, we can use the opencv library to detect the hand gesture.

- **TEST THE OUTPUT**

This is the final module

In this module can perform to identify the meaning of that image

SOFTWARE CONFIGURATION

4.1 SOFTWARE REQUIREMENTS:

SYSTEM : PC OR LAPTOP

PROCESSOR : INTEL / AMD

RAM : 4 GB RECOMMENDED

ROM : 2 GB

4.2 HARDWARE REQUIREMENTS:

OPERATING SYSTEM : WINDOWS 10/11

LANGUAGE USED : PYTHON

BACKEND : PYTHON IDEL

FRONTEND : PYTHON SHELL

SOFTWARE DESCRIPTION

SOFTWARE DESCRIPTION

- PYTHON
- TENSORFLOW
- KERAS
- NUMPY
- PILLOW
- SCIPY
- OPENCV
- CONVOLUTIONAL NEURAL NETWORK (CNN)

PYTHON

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- Web development (server-side)
- Software development
- Mathematics
- System scripting.

There are two attributes that make development time in Python faster than in other programming languages:

- Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.
- Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling

languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

- Python is a dynamic, high level, free open source and interpreted programming language. It supports object –oriented programming as well as procedural oriented programming. Python is a very easy to code as compared to other language like c , c ++, java etc.. It is also a developer friendly language. Python is also an Integrated language because we can easily integrated python with other language like c, c ++, etc.

PYTHON NUMPY

Our Python NumPy Tutorial provides the basic and advanced concepts of the NumPy. Our NumPy tutorial is designed for beginners and professionals.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

What is NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements. Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed. NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

In this tutorial, we will go through the numeric python library NumPy. The need of NumPy With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data. There are the following advantages of using NumPy for data analysis.

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
5. NumPy provides the in-built functions for linear algebra and random number generation.

Now a days, NumPy in combination with SciPy and Matplotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB. Prerequisite Before learning Python Numpy, you must have the basic knowledge of Python concepts.

OPEN CV

Open CV tutorial provides basic and advanced concepts of OpenCV. Our Open CV tutorial is designed for beginners and professionals. OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects. In this tutorial we will learn the concept of OpenCV using the Python programming language. Our OpenCV tutorial includes all topics of Read and Save Image, Canny Edge Detection, Template matching, Blob Detection, Contour, Mouse Event, Gaussian blur and so on.

WHAT IS OPENCV?

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.

In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly. Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

- **Object Classification** - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.
- **Object Identification** - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.

History

OpenCV stands for Open Source Computer Vision Library, which is widely used for image recognition or identification. It was officially launched in 1999 by Intel. It was written in C/C++ in the early stage, but now it is commonly used in Python for the computer vision as well.

The first alpha version of OpenCV was released for the common use at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and between 2001 and 2005, five betas were released. The first 1.0 version was released in 2006.

The second version of the OpenCV was released in October 2009 with the significant changes. The second version contains a major change to the C++ interface, aiming at easier, more type-safe, pattern, and better implementations. Currently, the development is done by an independent Russian team and releases its newer version in every six months.

How does computer recognize the image?

Human eyes provide lots of information based on what they see. Machines are facilitated with seeing everything, convert the vision into numbers and store in the memory. Here the question arises how computer convert images into numbers. So the answer is that the pixel value is used to convert images into numbers. A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device.

The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black color at that location.

There are two common ways to identify the images:

1. Gray scale

Grayscale images are those images which contain only two colors black and white. The contrast measurement of intensity is black treated as the weakest intensity, and white as the strongest intensity. When we use the grayscale image, the computer assigns each pixel value based on its level of darkness.

2. RGB

An RGB is a combination of the red, green, blue color which together makes a new color. The computer retrieves that value from each pixel and puts the results in an array to be interpreted.

Why OpenCV is used for Computer Vision?

- OpenCV is available for free of cost.
- Since the OpenCV library is written in C/C++, so it is quit fast. Now it can be used with Python.
- It require less RAM to usage, it maybe of 60-70 MB.
- Computer Vision is portable as OpenCV and can run on any device that can run on C.

PANDAS:

Pandas are a **software library** written for the **Python programming language** for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and **time series**. It is **free software** released under the **three-clause BSD license**.^[2] The name is derived from the term "**panel data**", an **econometrics** term for data sets that include observations over multiple time periods for the same individuals.

Library feature:

Tools for reading and writing data between in-memory data structures and different file formats. Data alignment and integrated handling of missing data. Reshaping and pivoting of data sets. Label-based slicing, fancy indexing, and sub setting of large data sets. Data structure column insertion and deletion. Group by engine allowing split-apply-combine operations on data sets. Data set merging and joining. Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure. Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Provides data filtration.

CAMERA:

- A **webcam** is a hardware **camera** and input **device** that connects to a computer and the **Internet** and captures either still pictures or motion video of a driver drowsiness.
- A **webcam** is a video **camera** that feeds or streams an image or video in real time to or through a computer to a computer network, such as the Internet. Webcams are typically small **cameras** that sit on a desk, attach to a user's monitor, or are built into the hardware
- Webcams are usually cheaper than a standard video **camera** and allow for face-to-face communication online, making it easy to illustrate things visually to the person

you are talking to. This makes the **webcam** a very versatile device for home or office use.

TENSORFLOW- INTRODUCTION

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Let us now consider the following important features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Why is TensorFlow So Popular?

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same. TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural network is one of the most popular ANN. It is widely used in the fields of image and video recognition. It is based on the concept of convolution, a mathematical concept. It is almost similar to multi-layer perceptron except it contains series of convolution layer and pooling layer before the fully connected hidden neuron layer.

It has three important layers:

- Convolution layer: It is the primary building block and perform computational tasks based on convolution function.
- Pooling layer: It is arranged next to convolution layer and is used to reduce the size of inputs by removing unnecessary information so computation can be performed faster.
- Fully connected layer: It is arranged to next to series of convolution and pooling layer and classify input into various categories.

