"Real-Time Sign Language Detection and Recognition"

A Project Report Submitted to
Rajiv Gandhi Proudyogiki Vishwavidyalaya

Towards Partial Fulfillment for the Award of
Bachelor of Engineering in *Computer Science Engineering*

Submitted by:
Sarthak Sharma (0827CS181187)
Preet Kaur Nagi (0827CS181154)
Rahul Ahuja (0827CS181160)
Poorti Rajani (0827CS181144)

Guided by:
Prof. Narendra Pal Rathore
Computer Science & Engineering Department
AITR, Indore

Acropolis Institute of Technology & Research, Indore
JAN - JUNE 2022

EXAMINER APPROVAL

The Project entitled ***"Real-Time Sign Language Detection and Recognition"*** submitted by **Sarthak Sharma (0827CS181187), Preet Kaur Nagi (0827CS181154), Rahul Ahuja (0827CS181160), Poorti Rajani (0827CS181144)** has been examined and is hereby approved towards partial fulfillment for the award of ***Bachelor of Engineering degree in Computer Science Engineering*** discipline, for which it has been submitted. It understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it has been submitted.

(Internal Examiner)
Date:

(External Examiner)
Date:

GUIDE RECOMMENDATION

This is to certify that the work embodied in this project entitled "***Real-Time Sign Language Detection and Recognition***" submitted by **Sarthak Sharma (0827CS181187), Preet Kaur Nagi (0827CS181154), Rahul Ahuja (0827CS181160), Poorti Rajani (0827CS181144)** is a satisfactory account of the bonafide work done under the supervision of **Prof. Narendra Pal Rathore**, is recommended towards partial fulfillment for the award of the Bachelor of Engineering (Computer Science Engineering) degree by Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal.

(Project Guide) **(Project Coordinator)**

STUDENTS UNDERTAKING

This is to certify that project entitled ***"Real-Time Sign Language Detection and Recognition"*** has developed by us under the supervision of ***Dr. Praveen Bhanodia.*** The whole responsibility of work done in this project is ours. The sole intension of this work is only for practical learning and research.

We further declare that to the best of our knowledge, this report does not contain any part of any work which has been submitted for the award of any degree either in this University or in any other University / Deemed University without proper citation and if the same work found then we are liable for explanation to this.

Sarthak Sharma (0827CS181187)

Preet Kaur Nagi (0827CS181154)

Rahul Ahuja (0827CS181160)

Poorti Rajani (0827CS181144)

Acknowledgement

We thank the almighty for giving us the strength and courage to sail out through the tough and reach on shore safely.

There are number of people without whom this projects work would not have been feasible. Their high academic standards and personal integrity provided me with continuous guidance and support.

We owe a debt of sincere gratitude, deep sense of reverence and respect to our guide and mentor **Prof. Narendra Pal**, AITR, Indore for her motivation, sagacious guidance, constant encouragement, vigilant supervision and valuable critical appreciation throughout this project work, which helped us to successfully complete the project on time.

We express profound gratitude and heartfelt thanks to **Dr. Kamal Kumar Sethi**, HOD CSE, AITR Indore for his support, suggestion and inspiration for carrying out this project. I am very much thankful to other faculty and staff members of CSE Dept, AITR Indore for providing us all support, help and advice during the project. We would be failing in our duty if do not acknowledge the support and guidance received from **Dr. S C Sharma**, Director, AITR, Indore whenever needed. We take opportunity to convey my regards to the management of Acropolis Institute, Indore for extending academic and administrative support and providing me all necessary facilities for project to achieve our objectives.

We are grateful to **our parent** and **family members** who have always loved and supported us unconditionally. To all of them, we want to say “Thank you”, for being the best family that one could ever have and without whom none of this would have been possible.

Sarthak Sharma (0827CS181187), Preet Kaur Nagi (0827CS181154), Rahul Ahuja (0827CS181160), Poorti Rajani (0827CS181144)

Executive Summary

Real-Time Sign Language Detection and Recognition

This project is submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal (MP), India for partial fulfillment of Bachelor of Engineering in Computer Science Engineering branch under the sagacious guidance and vigilant supervision of ***Prof. Praveen Bhanodia.***

The project is based on Machine Learning, developed for effective interaction between normal people and D&M people. A language barrier is created as sign language structure which is different from normal text. So they depend on vision based communication for interaction.

In our project we basically focus on producing a model which can recognise Fingerspelling based hand gestures in order to form a complete word by combining each gesture. If there is a common interface that converts the sign language to text the gestures can be easily understood by the other people.

Key words: Image Processing, Convolutional Neural Networks, Tensorflow.

“Where the vision is one

year, cultivate flowers;

Where the vision is ten years,

cultivate trees;

Where the vision is eternity,

cultivate people.”

- Oriental Saying

List of Figures

List of Abbreviations

Abbr1: OpenCV- Open Source Computer Vision

Abbr2: Convolution Neural Network (CNN)

Abbr3: Tensorflow

Abbr4: CIF- Count In Frame

Abbr5: GPU- Graphical Processing Unit

Table of Contents

Chapter 1. Introduction

Introduction

Real Time Sign Language detection by technology is an overlooked concept despite there being a large social group which could benefit by it. There are not many technologies which help in connecting this social group to the rest of the world. Understanding sign language is one of the primary enablers in helping users of sign language communicate with the rest of the society.

Image classification and machine learning can be used to help computers recognize sign language, which could then be interpreted by other people.

Convolutional neural networks have been employed in this paper to recognize sign language gestures. The image dataset used consists of static sign language gestures captured on an RGB camera. Preprocessing was performed on the images, which then served as the cleaned input. The paper presents results obtained by retraining and testing this sign language gestures dataset on a convolutional neural network model using Inception v3. The model consists of multiple convolution filter inputs that are processed on the same input. The validation accuracy obtained was above 90% .This paper also reviews the various attempts that have been made at sign language detection using machine learning and depth data of images. It takes stock of the various challenges posed in tackling such a problem, and outlines future scope as well.

1.1 Overview

Speech impaired people face problems in communicating with people around them. Their family members and close friends learn sign language to communicate with them but other than that, normally, people do not know sign language and don't even tryto learn, as they do not need it in their day- to-day life. The society does not understandhow difficult it is for them to Communicate with someone who does not know sign language.

Sign language is a language in which communication between people are made by visually transmitting the sign patterns to express meaning. Sign language also includes facial expression and body language, these are known as non-manual signs and they play a major role in understanding correct meaning of signs. Manual signs are the hand patterns and it contains the main information while non-manual signs are important for clarification. Speech impaired people learn sign language so they can communicate with people around them and fulfill their day-to-day needs.

Speech impaired people prefer sign language which is used in their region. To bridge the communication gap between speech impaired people and normal people, a sign language recognition system can be build that converts sign language to text or speech using existing technologies.

Sign language recognition can be done by two methods. First one is sensor based technique, in this technique a sensor based wearable gadgets are used such as gloves. This technique has two disadvantages, first one is that the gadgets are expensive and second these gadgets need to be carried everywhere otherwise the system would not work. Second approach involves computer vision based methods. In these methods, camera is used to capture images for recognition. The second approach is more convenient to use as it does not need the user to wear any gadgets.

Many research work has been done on sign language recognition but most of the work is done on American sign language. Other languages are not explored as much as ASL. ASL is a single handed sign language, which makes it easier to work with, whereas in ISL some signs are performed with one hand and some signs are performed using both the hands which makes it complex to work with.

