TALK2MUTE

A Project Report

Submitted by

NEETHUU N	**JEC17CS074**
SIDHARTH U	**JEC17CS095**
SREEHARI	**JEC17CS097**
VINCY ANTO	**JEC17CS104**

to

APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

in

COMPUTER SCIENCE & ENGINEERING

Under the guidance of

MS. SAJITHA I

CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Jyothi Engineering College

NAAC Accredited College with NBA Accredited Programmes*

Approved by AICTE & affiliated to APJ Abdul Kalam Technological University

A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR

JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777

NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

June 2021

DECLARATION

We the undersigned hereby declare that the project report "TALK2MUTE", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Ms. Sajitha I. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in this submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously used by anybody as a basis for the award of any degree, diploma or similar title of any other University.

Name of Students	**Signature**
NEETHUU N (JEC17CS074)	
SIDHARTH U (JEC17CS095)	
SREEHARI (JEC17CS097)	
VINCY ANTO (JEC17CS104)	

Place:

Date:

Jyothi Engineering College

NAAC Accredited college with NBA Accredited programmes

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JYOTHI ENGINEERING COLLEGE
LEAD KINDLY LIGHT
TET

CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

CERTIFICATE

This is to certify that the report entitled " **TALK2MUTE** " submitted by NEETHUU N (JEC17CS074), SIDHARTH U (JEC17CS095), SREEHARI (JEC17CS097), and VINCY ANTO (JEC17CS104) to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in **Computer Science & Engineering** is a bonafide record of the project work carried out by them under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Dr. Saju P John

Professor
Internal Supervisor & Head of the Department

ACKNOWLEDGEMENT

We take this opportunity to thank everyone who helped us profusely, for the successful completion of our project work. With prayers, we thank **God Almighty** for his grace and blessings, for without his unseen guidance, this project would have remained only in our dreams.

We thank the **Management** of Jyothi Engineering College and our Principal, **Dr. Sunny Joseph Kalayathankal** for providing all the facilities to carry out this project work. We are grateful to the Head of the Department **Prof. Dr. Saju P John** for his valuable suggestions and encouragement to carry out this project work. Our sincere thanks to **Dr. Vinith R**, former Head of the Department for permitting us to make use of the facilities available in the department to carry out the project successfully.

We would like to express our whole hearted gratitude to the project guide **Ms. Sajitha I** for her encouragement, support and guidance in the right direction during the entire project work.

We thank our Project Coordinators **Mr. Shaiju Paul** & **Dr. Swapna B Sasi** for their constant encouragement during the entire project work. We extend our gratefulness to all teaching and non teaching staff members who directly or indirectly involved in the successful completion of this project work.

Finally, we take this opportunity to express our gratitude to the parents for their love, care and support and also to our friends who have been constant sources of support and inspiration for completing this project work.

NEETHUU N (JEC17CS074)
SIDHARTH U (JEC17CS095)
SREEHARI (JEC17CS097)
VINCY ANTO (JEC17CS104)

VISION OF THE INSTITUTE

Creating eminent and ethical leaders through quality professional education with emphasis on holistic excellence.

MISSION OF THE INSTITUTE

- To emerge as an institution par excellence of global standards by imparting quality Engineering and other professional programmes with state-of-the-art facilities.
- To equip the students with appropriate skills for a meaningful career in the global scenario.
- To inculcate ethical values among students and ignite their passion for holistic excellence through social initiatives.
- To participate in the development of society through technology incubation, entrepreneurship and industry interaction.

VISION OF THE DEPARTMENT

Creating eminent and ethical leaders in the domain of computational sciences through quality professional education with a focus on holistic learning and excellence.

MISSION OF THE DEPARTMENT

- To create technically competent and ethically conscious graduates in the field of Computer Science & Engineering by encouraging holistic learning and excellence.
- To prepare students for careers in Industry, Academia and the Government.
- To instill Entrepreneurial Orientation and research motivation among the students of the department.
- To emerge as a leader in education in the region by encouraging teaching, learning,industry and societal connect

PROGRAMME EDUCATIONAL OBJECTIVES

PEO 1: The graduates shall have sound knowledge of Mathematics, Science, Engineering and Management to be able to offer practical software and hardware solutions for the problems of industry and society at large.