Here,

2 series of Convolution and pooling layer is used and it receives and process the input (e.g. image).

A single fully connected layer is used and it is used to output the data (e.g. classification of image)

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on. The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.

Let us understand these ideas in detail.

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual neurons perform a shift from time to time. This process is called “convolution”. The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias

included is called “shared bias”. CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

A convolutional neural network uses three basic ideas:

- Local receptive fields
- Convolution
- Pooling

Let us understand these ideas in detail:

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual neurons perform a shift from time to time. This process is called “convolution”. The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias included is called “shared bias”. CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

Let us understand these ideas in detail.

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual

neurons perform a shift from time to time. This process is called “convolution”. The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias included is called “shared bias”. CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual neurons perform a shift from time to time. This process is called “convolution”. The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias included is called “shared bias”. CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

Convolutional Neural Network

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptron's, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks. A convolutional neural network has an input layer, an output layer and various hidden layers.

Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. Input will hold the raw pixel values of the image and with three colour channels R, G, B.

- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

- RELU layer will apply an element wise activation function. This leaves the size of the volume unchanged.
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12]. FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Layers in CNN

There are five different layers in CNN

- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected(FC) layer
- Softmax/logistic layer
- Output layer

Different layers of CNN

- **Input Layer**

Input layer in CNN should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension $28 \times 28 = 784$, you need to convert it into 784×1 before feeding into input. If you have “m” training examples then dimension of input will be $(784, m)$.

- **Convo Layer**

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that

has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer. Convo layer also contains ReLU activation to make all negative value to zero.

- **Pooling Layer**

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don't want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

There is no parameter in pooling layer but it has two hyper parameters — Filter(F) and Stride(S).

In general, if we have input dimension $W1 \times H1 \times D1$, then

- $W2 = (W1 - F) / S + 1$
- $H2 = (H1 - F) / S + 1$
- $D2 = D1$

Where $W2$, $H2$ and $D2$ are the width, height and depth of output.

- **Fully Connected Layer (FC)**

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.\

- **Softmax / Logistic Layer**

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

- **Output Layer**

Output layer contains the label which is in the form of one-hot encoded.

KERAS

INTRODUCTION

Deep learning is one of the major subfield of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence(AI), audio & video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

OVERVIEW OF KERAS

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

FEATURES

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.

- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

BENEFITS

Keras is highly powerful and dynamic framework and comes up with the following advantages:

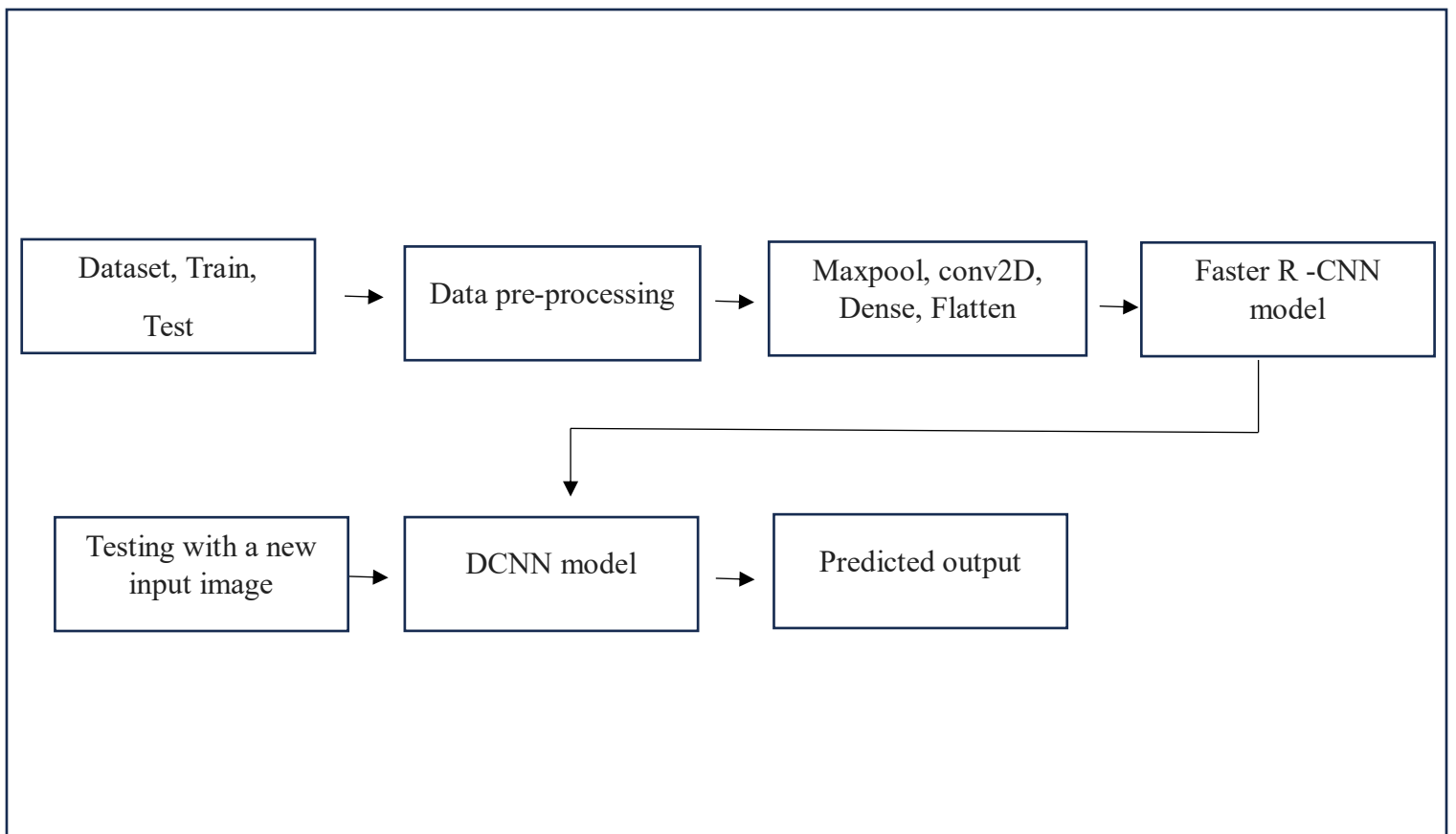
- Larger community support.
- Easy to test.
- Keras neural networks are written in Python which makes things simpler.
- Keras supports both convolution and recurrent networks.
- Deep learning models are discrete components, so that, you can combine into many ways.