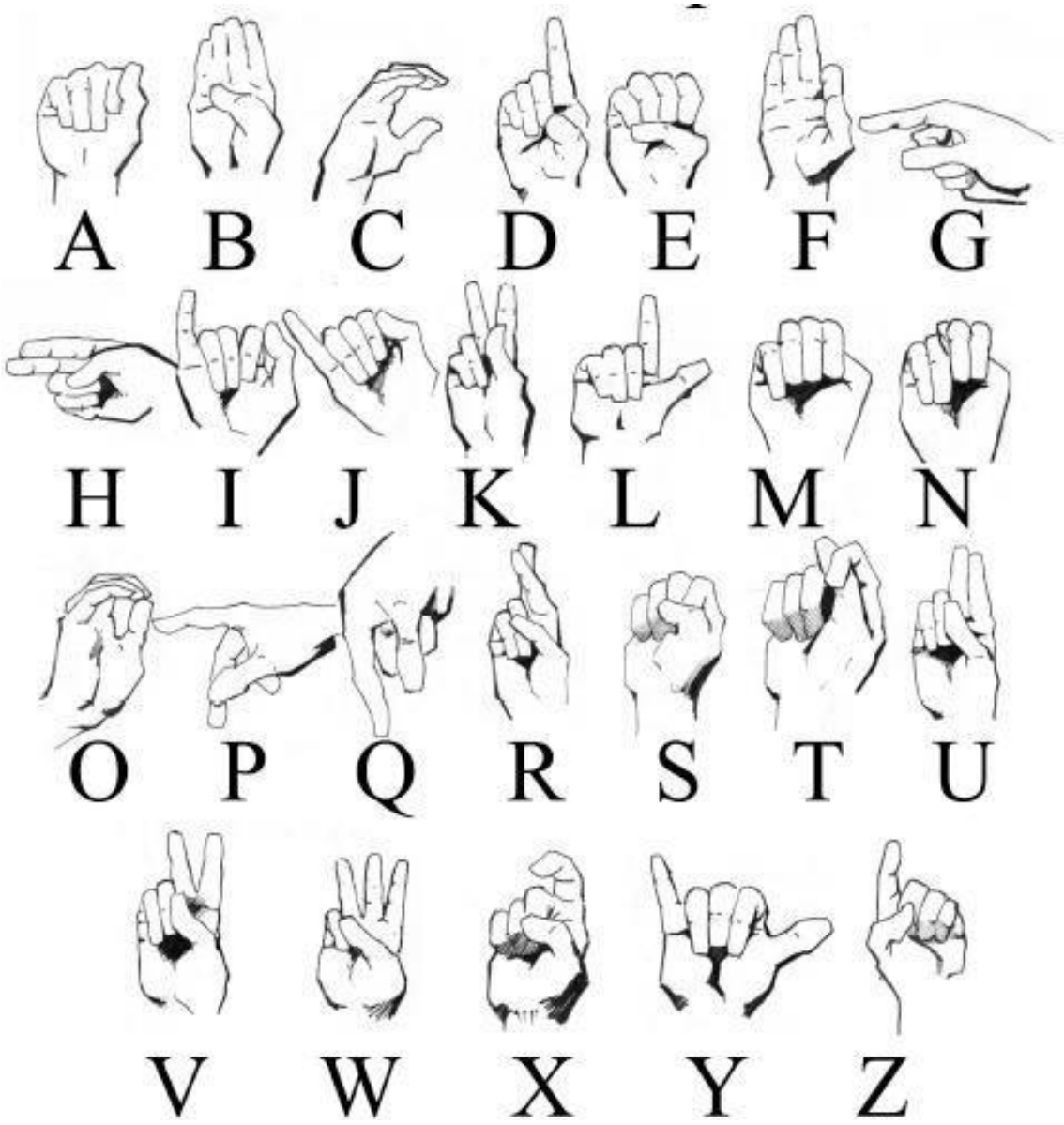


Figure 1-1: American Sign Language Gestures

1.2 Background and Motivation

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

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

The aim is to develop a user friendly human computer interfaces (HCI) where the computer understands the human sign language. There are various sign

languages all over the world, namely American Sign Language (ASL), French Sign Language, British Sign Language (BSL), Indian Sign language, Japanese Sign Language and work has been done on other languages all around the world.

1.3 Problem Statement and Objectives

Speech impaired people use hand signs and gestures to communicate. They face problems in communicating with people around them. Their family members and close friends learn sign language to communicate with them but other than that, normally, people do not know sign language and don't even try to learn, as they do not need it in their day-to-day life.

The society does not understand how difficult it is for them to communicate with someone who does not know sign language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people.

By using technologies such as image processing and machine learning one can build a system which converts sign language to text or speech. These systems can be of great help to dumb people as they can easily communicate with anyone using such system. So that, it bridges the gap between physically challenged people and normal people.

The objective of the Sign Language recognition project is to successfully create a sign detector, which will detect alphabets from A to Z that can very easily be extended to cover a vast multitude of other signs and hand gestures including numbers also. Basically the aim of the project is to build a machine learning model that can accurately and efficiently be able to classify various hand gestures used for fingerspelling in sign language. And also learn how to use OpenCV for Real-time detection and Tensorflow/Keras to build a Model with high accuracy. The project aims to aid individuals with inherent or acquired hearing disability facilitate real-time communication with deaf with high accuracy. We will develop this project using OpenCV and Keras modules of python. Main Objective of this Project is to create a high accuracy classification model which should give accurate predictions in training data as well as in testing data.

1.4 Scope of the Project

1. Since deaf people are usually deprived of normal communication with otherpeople, they have to rely on an interpreter or some visual communication. Now the interpreter can not always be available, so this project can help eliminate the dependency on the interpreter.
2. The system can be extended to incorporate the knowledge of standard signlanguage.
3. A mobile and web based version of the application will increase the reach tomore people.
4. Integrating hand gesture recognition system using computer vision forestablishing 2-way communication system.

1.5 Team Organization

- **Sarthak Sharma:**

Along with doing preliminary investigation and understanding the limitations of current system, I studied about the topic and its scope and surveyed various research papers related to the object detection and the technology that is to be used. I also worked on the implementation of tensorflow framework and the working of counting of objects in the project.

- **Preet Kaur Nagi :**

I investigated and found the right technology and studied in deep about it. For the implementation of the project , I collected the object data and trained the model for it. Implementation logic for the project objective and coding of internal functionalities is also done by me. Documentation is also a part of the work done by me in this project.

- **Rahul Ahuja:**

I contributed in completing individual deliverables and also documented theprocess. Assessed various alternative solutions to make sure they are effective. I also approved all changes to the project scope by verifying that

project deliverables meet the requirements or not. Performed Testing over above mentioned solutions to validate objectives.

- **Poorti Rajani:**

I helped in determining the methodology used on the project. And henceforth took the implementation with that respect.
Worked on Python libraries and its real time applications.
I established a precise project schedule by determining each phase in the concern.

All four of us divided equal modules of project for ensuring timely completion.

1.6 Report Structure

The project ***Real-time Sign Language Detection and Recognition*** is primarily concerned with creating an stable interface between normal and Specially disabled people so as to ensure effective communication among both the parties.
The whole project report is categorized into five chapters.

Chapter 1: Introduction- introduces the background of the problem followed by rationale for the project undertaken. The chapter describes the objectives, scope and applications of the project. Further, the chapter gives the details of team members and their contribution in development of project which is then subsequently ended with report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of existing system and highlights the issues and challenges of project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System - starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrate software engineering paradigm used along with different design representation. The chapter also includes block diagram and details of major modules of the project. Chapter also gives insights of different type of feasibility study carried out for the

project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation - includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interface designed in project along with their functionality. Further it discuss the experiment results along with testing of the project. The chapter ends with evaluation of project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion - Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2 . Review of Literature

Review of Literature

Sign Language Detection and Recognition is being studied for more than four decades now and significant efforts have been paid to develop representation schemes and algorithms aiming at recognizing gestures in images taken under different imaging conditions. A real-time sign language translator is an important milestone in facilitating communication between the deaf community and the general public. We hereby present the development and implementation of an American Sign Language (ASL) fingerspelling translator based on a convolutional neural network.

Within a limited scope of distinct objects in detection and recognition, such as handwritten digits, fingerprints, faces, gestures and static figures substantial success has been achieved. In addition, significant progress towards real-time application from images has been made in the recent years.

2.1. Preliminary Investigation

2.1.1 Current System

We have gone through other similar works that are implemented in the domain of sign language recognition. The summaries and drawbacks of each of the project works are mentioned below:

A Survey of Hand Gesture Recognition Methods in Sign Language Recognition: Issues were faced relating to the accuracy of the model. The accuracy achieved was roughly estimated to be around 81.64%, which didn't match the required criterion so as to abide by the standard characteristics of detecting and pre-processing image dataset.

Communication between Deaf-Dumb People and Normal People:

They build are cognition system for Flemish sign language using convolutional

neural networks and achieve an error rate of 2.5%. It is a non-real-time application that assures fluctuating accuracy due to use of background subtraction algorithms. Sign language recognition using a combination of new vision based features: are cognition model is built using hidden Markov model classifier and a vocabulary of 30 words and they achieve an error rate of 10.90.

Over the past two decades, researchers have used classifiers from a variety of categories that we can group roughly into linear classifiers, neural networks and Bayesian networks .While linear classifiers are easy to work with because they are relatively simple models, they require sophisticated feature extraction and preprocessing methods to be successful.