PEO 2: The graduates shall be able to establish themselves as practising professionals, researchers or Entrepreneurs in computer science or allied areas and shall also be able to pursue higher education in reputed institutes.

PEO 3: The graduates shall be able to communicate effectively and work in multidisciplinary teams with team spirit demonstrating value driven and ethical leadership.

PROGRAMME SPECIFIC OUTCOMES

Graduate possess -

PSO 1: An ability to apply knowledge of data structures and algorithms appropriate to computational problems.

PSO 2: An ability to apply knowledge of operating systems, programming languages, data management, or networking principles to computational assignments.

PSO 3: An ability to apply design, development, maintenance or evaluation of software engineering principles in the construction of computer and software systems of varying complexity and quality.

PSO 4: An ability to understand concepts involved in modeling and design of computer science applications in a way that demonstrates comprehension of the fundamentals and trade-offs involved in design choices.

PROGRAMME OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OUTCOMES

COs	Description
C410.1	The students will be able to analyse a current topic of professional interest and present it before an audience.
C410.2	Students will be able to identify an engineering problem, analyse it and propose a work plan to solve it.
C410.3	Students will have gained thorough knowledge in design, implementations and execution of Computer science related projects.
C410.4	Students will have attained the practical knowledge of what they learned in theory subjects.
C410.5	Students will become familiar with usage of modern tools.
C410.6	Students will have ability to plan and work in a team.

CO MAPPING TO POs

	POs											
COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C410.1	3	2	3	3	3	3	2	3	3	2	3	2
C410.2	2	3	3	3	3	3	2	3	2	3	3	3
C410.3	3	2	2	3	3	3	3	3	2	3	3	3
C410.4	3	3	2	3	3	3	3	2	3	3	3	3
C410.5	2	3	3	3	2	3	3	2	2	3	2	2
C410.6	3	2	3	2	2	3	2	3	2	3	2	2
Average	2.67	2.5	2.67	2.83	2.67	3	2.5	2.67	2.33	2.83	2.67	2.67

CO MAPPING TO PSOs

	PSOs			
COs	PSO1	PSO2	PSO3	PSO4
C410.1	3	3	1	3
C410.2	2	3	2	3
C410.3	3	1	3	2
C410.4	2	1	3	2
C410.5	3	1	2	2
C410.6	3	2	3	3
Average	2.67	1.833	2.33	2.5

ABSTRACT

Speech impairment is a disability which affects one's ability to communicate with others. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people. Sign Language Recognition is one of the most growing fields of research area. Many new techniques have been developed recently in this area. The Sign Language is mainly used for communication of the deaf-dumb people. The people who know sign language can communicate with each other efficiently. However, when it comes to communicating with people who don't understand sign language it causes a lot of problems. Therefore, we are proposing a system that translate sign language to English and Malayalam languages in the form of text and audio, thus aiding more effective and easier communication using sign language. The Convolutional Neural Network is used for gesture recognition and the recognized gesture is converted into text and voice format. The main goal of this project is to validate the usage of this methodology to identify the sign language gestures shown by a signer and translate it into its corresponding text and audio output.

Keywords: Sign Language Recognition, Convolutional Neural Network, Feature Extraction, Gesture, Classification.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