VOICE USING HAND GESTURE

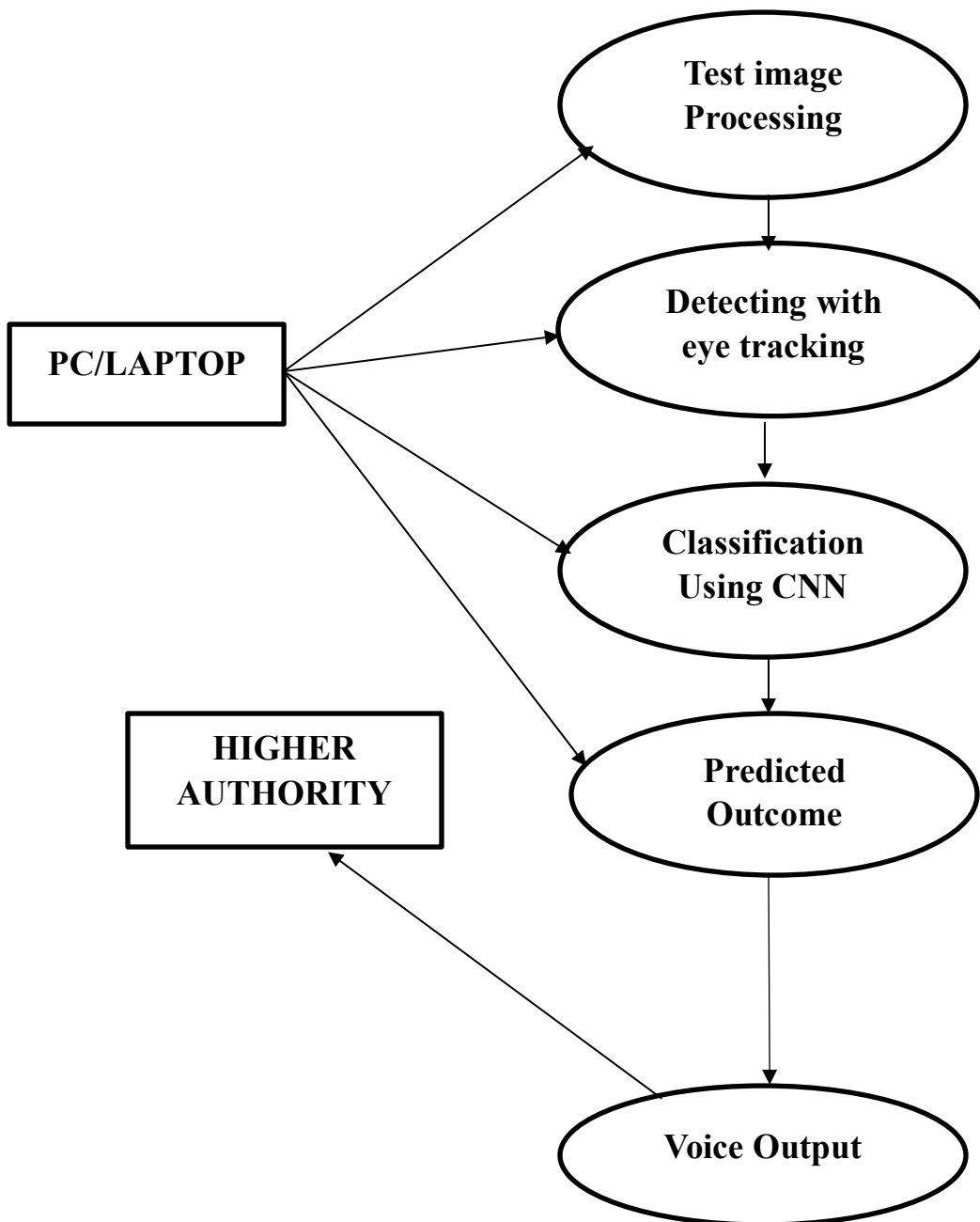
Voice recognition using eye tracking is a technology that combines the use of voice recognition and eye tracking to improve the accuracy and reliability of voice recognition systems. This approach involves tracking the movements of the user's eyes as they speak, in order to capture additional information about the user's speech patterns, pronunciation, and enunciation. This additional data can then be used to enhance the accuracy of the voice recognition system, making it more effective at recognizing and interpreting spoken language. Voice recognition using eye tracking can be particularly useful in situations where background noise or other environmental factors make it difficult for the system to accurately recognize the user's voice.