2.1.2 Limitations of Current System

The limitations of these are as follows:

- At the economic front, it is not feasible as choosing one method to be focused on, tends to make other method that may be better suit for Sign Language Recognition, not being tested. Trying out other methods makes researchers barely develops one method to its fullest potentials.
- Fair and direct comparison between approaches are limited because of the variation of sign language in different countries and the difference in limitation set by each researcher. Variation of sign language in most of the country is based on their grammar and their way to present each word, such as presenting the languageby word or by sentence.
- The problems of developing sign language recognition ranges from the image acquisition to the classification process. Researchers are still finding the best method for the image acquisition. Gathering images using camera gives the difficulties of image pre-processing.
- Meanwhile, using active sensor device can be costly. Classification methods also give researchers some drawbacks. Wide choice of recognition method makesresearchers unable to focus on one best method.

2.2 Requirement Identification and Analysis for Project

Significant work has been done in the field of Sign Language Detection and Recognition, however, it is not easy to achieve desired results and also to perform it in real-time approach. The review of literature leads to draw certain major findings which areas under:

- The study brought out that moving targets are extracted from video- streaming and they are classified according to the pre defined categories. Objects are detected by using pixel wise difference between consecutive frames and they are categorized mainly as- humans, vehicles and background clutter. They are then tracked by template matching. There are two key elements which make the system robust - the classification system which is based on temporal consistency and the tracking system based on a combination of temporal differencing and correlation matching.
- The paper presents a new approach to tracking of non-rigid objects which is based on features like color, texture. It is appropriate for a variety of objects with different color patterns. The mean shift iterations are employed to find the target candidate that is most similar to target model and this similarity is based on a metric which in turn, is based on Bhattacharya coefficient.
- The paper focused on the problem of learning to detect signals from a small training database. The performance depends crucially on the features that are used to represent the signs. Specifically, it shows that using local edge orientation histograms (EOH) as features can significantly improve performance compared to the standard linear features used in existing systems. For frontal faces, local orientation histograms enable state of the art performance using only a few hundred training examples. For profile view faces, local orientation histograms enable learning a system that seems to outperform the state of the art in real-time systems even with a small number of training examples.
- The paper advocated the use of randomized trees as the classification technique. It is both fast enough for real-time performance and more robust. It also gives us a principled way not only to match key points but to select during a training phase those that are the most recognizable ones. It is robust to

illuminations changes, scale changes and occlusions. The work presents a real time system for multiple objects tracking in dynamic scenes. A unique characteristic of the system is its ability to cope with long duration and complete occlusion without a prior knowledge about the shape or motion of objects. The system produces good segment and tracking results at a frame rate of 15-20 fps for image size of 320x240, as demonstrated by extensive experiments performed using video sequences under different conditions indoor and outdoor with long-duration and complete occlusions in changing background.

- The work presented a real time robust human detection and tracking system for video surveillance which can be used in varying environments. This system consists of human detection, human tracking and false sign detection. The human detection utilizes the background subtraction to segment the blob and use codebook to classify different languages such as American, Indian, Flemish etc. The optimal design algorithm of the codebook is proposed. Bayesian networks like Hidden Markov Models have also achieved high accuracies. These are particularly good at capturing temporal patterns, but they require clearly defined models that are defined prior to learning. Starner and Pentland used a Hidden Markov Model (HMM) and a 3-D glove that tracks hand movement. Since the glove is able to obtain 3-D information from the hand regardless of spatial orientation, they were able to achieve an impressive accuracy of 99.2% on the test set. Their HMM uses time series data to track hand movements and classify based on where the hand has been in recent frames.
- The paper describes a two level approach to solve the problem of real- time vision- based hand gesture classification. The lower level of the approach implements the posture recognition with Haar-like features and the Ada-Boost learning algorithm. With this algorithm, real-time performance and high recognition accuracy can be obtained. The higher level implements the linguistic hand gesture recognition using a context-free grammar-based syntactic analysis. Given an input gesture, based on the extracted postures, the composite gestures can be parsed and recognized with a set of primitives and production rules.

- The paper brings in new techniques to detect and analyze periodic motion as seen from both a static and a moving camera. By tracking objects of interest, it computes a self-similarity measure as it evolves in time. For periodic motion, the self-similarity measure is also periodic and we apply Time- Frequency analysis to detect and characterize the periodic motion. The periodicity is also analyzed robustly using the 2D lattice structures inherent in similarity matrices.
- This paper presents a general, trainable system for real-time detection in unconstrained, cluttered scenes. The system derives much of its power from a representation that describes an class. It quantifies how the representation affects detection performance by considering several alternate representations including pixels and principal components. Paper proposes new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information.

2.2.1 Conclusion

This chapter reviews the literature surveys that have been done during the research work. The related work that has been proposed by many researchers has been discussed. The research papers related to Sign language detection and recognition of objects from 1980 to 2019 have been shown which discussed about different methods and algorithm to identify sign language gestures.

Chapter 3 . Proposed System

Proposed System

3.1 The Proposal

In our Project, we will be creating a real-time automatic sign language gesture recognition system, using different tools (Deep Learning, OpenCV, and Tensorflow), which will be helpful for deaf and disabled people. We will be creating our own dataset which uses rawimages to match our requirements.

While other systems used MNIST dataset for implementation of this project that affects the accuracy of the model. Most of the existing systems use Kinect devices but our main aim is to create a project which can be used with readily available resources. A sensor like Kinect not only isn't readily available but also is expensive for most of audience to buy and our model uses a normal webcam of the laptop hence it is great plus point.

3.2 Benefits of the Proposed System

The current system had a lot of challenges that are overcome by this system:

- **Economic :** The proposed system is economic as there will not be any incorrectly determined sign, gestures pertaining to American sign language alphabets dataset.
- **Convenient :** This project uses self created dataset, hence it would yield out to be more convenient and compatible to work on various platforms. ASL signs are recorded individually and not in a random manner to avoid complicated scenarios while implementing it altogether.
- **Real-Time Observation :** as the system works on real-time scale, hence accuracy is unaltered and serves maximum purpose for our project theme.

- **Statistical analysis :** The number of signs and gestures can be examined individually and kepta record for calculating various factors.

3.3 Block Diagram

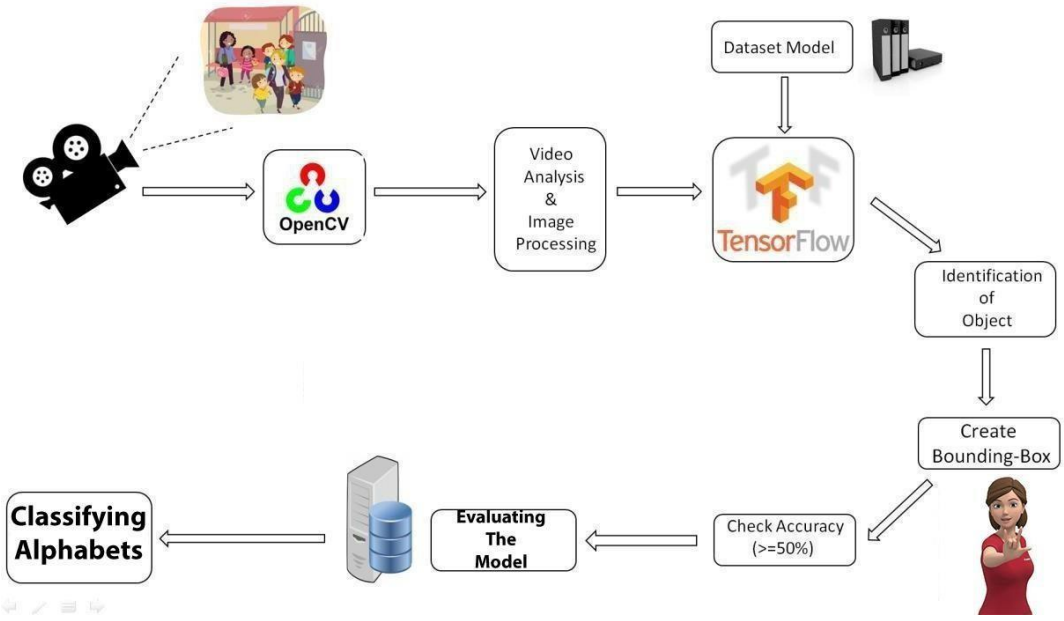


Figure 3-1: Block Diagram

3.4 Feasibility Study

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.4.1 Technical

For any real-time detection system, there is a need to process images from the video. For this, the kind of framework used must be the one that is capable of extracting those objects from the images easily and accurately in real-time. The framework used in this is Tensorflow, which is a framework designed by Google for efficiently dealing with deep learning and concepts like neural networks , making the system technically feasible.