CNN : Convolution Neural Networks

DFD ; Data Flow Diagram

KNN : K Nearest Neighbour

LDA : Linear Discriminant Analysis

ASL : American Sign Language

ISL : Indian Sign Language

SDLC : Software Development Life Cycle

SLR : Sign Language Recognition

API : Application Programming Interface

ROI : Region Of Interest

TTS : Text To Speech

PCA : Principal Component Analysis

HOG : Histogram of Oriented Gradients

SVM : Support Vector Machine

CHAPTER 1

INTRODUCTION

1.1 Overview

Sign language is an important part of life for deaf and mute people. They rely on it for everyday communication with their peers. A sign language consists of a well-structured code of signs, and gestures, each of which has a particular meaning assigned to it. They have their own grammar and lexicon. It includes a mixture of hand positioning, shapes and movements of the hand. The people who know sign language can communicate with each other efficiently. However, when it comes to communicating with people who don't understand sign language it causes a lot of problem. Communication is a very important part of our lives. We interact with our mates at offices, schools, hospitals and other public places. Deaf and mute people may find it difficult to express themselves in such situations because not everyone understands sign language. There are many highly talented people suffering from speech impediment. We feel that their disability should not become a hindrance to achieve their goals. Adding them into the workforce will only improve the socio-economic development of the country. Deaf and mute people usually depend on sign language interpreters for communication. However, finding a good interpreter is difficult and often expensive. Thus, a computerized interpreter could be a reliable and cheaper alternative. A system that can translate sign language into plain text or audio can help in real-time communication. It can also be used to provide interactive learning of sign language.

This project aims to get the deaf and dumb people more involved into communication and the idea of a camera-based sign language recognition system that would be in use for converting sign language gestures to text (English) and then to Malayalam.The proposed system contains modules such as: Data Acquisition, Sign Identification and Translation. The Convolutional Neural Network (CNN) algorithm, was used for gesture recognition and the recognized gesture is converted into text and voice. This system makes the communication between speech-impaired people and the rest simple and comfortable.

1.2 Objectives

The main objective of this project is to identify the sign language gestures which are shown to a web camera using the CNN Algorithm in machine learning and given as text and speech output. Thereby all people can understand the intended meaning of each sign language gesture regardless of whether the person is trained or not.

1.3 Data Description

The data for this project is taken from an open source platform known as GitHub and is available at https://github.com/loicmarie/sign-language-alphabet-recognizer. The complete data set is divided into training and validation set. As is the case with a usual machine learning problem, we would be training the model using the training data set and evaluating the performance with the test data set.

1.4 Organization of the project

The report is organised as follow:

- **Chapter 1:Introduction** Gives an introduction to ”TALK2MUTE”, our project which identifies the sign language gestures and converts it into text and speech.
- **Chapter 2:Literature Survey** Summarizes the various existing techniques that helps in achieving the desired result.
- **Chapter 3: Methodology** Describes the various steps involved to produce this project.
- **Chapter 4:Results & Discussions** Contains the results of work done and discussion.
- **Chapter 5:Conclusion & Future Scope** Concludes with the future scope of implementation.
- **References** Includes the references for the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Sign Language Converter

The aim of this paper is to improve the communication with the people who has hearing difficulties and using any sign language to express themselves. At the first sight, as an idea, how difficult could make a sign languages converter. After detailed research about sign language linguistics, it is figured out about 240 sign languages have exist for spoken languages in the world. Infrastructure of a sign language system consists of three main branches as Sign Language, Speech Recognition and Implementation with MS Kinect XBOX 360TM. Infrastructure of this sign language system consists of three main branches as Sign Language, Speech Recognition and Implementation with MS Kinect XBOX 360TM[1].

2.1.1 Implementation

Sign Language

It is easy to find a wide number of sign languages all over the world and almost every spoken language has its respective sign language, so there are about more than 200 languages available. American Sign Language (ASL) is well-known and the best studied sign language in the world. The different basic signs from the ASL dictionary have been taken and the parameters assigned to these characteristics as respect to the position, motion and plane.

Microsoft Kinect XBOX 360TM

Microsoft Kinect Sensor XBOX 360TM used for capturing abilities and technical features to the motion capture of sign to voice conversion. Part A is a depth sensor or called as 3D sensor too. It is a combine of infrared laser projector with a CMOS sensor to let the Kinect sensor to process 3D scenes in any environmental light conditions. Part B is RGB camera which has 32 bits high color resolution. It can use 2 dimensional color video of the scene. Part C is motorized tilt which is concerned with the field of view. Part D contains an array of 4 microphones which is located along the horizontal bar. It is useful for speech recognition, ambient noise suppression and echo cancellation[5].