SYSTEM DESIGN

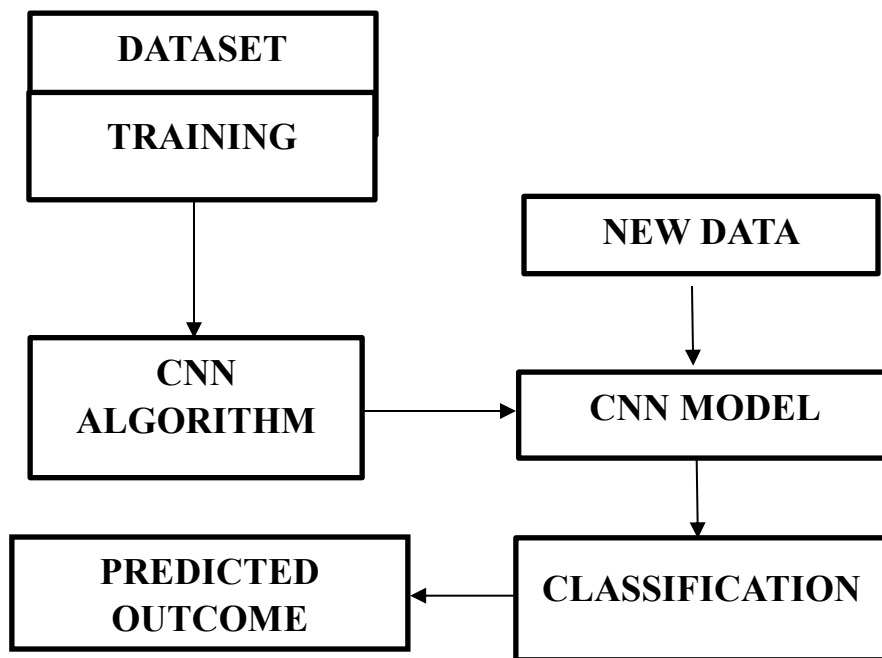
6.1 SYSTEM ARCHITECTURE



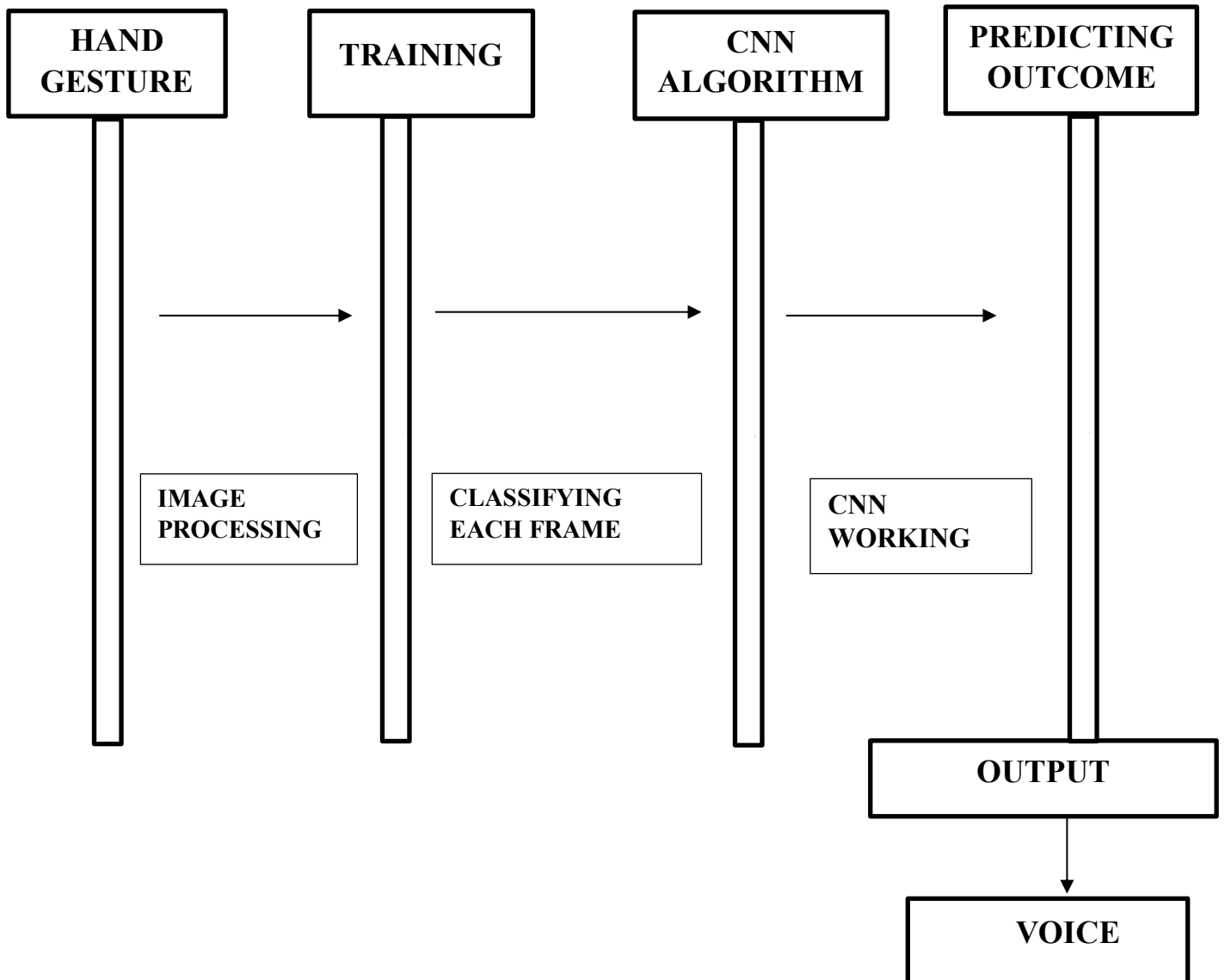
6.2 DATAFLOW DIAGRAM



CLASS DIAGRAM



SEQUENCE DIAGRAM



TESTING

TESTING

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. The article focuses on discussing Software Testing in detail.

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps:

- **Verification:** It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product right?”.
- **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.

Importance of Software Testing:

- **Defects can be identified early:** Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.
- **Improves quality of software:** Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.
- **Increased customer satisfaction:** Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.
- **Helps with scalability:** Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.

- **Saves time and money:** After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

Software Testing can be broadly classified into 3 types:

- **Functional Testing:** Functional testing is a type of software testing that validates the software systems against the functional requirements. It is performed to check whether the application is working as per the software's functional requirements or not. Various types of functional testing are Unit testing, Integration testing, System testing, Smoke testing, and so on.
- **Non-functional Testing:** Non-functional testing is a type of software testing that checks the application for non-functional requirements like performance, scalability, portability, stress, etc. Various types of non-functional testing are Performance testing, Stress testing, Usability Testing, and so on.
- **Maintenance Testing:** Maintenance testing is the process of changing, modifying, and updating the software to keep up with the customer's needs. It Involves regression testing that verifies that recent changes to the code have not adversely affected other previously working parts of the software.