The system once set up completely, works automatically without needing any person

to operate it. The result (count and other information), gets automatically saved in the database, without requiring any manual effort for saving it. For making the system technically feasible, there is a requirement of GPU built system with high processor for better performance.

3.4.2 Economical

For any real-time detection system, there is a need of a High definition Camera for better and accurate results.

Since the system is completely automated, there is a need of continuous electricity supply for it to operate.

The Tensorflow framework used in the system works great with GPU built systems, whichare a little on the expensive side.

Since the system uses high performance processors continuously, so to save any disaster from occurring due to very high temperatures, there is a requirement of a cooling system inthe environment where it is implemented.

3.4.3 Operational

We aim to produce a robust model that consistently classifies letters correctly with first- time users and another that correctly classifies ASL letters in a majority of cases. Given the limitations of the datasets and the encouraging results achieved, we are confident that with further research and more data, we can produce a fully generalizable translator for all ASL letters.

3.5 Design Representation

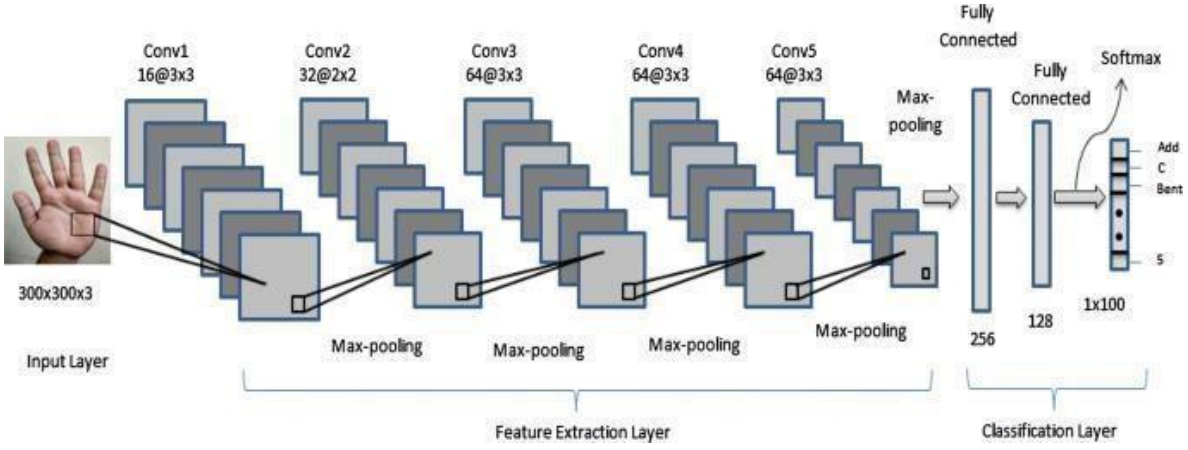


Figure 3-2: CNN Architecture

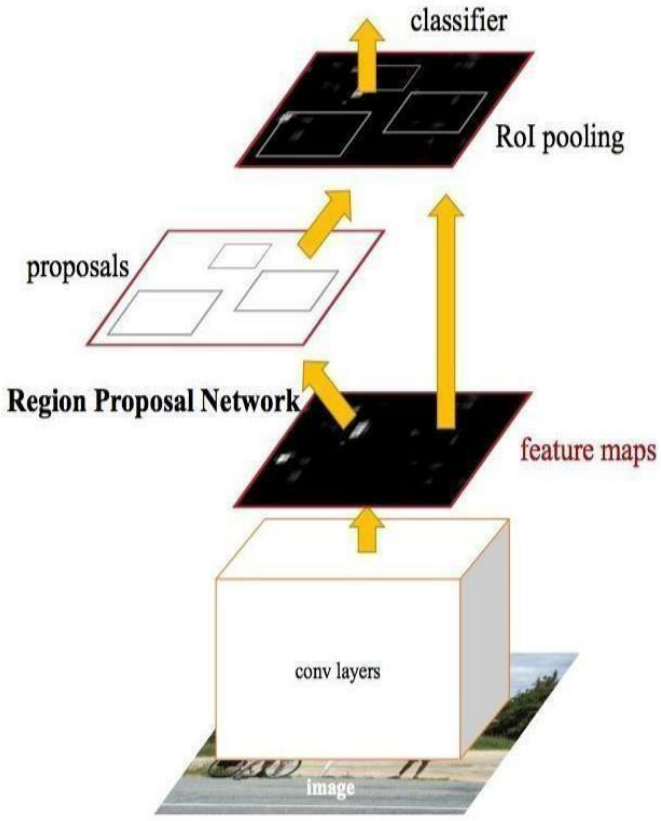


Figure 3-3: Faster CNN Architecture

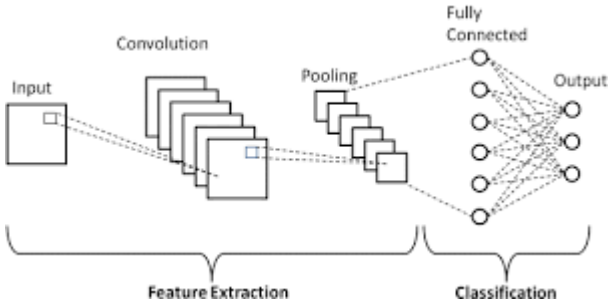


Figure 3-4: CNN Model

3.5.1 Data Flow Diagrams

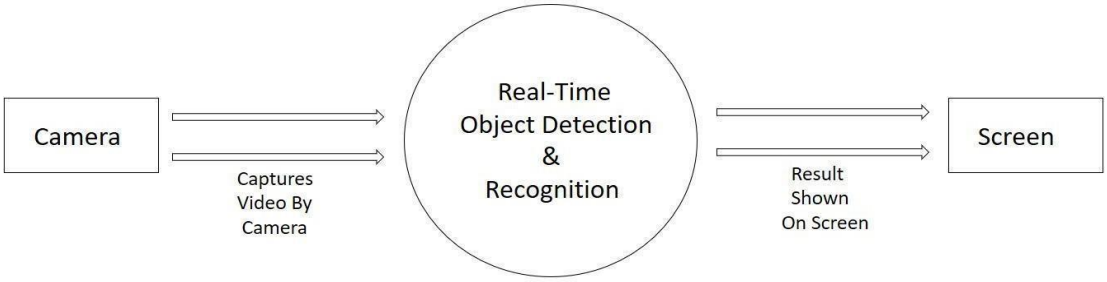


Figure 3-5 Data Flow Diagram Level 0

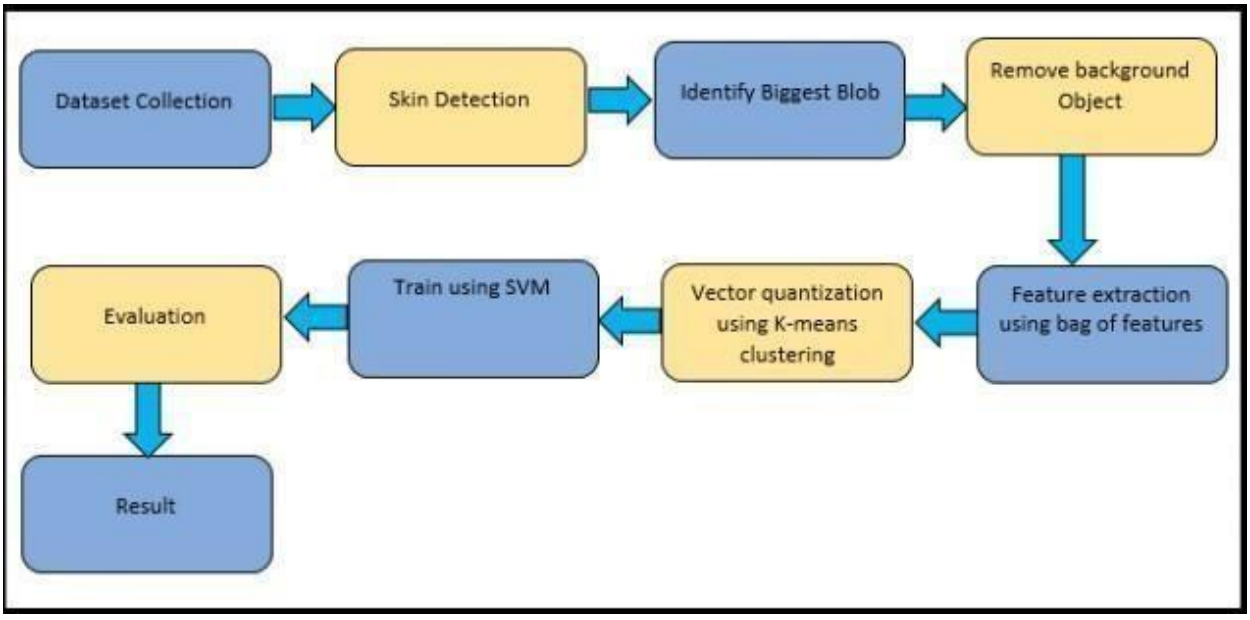


Figure 3-6 Data Flow Diagram Level 1

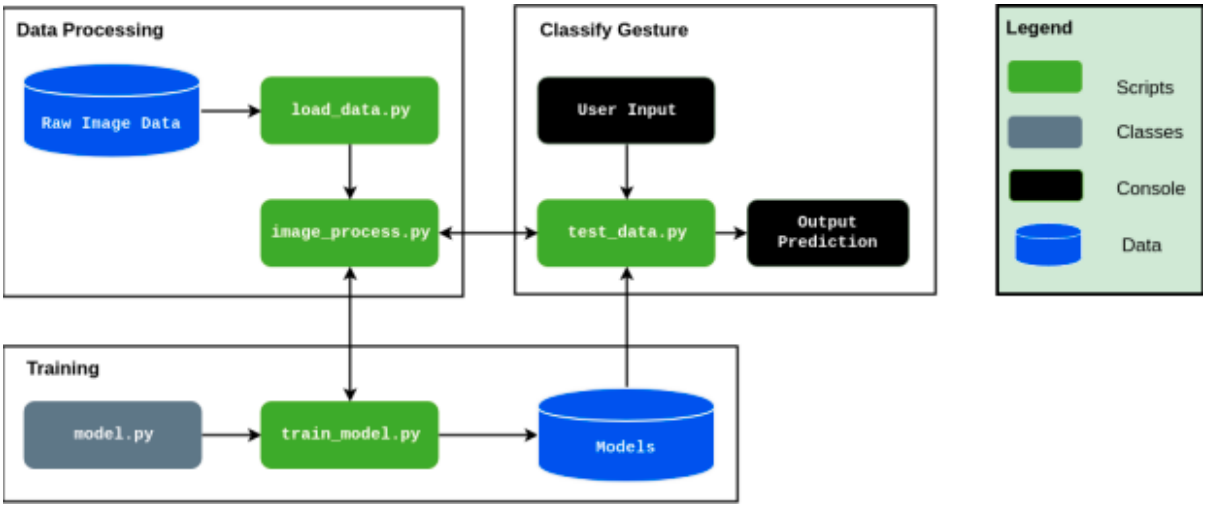


Figure 3-7 Data Flow Diagram Level 2

3.6 Deployment Requirements

There are various requirements (hardware, software and services) to successfully deploy the system. These are mentioned below:

3.6.1 Hardware

- 32-bit, x86 Processing system Windows 7 or later operating system
- High processing computer system without GPU or with GPU(high performance)
- High- definition Camera

3.6.2 Software

- OpenCV
- Python and its supported libraries
- Tensor Flow
- If Installing Tensorflow in GPU systems :
    1. CUDA® Toolkit 9.0.
    2. The NVIDIA drivers associated with CUDA Toolkit 9.0. cuDNN v7.0.
    3. GPU card with CUDA Compute Capability 3.0 or higher

Chapter 4 . Implementation

Implementation

The system is a vision based approach. All the signs are represented with bare hands and so iteliminates the problem of using any artificial devices for interaction.

4.1 Data Set Generation

For the project we tried to find already made datasets but we couldn't find dataset in the form of raw images that matched our requirements. All we could find were the datasets in the form of RGB values. Hence we decided to create our own data set. Steps we followed to create our data set are as follows.

We used Open computer vision (OpenCV) library in order to produce our dataset. Firstly we captured around 800 images of each of the symbol in ASL for training purposes and around 200 images per symbol for testing purpose.

First we capture each frame shown by the webcam of our machine. In the each frame we define a region of interest (ROI) which is denoted by a blue bounded square as shown in the image below.

From this whole image we extract our ROI which is RGB and convert it into gray scale Imageas shown below.

Figure 4-1: Hand Image Captured by Web Cam

Finally we apply our Gaussian blur filter to our image which helps us extracting various features of our image. The image after applying Gaussian blur looks like below.

Figure 4-2: Captured Image Converted to Gaussian Image

4.2 GESTURE CLASSIFICATION

The approach which we used for this project is :
Our approach uses two layers of algorithm to predict the final symbol of the user.

4.2.1 Algorithm Layer 1:

1. Apply gaussian blur filter and threshold to the frame taken with opencv to getthe processed image after feature extraction.
2. This processed image is passed to the CNN model for prediction and if a letter is detected for more than 50 frames then the letter is printed and takeninto consideration for forming the word.
3. Space between the words are considered using the blank symbol.

4.2.2 Algorithm Layer 2:

1. We detect various sets of symbols which show similar results on gettingdetected.
2. We then classify between those sets using classifiers made for those sets only.

Layer1

4.3 CNN:

1st Convolution Layer: The input picture has resolution of 128x128 pixels. It is first processed inthe first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126pixel image, one for each Filter-weights.

1. **1st Pooling Layer :** The pictures are down sampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels.
2. **2nd Convolution Layer :**

   Now, these 63 x 63 from the output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each).This will result in a 60 x 60 pixel image.
3. **2nd Pooling Layer :**

   The resulting images are down sampled again using max pool of 2x2 and is reducedto 30 x 30 resolution of images.
4. **1st Densely Connected Layer :**

   Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.
5. **2nd Densely Connected Layer :**

   Now the output from the 1st Densely Connected Layer are used as an input to a fully connected layer with 96 neurons.
6. **Final layer:**