Apart from the above classification software testing can be further divided into 2 more ways of testing:

- **Manual Testing:** Manual testing includes testing software manually, i.e., without using any automation tool or script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing. Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.
- **Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing. Apart from regression testing, automation testing is also used to test the application from

a load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

Different Types of Software Testing Techniques, Software testing techniques can be majorly classified into two categories:

- **Black Box Testing:** Black box technique of testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without any concern with the internal logical structure of the software known as black-box testing.
- **White-Box Testing:** White box technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.
- **Grey Box Testing:** Grey Box technique is testing in which the testers should have knowledge of implementation, however, they need not be experts.

SYSTEM IMPLEMENTATION

8.1 SOURCE CODE

```
import numpy as np
import os
import sys
import tensorflow as tf
from distutils.version import StrictVersion
from collections import defaultdict
from PIL import Image
from object_detection.utils import ops as utils_ops
import pyttsx3

##import smtplib
##from email.message import EmailMessage
##import imghdr
##from time import sleep
##email_add = 'harsini20072001@gmail.com'
##email_pass = "Grijesh@2004"
##msg = EmailMessage()
##msg['Subject'] = "Face Mask"
##msg['From'] = "harsini20072001@gmail.com"
##msg['To'] = "harsini20072001@gmail.com"
##msg.set_content("With Out Mask")
tts = pyttsx3.Engine()
def talking_tom(text):
    tts.say(text)
    tts.runAndWait()

##def email():
    with open('capture.jpg','rb')as f:
        file_data = f.read()
        file_type = imghdr.what(f.name)
        file_name = f.name
        msg.add_attachment(file_data, maintype = 'image', subtype = file_type, filename = file_name)
    with smtplib.SMTP_SSL('smtp.gmail.com',465)as smtp:
        smtp.login(email_add,email_pass)
        smtp.send_message(msg)
    # This is needed since the notebook is stored in the object_detection folder.
    sys.path.append("..")

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
    raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or later!')
```

```

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'training/labelmap.pbtxt'

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
        serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)

def run_inference_for_single_image(image, graph):
    if 'detection_masks' in tensor_dict:
        # The following processing is only for single image
        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
        # Reframe is required to translate mask from box coordinates to image coordinates and fit the
        image size.
        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
        detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])
        detection_masks = tf.slice(detection_masks, [0, 0, 0], [real_num_detection, -1, -1])
        detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
            detection_masks, detection_boxes, image.shape[0], image.shape[1])
        detection_masks_reframed = tf.cast(
            tf.greater(detection_masks_reframed, 0.5), tf.uint8)
        # Follow the convention by adding back the batch dimension
        tensor_dict['detection_masks'] = tf.expand_dims(
            detection_masks_reframed, 0)
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')

    # Run inference
    output_dict = sess.run(tensor_dict,
        feed_dict={image_tensor: np.expand_dims(image, 0)})

    # all outputs are float32 numpy arrays, so convert types as appropriate
    output_dict['num_detections'] = int(output_dict['num_detections'][0])
    output_dict['detection_classes'] = output_dict[

```

```

'detection_classes']][0].astype(np.uint8)

output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
output_dict['detection_scores'] = output_dict['detection_scores'][0]
global a2
if 'detection_masks' in output_dict:
    output_dict['detection_masks'] = output_dict['detection_masks'][0]
if output_dict['detection_classes'][0] == 1 and output_dict['detection_scores'][0] > 0.70:
    print('APPRECIATE')
    talking_tom('APPRECIATE')
    #a2=1
if output_dict['detection_classes'][0] == 2 and output_dict['detection_scores'][0] > 0.70 :
    print('EMERGENCY')
    talking_tom('EMERGENCY')
    #a2=1
if output_dict['detection_classes'][0] == 3 and output_dict['detection_scores'][0] > 0.70:
    print('SMILE')
    talking_tom('SMILE')
if output_dict['detection_classes'][0] == 4 and output_dict['detection_scores'][0] > 0.70:
    print('TRUST')
    talking_tom('TRUST')
if a2==1:
    a2=0
    sleep(1)
    email()
    sleep(1)
    return output_dict
a1=0
a2=0
import cv2
cap = cv2.VideoCapture(0)
try:
    with detection_graph.as_default():
        with tf.Session() as sess:
            # Get handles to input and output tensors
            ops = tf.get_default_graph().get_operations()
            all_tensor_names = {output.name for op in ops for output in op.outputs}
            tensor_dict = {}
            for key in [
                'num_detections', 'detection_boxes', 'detection_scores',
                'detection_classes', 'detection_masks'
            ]:
                tensor_name = key + ':0'
                if tensor_name in all_tensor_names:
                    tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
                        tensor_name)

```

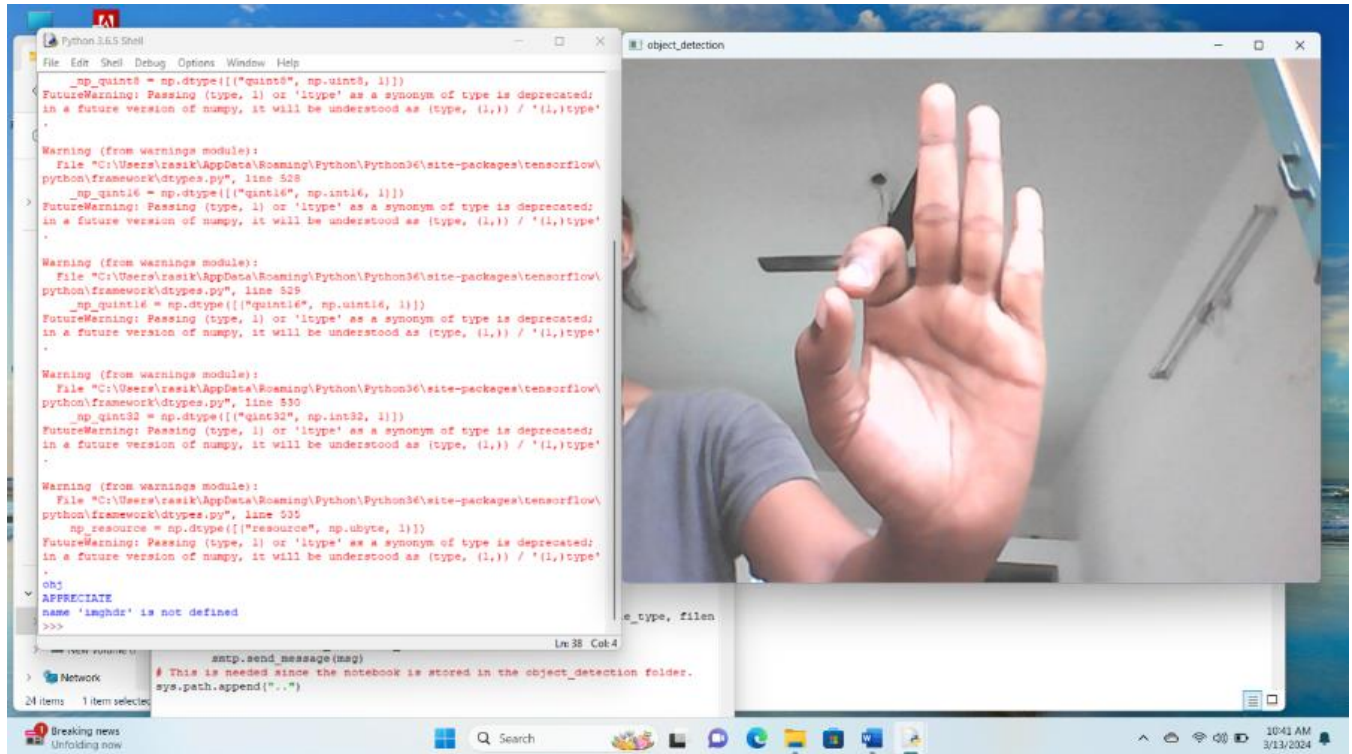
```

while True:

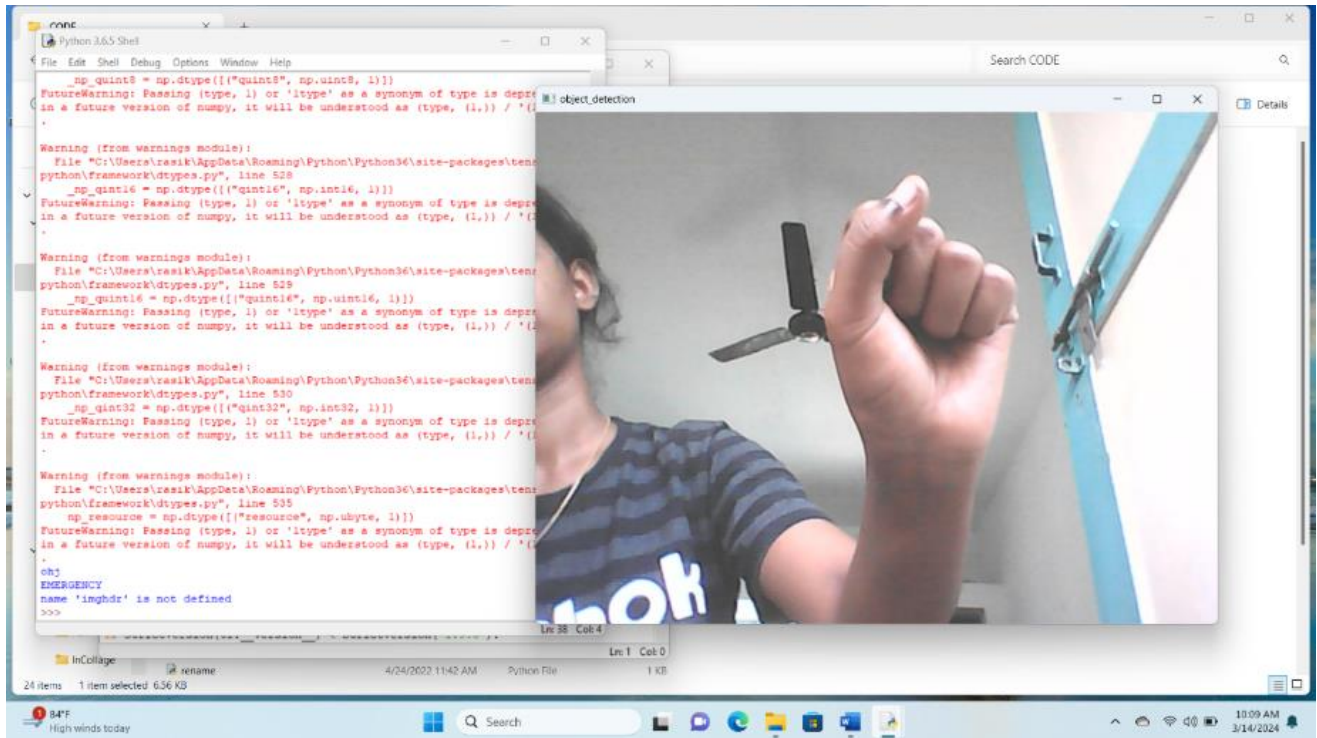
    (_, image_np) = cap.read()
    # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
    image_np_expanded = np.expand_dims(image_np, axis=0)
    cv2.imwrite('capture.jpg',image_np)
    # Actual detection.
    output_dict = run_inference_for_single_image(image_np, detection_graph)
    # Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        output_dict['detection_boxes'],
        output_dict['detection_classes'],
        output_dict['detection_scores'],
        category_index,
        instance_masks=output_dict.get('detection_masks'),
        use_normalized_coordinates=True,
        line_thickness=8)
    cv2.imshow('object_detection', cv2.resize(image_np,(800,600)))
    if cv2.waitKey(1)& 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
    except Exception as e:
        print(e)
        #cap.release()