   The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 31, 31, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 64)                294976
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_3 (Dense)              (None, 26)                3354
=================================================================
Total params: 416,410
Trainable params: 416,410
Non-trainable params: 0
```

Figure 4-3: Our Model Architecture

4.4 Activation Function :

We have used ReLu (Rectified Linear Unit) in each of the layers(convolutional as well as fully connected neurons).

ReLu calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

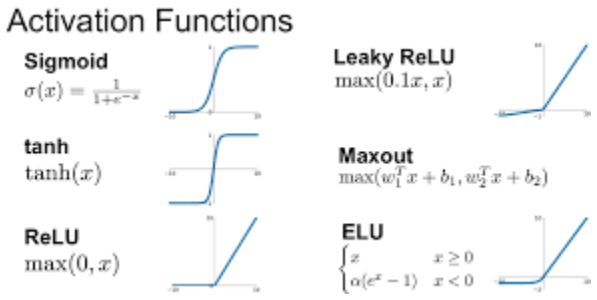


Figure 4-4: Activation Function

4.5 Pooling Layer :

We apply **Max** pooling to the input image with a pool size of (2, 2) with relu activation function. This reduces the amount of parameters thus lessening the computation cost andreduces overfitting.

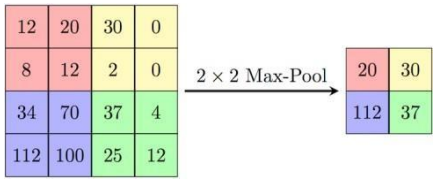


Figure 4-5: Pooling Layer

4.6 Dropout

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples.This layer "drops out" a random set of activations in that layer by setting them to zero.The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out.

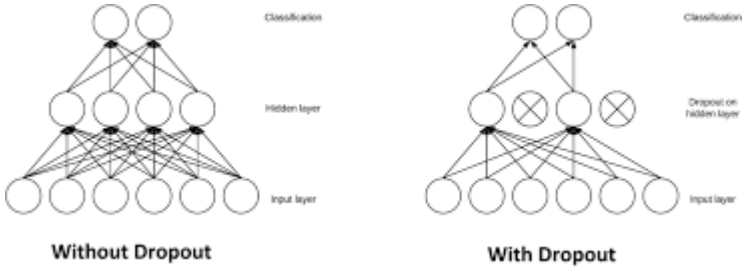


Figure 4-6: Dropout Layers

4.7 Optimizer:

We have used Adam optimizer for updating the model in response to the output of the loss function. Adam combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm(ADA GRAD) and root mean square propagation(RMSProp).

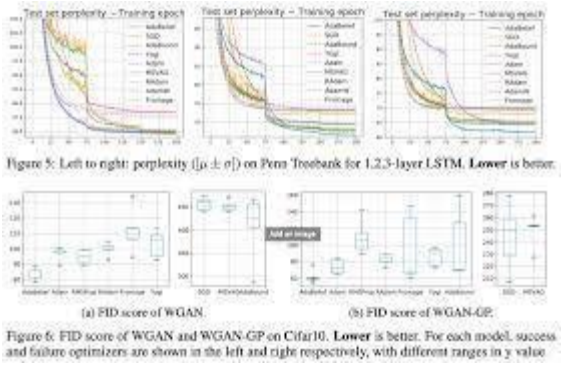


Figure 4-7: Optimizer

4.8 Finger spelling sentence formation Implementation :

1. Whenever the count of a letter detected exceeds a specific value and no other letter is close to itby a threshold we print the letter and add it to the current string(In our code we kept the valueas 50 and difference threshold as 20).

2. Otherwise we clear the current dictionary which has the count of detections of present symbol toavoid the probability of a wrong letter getting predicted.

3. Whenever the count of a blank (plain background) detected exceeds a specific value and if thecurrent buffer is empty no spaces are detected.

4. In other case it predicts the end of word by printing a space and thecurrent gets appended to the sentence below.

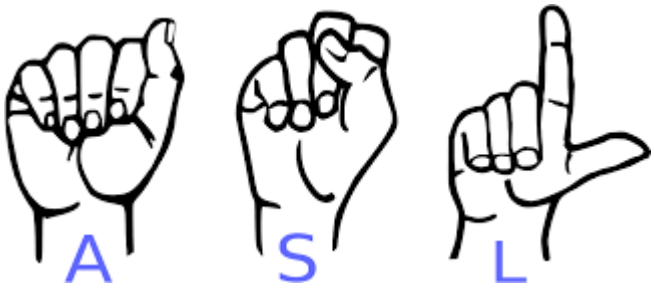


Figure 4-8: Alphabets Hand Gesture

4.9 Autocorrect Feature :

A python library **Hunspell_suggest** is used to suggest correct alternatives for each (incorrect) input word and we display a set of words matching the current word in which the user can select a word to append it to the current sentence. This helps in reducing mistakescommitted in spellings and assists in predicting complex words.

4.10 Training and Testing :

We convert our input images(RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 128 x 128.

We feed the input images after preprocessing to our model for training and testing after applying all theoperations mentioned above.

The prediction layer estimates how likely the image will fall under one of the classes. So the output is normalized between 0 and 1 and such that the sum of each values in each class sumsto 1. We have achieved this using softmax function.

At first the output of the prediction layer will be somewhat far from the actual value. To make it better we have trained the networks using labeled data. The cross-entropy is a performance measurement used in the classification. It is a continuous function which is positive at values which is not same as labeled value and is zero exactly when it is equal to the labeled value. Therefore we optimized the cross-entropy by minimizing it as close to zero. To do this in our network layerwe adjust the weights of our neural networks.
TensorFlow has an inbuilt function to calculate the cross entropy.

As we have found out the cross entropy function, we have optimized it using Gradient Descent in fact with the best gradient descent optimizer is called AdamOptimize.

This is our model Accuracy and Loss Graph:

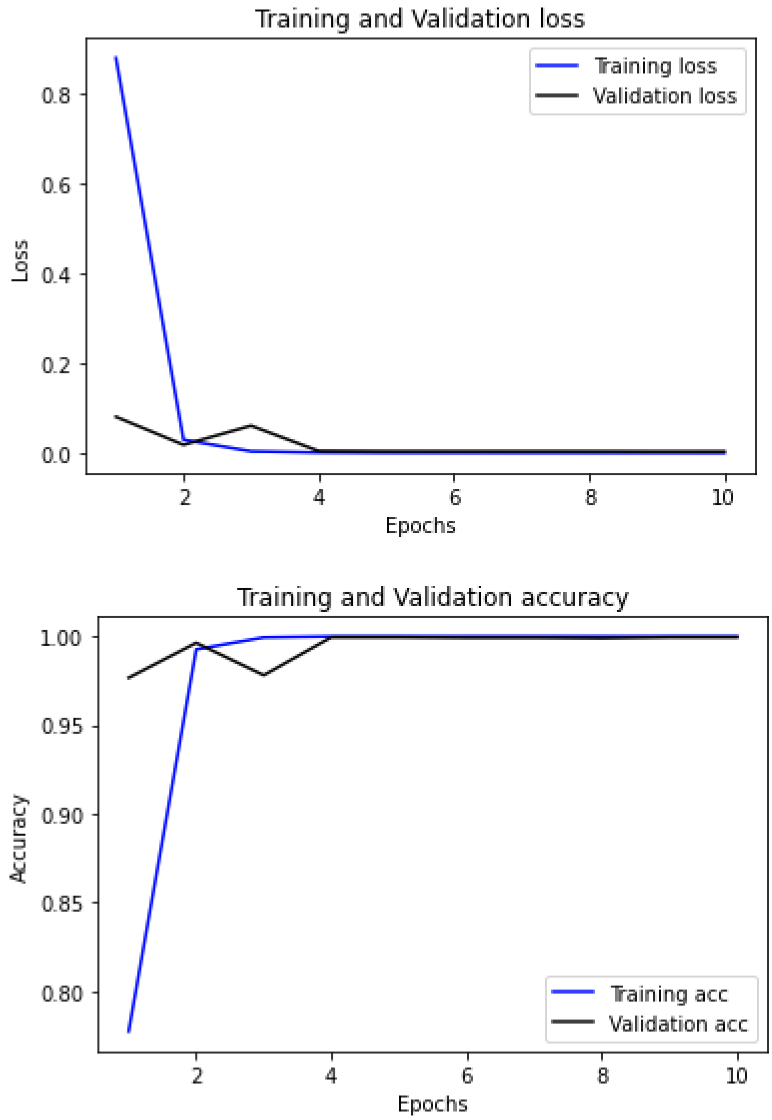


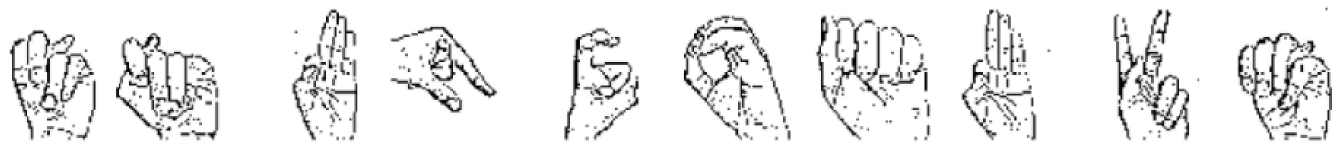
Figure 4-9: Loss & Accuracy Graph

Evaluation of our Model:

```
predictions on a small set of test data--

N   T   F   Q   X   O   A   F   K   M

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



```
Actual labels
N   T   F   Q   X   O   A   F   K   M   (10, 64, 64, 3)
```

Figure 4-10: Testing of Model

Chapter 5. Conclusion

Conclusion

5.1 Conclusion

In this report, a functional real time vision based American sign language recognition for D&M people have been developed for asl alphabets. We achieved final accuracy of 98.0%-99.0% on our dataset. We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other.

This way we are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate

```
Epoch 1/10
1238/1238 [==============================] - 391s 313ms/step - loss: 0.8798 - accuracy: 0.7773 - val_loss: 0.0809 - val_accu
racy: 0.9767
Epoch 2/10
1238/1238 [==============================] - 246s 199ms/step - loss: 0.0298 - accuracy: 0.9926 - val_loss: 0.0185 - val_accu
racy: 0.9964
Epoch 3/10
1238/1238 [==============================] - 117s 94ms/step - loss: 0.0041 - accuracy: 0.9994 - val_loss: 0.0610 - val_accur
acy: 0.9781
Epoch 4/10
1238/1238 [==============================] - 94s 76ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0047 - val_accura
cy: 0.9995
Epoch 5/10
1238/1238 [==============================] - 97s 79ms/step - loss: 6.8857e-04 - accuracy: 1.0000 - val_loss: 0.0041 - val_ac
curacy: 0.9995
Epoch 6/10
1238/1238 [==============================] - 102s 83ms/step - loss: 5.6559e-04 - accuracy: 1.0000 - val_loss: 0.0040 - val_a
ccuracy: 0.9993
Epoch 7/10
1238/1238 [==============================] - 102s 82ms/step - loss: 4.8434e-04 - accuracy: 1.0000 - val_loss: 0.0040 - val_a
ccuracy: 0.9993
Epoch 8/10
1238/1238 [==============================] - 170s 137ms/step - loss: 4.2691e-04 - accuracy: 1.0000 - val_loss: 0.0039 - val_
accuracy: 0.9990
Epoch 9/10
1238/1238 [==============================] - 274s 221ms/step - loss: 3.8067e-04 - accuracy: 1.0000 - val_loss: 0.0037 - val_
accuracy: 0.9995
Epoch 10/10
1238/1238 [==============================] - 237s 191ms/step - loss: 3.4302e-04 - accuracy: 1.0000 - val_loss: 0.0037 - val_
accuracy: 0.9995
```

Figure 5-1: Model Accuracy

5.2 Limitations of the Work

- The biggest challenge for us was to find the right dataset. Some

Datasets had very few images while others had only a few symbols.