```

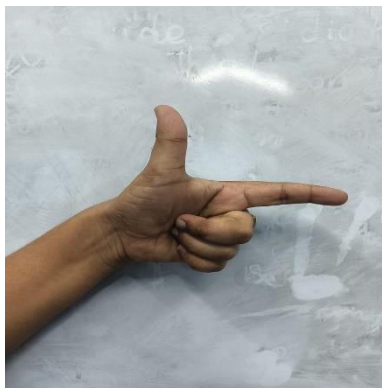
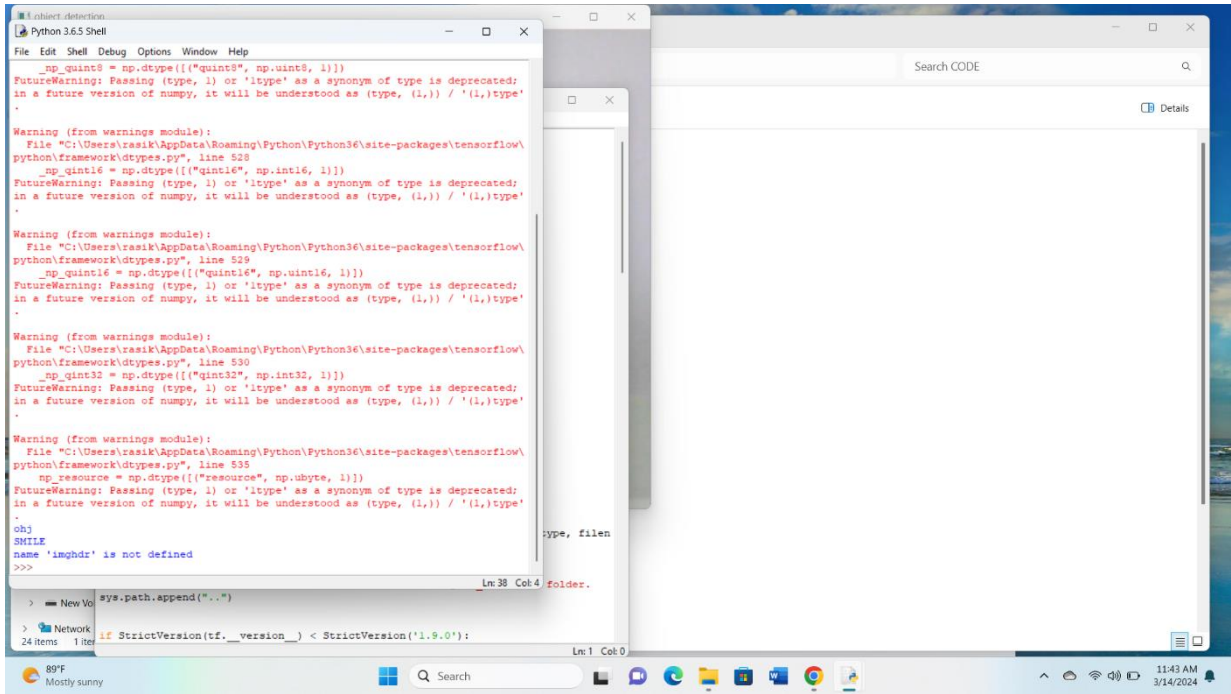