- Choosing the correct dataset-model combination was tricky as there were models whichtended to overfit to a particular dataset.
- The next challenge faced was modifying the Inception v3 model for our use, since it isrelatively new, not many references was available online.
- Also we first started with the Indian Sign Language, but due to the lack of dataset wehad to switch to American Sign Language.

5.3 Suggestion and Recommendations for Future Work

Future scope includes but is not limited to:

- The implementation of our model for other sign languages such as Indian SignLanguage, as it is currently limited to American Sign Language
- Enhancement of the model to recognize common words and expressions.
- Further training the neural network to efficiently recognize symbols involving twohands.
- Improving the results by using linear classifiers.
- Including dynamic hand gestures in addition to the current static fingerspelling
- Using convolutional neural networks to also account for depth data in order to detectgestures captured by devices such as Kinect.

Output Images:


V


C


O

Bibliography

[1] H. Fujiyoshi, A. J. Lipton and R. S. Patil, "Moving Target Classification andTracking from Real-time Video," in *Fourth IEEE workshop*, 1998.

[2] D. Comaniciu, R. Vishwanathan and P. Meer, "Real-Time Tracking of Non-RigidObjects using Mean Shift," in *IEEE*, 2000.

[3] K. Levi and Y. Weiss, "Learning Object Detection from a Small Number of Examples: the Importance of Good Features," in *IEEE Computer Society Conference*, 2004.

[4] V. Lepetit, P. Lagger and P. Fua, "Randomized Trees for Real-Time KeypointRecognition," in *IEEE Computer Society Conference*, 2005.

[5] T. Yang, S. Z. Li and J. Li, "Real-time Multiple Objects Tracking with Occlusion Handling in Dynamic Scenes," in *IEEE Computer Society Conference*, 2005.

[6] J. Zhou and J. Hoang, "Real Time Robust Human Detection and Tracking System,"in *IEEE Computer Society Conference*, 2005.

[7] C. P. Papageorgiou, M. Oren and T. Poggio, "A General Framework for Object Detection," in *Center for Biological and Computational Learning Artificial Intelligence Laboratory*, Cambridge, 2005.

[8] R. D. Charette and F. Nashashibi, "Real TimeVisual Traffic Lights Recognition Based on Spot Light Detection andAdaptive Traffic Lights Template," in *IEEE*, 2009.

[9] S. K. Nayar, S. A. Nene and M. Hiroshi, "Real-Time 100 Object Recognition System," in *IEEE International Conference*, 1996.

[10] A. Adam, E. Rivlin, S. Ilan and D. Reinitz, "Robust Real-Time Unusual Event Detection Using Multiple Fixed-Location Monitors," in *IEEE*, 2008.

[11] C. Bahlmann, Y. Zhu, R. Vishwanathan, M. Pelkoffer and T. Koehler, "A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information," in *IEEE*, 2005.

[12] M. Betke, E. Haritaoglu and L. S. Davis, "Real-time multiple vehicle detection and tracking from a moving vehicle," *Machine Vision and Applications,* p. 12, 2000.

[13] Q. Chen, N. D. Georganas and E. M. Petriu, "Real-time Vision-based Hand Gesture Recognition Using Haar-like Features," in *IEEE Conference*, 2007.

[14] R. Cutler and L. S. Davis, "Robust Real-Time Periodic Motion Detection, Analysis, and Applications," *IEEE transactions on Pattern Analysis and Machine Intelligence,* 2000.

[15] J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," *Image and Vision Computing,* 2004

[16] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition," in *IEEE International Conference*, 2015.

[17] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *Internation Journal of Computer Vision,* 2000.

[18] J. Redmon, S. Divvala, R. Girshik and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE conference on Computer Vision andPattern Rocgnition*, 2016.

[19] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," *Computer Vision and Pattern Recognition,* 2011.

[20] K. P. A. Menon, "Management of Agriculture and Rural Development Programme: Problems and Prospects, India," *J Extension Education,* vol. 21, no. 1, pp. 94-98, 1985.

Guide Interaction Sheet

Date	Discussion	Action Plan
08/04/2022	Discussed about the centralised idea of our project with our Project Guide. Prof. Narendra Pal Rathore.	Real-time sign language detection and Recognition was decided as the title.
08/04/2022	Discussion on the technology to be used for Sign Language detection in real-time.	Deep learning, Tensorflow , OpenCV and other tools were finalized
12/04/2022	Discussion of the creation of synopsis of the project.	Gathering of information for synopsis creation and approved it from our Project supervisor.
13/04/2022	Suggestions on how to do a literature survey and preliminary investigation on the topic.	Many research papers were read , understood and their abstract were to be written.
20/04/2022	Digital Poster creation along with Animated video.	Divided tasks within out team members and worked in a flow to ensure timely completion. Approved both poster and video from our project guide.
23/04/2022	Discussion on the implementation of the project.	Elaborated the entire workflow along with the implementation status.
23/04/2022	Worked on project Research paper	Referred various UGC journals to get familiar with the format. Then did various Research related to our project so as to complete it upto the mark.
23/04/2022	Discussion on project documentation	Decided to write the content and integrate it in the proper format of the report

Source Code

1.) Dataset Creation Code:

Dataset_Creation_File.py

```
#importing required libraries:

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

import warnings
import numpy as np
import cv2
warnings.simplefilter(action='ignore', category=FutureWarning)

# declaring variables:
background = None
accumulated_weight = 0.5
```

```
# Setting the dimensions of ROI:
ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350

#Creating function to calculate average accumulate weight:
def cal_avg_acc_weight(frame,accumulated_weight):
    global background
    if background is None:
        background = frame.copy().astype("float")
        return None
    cv2.accumulateWeighted(frame, background, accumulated_weight)

#Creating a function to calculate threshold value of each frame:
def find_hand(frame, threshold=25):
    global background
    difference = cv2.absdiff(background.astype("uint8"),frame)
    _ , threshold_value = cv2.threshold(difference,threshold,255,cv2.THRESH_BINARY)

    #grabing the external counters for the image:
    contours, hierarchy = cv2.findContours(threshold_value.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    if len(contours)==0:
        return None
```

```
    else:
        max_hand_prob = max(contours,key=cv2.contourArea)
        return (threshold_value,max_hand_prob)

# Main Code for Capturing images from Webcam and saving it in a particular folder:

camera = cv2.VideoCapture(0)
no_of_frames = 0
element = 'Z'
no_of_img_taken = 0

while True:
    ret,frame = camera.read()
    # flipping the frame to prevent inverted image of captured frame:
    frame = cv2.flip(frame,1)
    copy_frame = frame.copy()
    region_of_interest = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]
    gray_frame = cv2.cvtColor(region_of_interest, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_frame,(9,9),0)
    if no_of_frames<60:
        cal_avg_acc_weight(gray_frame,accumulated_weight)
        if no_of_frames<=59:
            cv2.putText(copy_frame, 'Fetching Background, Get Ready!!', (80,400), cv2.FONT_HERSHEY_SIMPLEX,0.9,(0,255,0),2)
    elif no_of_frames<=150:
        hand = find_hand(gray_frame)
```

```
        cv2.putText(copy_frame,'Create Gesture for'+str(element),(200,400),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),2)

        if hand is not None:
            threshold_value, hand_segment = hand
            cv2.drawContours(copy_frame,[hand_segment+(ROI_right,ROI_top)],-1,(255,0,0),1)

cv2.putText(copy_frame,str(no_of_frames)+"For"+str(element),(70,45),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
            cv2.imshow('Threshold Image',threshold_value)
    else:
        hand = find_hand(gray_frame)
        if hand is not None:

            # unpack the thresholded img and the max_contour...
            threshold_value, hand_segment = hand
            # Drawing contours around hand segment
            cv2.drawContours(copy_frame, [hand_segment + (ROI_right,
            ROI_top)], -1, (255, 0, 0),1)

            cv2.putText(copy_frame, str(no_of_frames), (70, 45),cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
            cv2.putText(copy_frame, str(no_of_img_taken) + 'images' +"For" + str(element), (200, 400), cv2.FONT_HERSHEY_SIMPLEX, 1,(0,0,255), 2)
            cv2.imshow("Threshold Image", threshold_value)
            # Displaying the thresholded image
            cv2.imshow("Thresholded Hand Image", threshold_value)
```

```
            if no_of_img_taken <= 40:

cv2.imwrite(r"C:\\Users\\KRISHNA\\Sign_Language_Detection\\Dataset\\test\\"+
str(element)+"\\"+str(no_of_img_taken+300)+'.jpg',threshold_value)
            else:
                break
            no_of_img_taken += 1

        else:
            cv2.putText(copy_frame,'No                                    hands
detected',(200,400),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)
    # Drawing ROI on frame copy

cv2.rectangle(copy_frame,(ROI_left,ROI_top),(ROI_right,ROI_bottom),(255,128,
0),3)
    cv2.putText(copy_frame,"Sign                                    Language
Recognition",(10,20),cv2.FONT_ITALIC,0.5,(51,255,51),1)

    no_of_frames+=1
    cv2.imshow('Sign Detection',copy_frame)
    k = cv2.waitKey(1) & 0xFF
    if k==27:
        break

# Releasing the camera & destroying all the windows:
cv2.destroyAllWindows()
camera.release()
```

2.) Model Creation for dataset:

Model_Creation.py

```
#importing required libraries:

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten,
BatchNormalization, Conv2D, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

import warnings
import numpy as np
import cv2
warnings.simplefilter(action='ignore', category=FutureWarning)

train_path = r'C:\Users\KRISHNA\Sign_Language_Detection\Sign_Gestures\train'
test_path = r'C:\Users\KRISHNA\Sign_Language_Detection\Sign_Gestures\test'

train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preproce
ss_input).flow_from_directory(directory=train_path,
target_size=(64,64),class_mode='categorical', batch_size=10,shuffle=True)
```

```
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.preproce
ss_input).flow_from_directory(directory=test_path,
target_size=(64,64),class_mode='categorical', batch_size=10, shuffle=True)

# train_datagen = ImageDataGenerator(
#         rescale=1./255,
#         shear_range=0.2,
#         zoom_range=0.2,
#         horizontal_flip=True)

# train_datagen = ImageDataGenerator(rotation_range=15,
#                         rescale=1./255,
#                         shear_range=0.1,
#                         zoom_range=0.2,
#                         horizontal_flip=True,
#                         width_shift_range=0.1,
#                         height_shift_range=0.1
#                         )

# train_batches  =  train_datagen.flow_from_directory(train_path,
#                                     target_size=(128, 128),
#                                     batch_size=10,
#                                     class_mode='categorical')

# test_datagen  =  ImageDataGenerator(rotation_range=15,
#                         rescale=1./255,
```

```
#                       shear_range=0.1,
#                       zoom_range=0.2,
#                       horizontal_flip=True,
#                       width_shift_range=0.1,
#                       height_shift_range=0.1)

# test_batches = test_datagen.flow_from_directory(test_path,
#                                 target_size=(128, 128),
#                                 batch_size=10,
#                                 color_mode='grayscale',
#                                 class_mode='categorical')

# Creating Model:

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Flatten())
model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.2))
```

```
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.3))
model.add(Dense(26,activation ="softmax"))

# model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
# reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
# early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')

model.compile(optimizer=SGD(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0005)

early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')

#Training our model:

history = model.fit(train_batches, epochs=10, callbacks=[reduce_lr, early_stop], validation_data = test_batches)

# Saving the model:

model.save(r'Sign_Language_Recognition_model.h5')
```

Model Evaluation Code:

```
imgs, labels = next(test_batches)

model = keras.models.load_model(r"Sign_Language_Recognition_model.h5")

scores = model.evaluate(imgs, labels, verbose=0)
print(f'{model.metrics_names[0]} of {scores[0]}; {model.metrics_names[1]} of
{scores[1]*100}%')

# model.summary()

scores #[loss, accuracy] on test data...
model.metrics_names

# word_dict =
{A:'A',B:'B',B:'B',C:'C',D:'D',E:'E',F:'F',G:'G',H:'H',I:'I',J:'J',K:'K',L:'L',M:'M',N:'N'
,O:'O',P:'P',Q:'Q',R:'R',S:'S',T:'T',U:'U',V:'V',W:'W',X:'X',Y:'Y',Z:'Z'}
import matplotlib.pyplot as plt
%matplotlib inline

def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(30,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        ax.imshow(img)
        ax.axis('off')
```

```
    plt.tight_layout()
    plt.show()

word_dict =
{0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O
',15:'P',16:'Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X',24:'Y',25:'Z'}

predictions = model.predict(imgs, verbose=0)
print("predictions on a small set of test data--")
print("")
for ind, i in enumerate(predictions):
    print(word_dict[np.argmax(i)], end=' ')

plotImages(imgs)

print('Actual labels')
for i in labels:
    print(word_dict[np.argmax(i)], end=' ')

print(imgs.shape)

# history.history
```

3.) Creating Interface For Real-Time Sign Language Recognition:

Real_time_Sign_Language_Detection.py

#importing Library:

```
import numpy as np
import cv2
import keras

from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tkinter import *
from gtts import gTTS
import os
from playsound import playsound
from translate import Translator

model =  keras.models.load_model(r"Sign_Language_Recognition_model.h5")#
```

Main Code:

```
camera = cv2.VideoCapture(0)
# num_frames =0
word_dict                                                                    =
{0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',13:'N',14:'O
```

```
',15:'P',16:'Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X',24:'Y',25:'Z'}


# def speak(msg):
#     english_lang = 'en'
#     speech = gTTS(text = msg , lang = english_lang)
#     speech.save('Sarthak.mp3')
#     playsound('Sarthak.mp3')
#     reset()


# def reset():
#     os.remove('Sarthak.mp3')

while True:
    ret, frame = camera.read()
    # filpping the frame to prevent inverted image of captured frame...
    frame = cv2.flip(frame, 1)
    frame_copy = frame.copy()


    # Coordinates of the ROI
#     x1 = int(0.5*frame.shape[1])
#     y1 = 10
#     x2 = frame.shape[1]-10
#     y2 = int(0.5*frame.shape[1])


    ROI_top = 100
    ROI_bottom = 300
    ROI_right = 150
    ROI_left = 350
    # Drawing the ROI
```

```
    # The increment/decrement by 1 is to compensate for the bounding box
#     cv2.rectangle(frame, (220-1, 9), (620+1, 419), (255,0,0) ,1)
    # Extracting the ROI
#     roi = frame[10:410, 220:520]
    roi = frame[ROI_top:ROI_bottom, ROI_right:ROI_left]



#     roi = cv2.resize(roi, (64, 64))
#     cv2.imshow("Frame", frame)
    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray,(5,5),2)
    th3 = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.
THRESH_BINARY_INV,11,2)
#     ret, test_image = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#     time.sleep(5)
    ret, res = cv2.threshold(th3, 70, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    thresholded = cv2.resize(res,(64,64))
    thresholded = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2RGB)
    thresholded = np.reshape(thresholded, (1,thresholded.shape[0],thresholded.shape[1],3))

    #Predictions
    pred = model.predict(thresholded)
    cv2.putText(frame, word_dict[np.argmax(pred)], (170, 45), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
```

```
#     msg = word_dict[np.argmax(pred)]
#     speak(msg)

    # Draw ROI on frame_copy
    cv2.rectangle(frame, (ROI_left, ROI_top), (ROI_right, ROI_bottom), (255,128,0), 3)

    # Display the frame with segmented hand
    cv2.putText(frame_copy, "Real-Time Sign Language Recognition", (10, 20), cv2.FONT_ITALIC, 0.5, (51,255,51), 1)
    cv2.imshow("Sign Detection", frame)

    # Close windows with Esc
    k = cv2.waitKey(1) & 0xFF

    if k == 27:
        break

# Release the camera and destroy all the windows
camera.release()
cv2.destroyAllWindows()
```