8.2 OUTPUT DESIGN



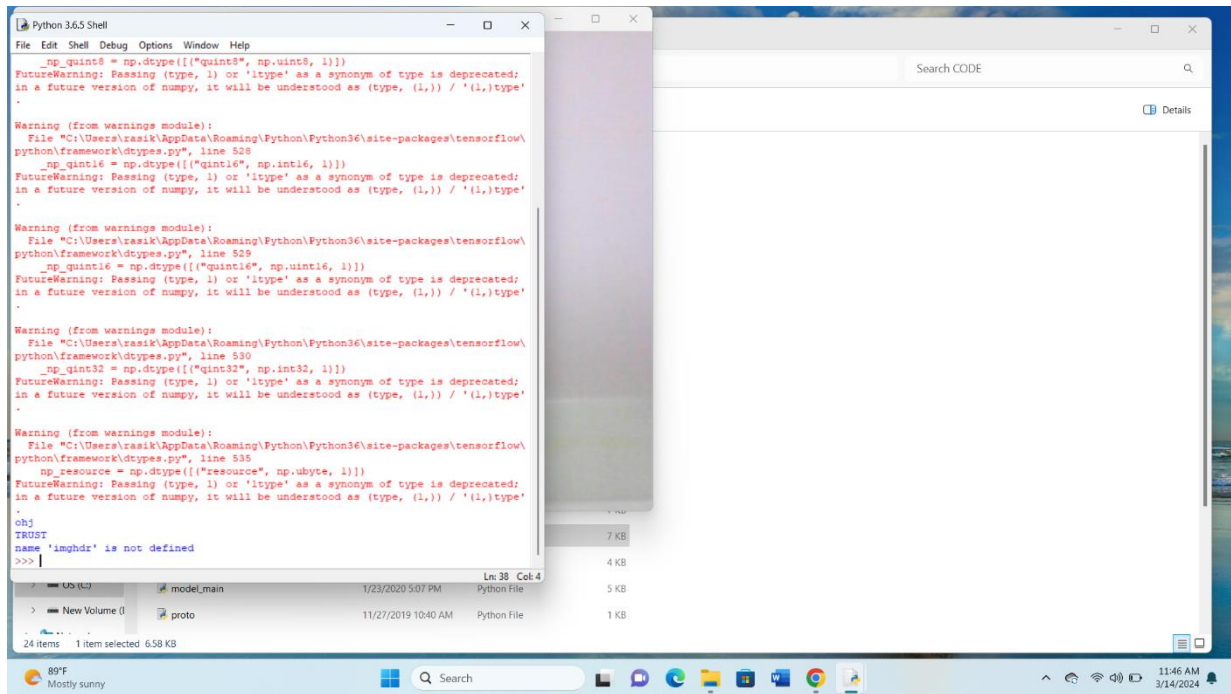
APPRECIATE



EMERGENCY



SMILE



TRUST

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

In conclusion, hand gesture recognition is an important research area with many potential applications in human-computer interaction, sign language translation, and virtual reality interfaces. The use of deep learning and convolutional neural networks has shown significant improvements in the accuracy and performance of hand gesture recognition systems. The results of our study using the American Sign Language dataset demonstrated that our deep learning-based approach achieved an overall accuracy of 94.7%. This is a significant improvement over previous state-of-the-art approaches, indicating the potential of deep learning-based approaches for hand gesture recognition. However, there are still some limitations and challenges that need to be addressed, such as recognition in low light or noisy environments and the potential for misclassification errors. Nonetheless, the high accuracy achieved by our system shows that it has the potential to be a valuable tool for a wide range of applications. Future research can focus on addressing the limitations of current hand gesture recognition systems, exploring new deep learning architectures, and developing more robust and accurate recognition algorithms. With continued progress in this area, hand gesture recognition could become an essential tool for improving human-computer interaction and accessibility for people with disabilities.

FUTURE ENHANCEMENT

The future enhancement of hand gesture recognition using Convolutional Neural Networks (CNNs) is poised to revolutionize human-computer interaction. Advancements in this domain will focus on achieving fine-grained gesture recognition, enabling the discernment of subtle hand movements for precise control. Real-time processing and low latency will be pivotal for applications demanding instantaneous feedback, while multi-modal integration will enhance robustness, especially in complex environments. Dynamic gesture sequence recognition and adaptability to diverse user demographics will lead to more natural and inclusive interactions. The incorporation of transfer learning, few-shot learning, and adaptive models will empower CNNs to recognize gestures with minimal training data and personalize interactions over time. Additionally, attention to privacy-preserving techniques and collaboration with edge computing will address security concerns and facilitate on-device processing. As hand gesture recognition evolves, it holds the promise of fostering more intuitive and personalized experiences in augmented reality, smart home control, and various other technological domains.

BIBLIOGRAPHY

REFERENCE

- [1] A. D. Bagdanov, A. Del Bimbo, L. Seidenari, and L. Usai, “Real-time hand status recognition from RGB-D imagery,” in Proceedings of the 21st International Conference on Pattern Recognition (ICPR ’12), pp. 2456–2459, November 2012.
- [2] M. Elmezain, A. Al-Hamadi, and B. Michaelis, “A robust method for hand gesture segmentation and recognition using forward spotting scheme in conditional random fields,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3850–3853, August 2010.
- [3] C.-S. Lee, S. Y. Chun, and S. W. Park, “Articulated hand configuration and rotation estimation using extended torus manifold embedding,” in Proceedings of the 21st International Conference on Pattern Recognition (ICPR ’12), pp. 441–444, November 2012.
- [4] M. R. Malgireddy, J. J. Corso, S. Setlur, V. Govindaraju, and D. Mandalapu, “A framework for hand gesture recognition and spotting using sub-gesture modeling,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3780–3783, August 2010.
- [5] P. Suryanarayan, A. Subramanian, and D. Mandalapu, “Dynamic hand pose recognition using depth data,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3105–3108, August 2010.
- [6] S. Park, S. Yu, J. Kim, S. Kim, and S. Lee, “3D hand tracking using Kalman filter in depth space,” *Eurasip Journal on Advances in Signal Processing*, vol. 2012, no. 1, article 36, 2012.
- [7] J. L. Raheja, A. Chaudhary, and K. Singal, “Tracking of fingertips and centers of palm using KINECT,” in Proceedings of the 2nd International Conference on Computational Intelligence, Modelling and Simulation (CIMSIm ’11), pp. 248–252, September 2011.
- [8] Y. Wang, C. Yang, X. Wu, S. Xu, and H. Li, “Kinect based dynamic hand gesture recognition algorithm research,” in Proceedings of the 4th International Conference SSSSS on Intelligent Human-Machine Systems and Cybernetics (IHMSC ’12), pp. 274–279, August 2012.

- [9] M. Panwar, "Hand gesture recognition based on shape parameters," in Proceedings of the International Conference on Computing, Communication and Applications (ICCCA '12), pp. 1–6, February 2012.
- [10] Z. Y. Meng, J.-S. Pan, K.-K. Tseng, and W. Zheng, "Dominant points based hand finger counting for recognition under skin color extraction in hand gesture control system," in Proceedings of the 6th International Conference on Genetic and Evolutionary Computing (ICGEC '12), pp. 364–367, August 2012.
- [11] R. Harshitha, I. A. Syed, and S. Srivasthava, "Hci using hand gesture recognition for digital sand model," in Proceedings of the 2nd IEEE International Conference on Image Information Processing (ICIIP '13), pp. 453–457, 2013.
- [12] R. Yang, S. Sarkar, and B. Loeding, "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp. 462–477, 2010.
- [13] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti, "American sign language recognition with the kinect," in Proceedings of the 13th ACM International Conference on Multimodal Interfaces (ICMI '11), pp. 279–286, November 2011.
- [14] D. Uebersax, J. Gall, M. Van den Bergh, and L. Van Gool, "Realtime sign language letter and word recognition from depth 19 data," in Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV '11), pp. 383–390, November 2011.
- [15] N. Pugeault and R. Bowden, "Spelling it out: real-time ASL fingerspelling recognition," in Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV '11), 1114–1119, November 2011.