Open NI Middleware

Open NI is a multi-language that defines APIs for writing applications as a cross platform framework. Main purpose of Open NI is to form a standard API that enables communication with the sensors in the system such as vision and audio sensors. Open NI standard API let the

natural interaction application developers to 3D scenes by utilizing data types like array of the pixels in a depth map.Open NI has 3 layers. Bottom layer contains devices that collect visual and audio data from the real world. Second layer contains Middleware components to analyze these data which is collected from the real world. The top layer contains software which implements natural application such as Sign Language Translator.

2.1.2 Methodology

There are three parts of methodology:

1. Database
2. Voice Recognition Procedure
3. Motion Capture Procedure

Database

Words for Speech Recognition, .gif images and Motions together create the database.

Voice Recognition Procedure

Speech processing is the field which works on the speech signals and the processing of them. Thesignals are usually processed in a digital representation, although the signals are analog. Once the user press the button to record the speech, computer's microphone starts to listen, and after catching the voice with the help of CMU Sphinx, it finds the meaning as the text. Then in Java it is matched with the proper .gif image, so that the other user will understand.

Motion Capture Procedure

In this procedure, image processing is really important. Image processing is used commonly in our life recently, and it seems that future will bring much more than that. the motion capturing is the part where Kinect Sensor is used. Once the user press the button to record the motion, Kinect sensor starts to capture motions, but it starts to record the motion with a specific "starting motion". After the "Starting Motion", Kinect captures the motions and it converts them to the text. On the computer, this text is converted to the voice and then the other user can hear the meaning of the sign.

2.1.3 Results

This paper is about a system that can support the communication between deaf and ordinary people. The aim of the study is to provide a complete dialog without knowing sign language. After this system, it is an opportunity to use this type of system in any places such

as schools, doctor offices, colleges, universities, airports, social services agencies, community service agencies and courts, briefly almost everywhere.

2.2 Conversion of Sign Language to Text and Speech Using Machine Learning techniques

Communication has been defined as an act of conveying intended meanings from one entity or group to another through the use of mutually understood signs and semiotic rules. It plays a vital role in the existence and continuity of human. For an individual to progress in life and coexist with other individuals there is the need for effective communication. Effective communication is an essential skill that enables us to understand and connect with people around us. It allows us to build respect and trust, resolve differences and maintain sustainable development in our environment where problem solving, caring and creative ideas can thrive. Poor communication skills are the largest contributor to conflict in relationships[2].

2.2.1 PCL System

Research has shown that over nine billion people at intervals, all over the world are physically challenged in terms of communication; blind, deaf or mute. Academic and industrial researchers have recently been focusing on analyzing images of people and there has been a surge interest in recognizing human gestures. A research on scene segmentation of images was carried out using deep learning techniques; the classification yielded 53.8 percentage accuracy. In relation to conversion of sign language, there is the need to explore other image classification techniques to enhance accurate classification.

2.2.2 Methods and Materials

The aim of the study as earlier stated was to provide an unsupervised learning feature of signed hand gestures while the system returns corresponding output as text and speech. The following were the measurable methods employed in actualising the aim:

1. Segmentation of captured signed gestures of ASL as inputs
2. Feature extraction of the segmented images
3. UFL and classification of several images
4. Text and Speech synthesis of classified images

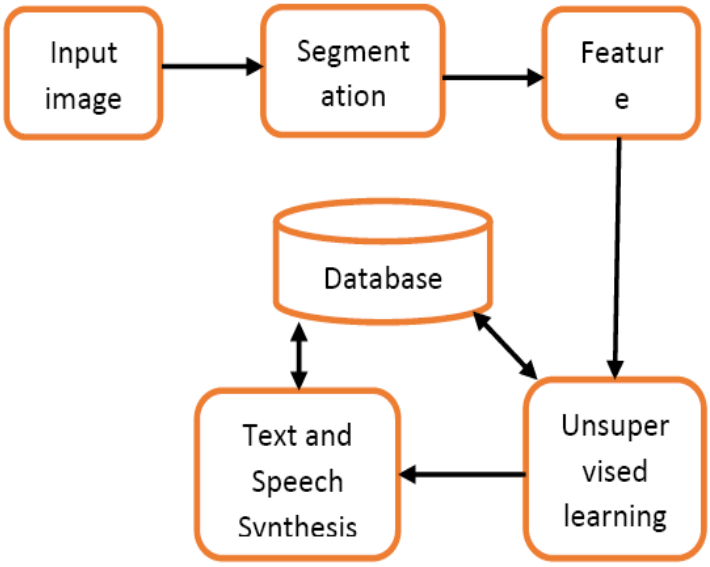

Figure 2.1: Overview of the system

- **Segmentation of captured signed gestures of ASL as inputs**:

  The aim of segmentation was to convert images into more meaningful and easy to analyse portions. Segmentation does the job of partitioning an image into multiple segments which help to locate the objects and boundaries (curves, arcs, lines, etc.) in an image in binary form. The set of images captured from the Kinect sensor using the Image Acquisition Tool in MATLAB would be selected and fed into the Image Segmenter in MATLAB which is then converted to grayscale image. The threshold of the images is then obtained by converting grayscale images into binary images to determine the high level contrast of the images. Such images can then be cropped or resized.

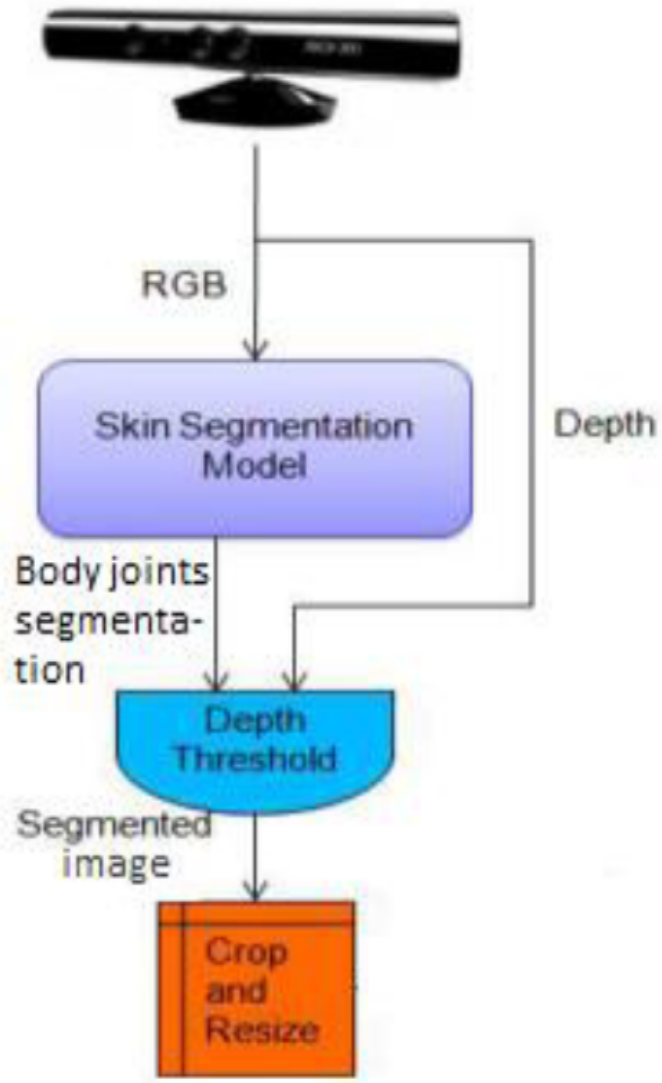

Figure 2.2: Image Segmentation

- **Feature extraction of the segmented images**:

  Sign Language usually involves movement in the upper part of the body; head, shoulder, hands and elbows coordinates are retained while other parts are discarded. To satisfy this need, key points corresponding to high-contrast locations such as object edges and corners were used. These features are intended to be non-redundant, informative and relevant for the intended use. Extracting ROI from images has been very much challenging as it is the base for further image analysis, interpretation and classification. A rectangular ROI whose outline consists of four segments joining the four corner points is used to make computational statistics feasible.Batch segmentation for all training samples was carried out to convert the coloured images into binary form

with MATLAB Image Segmenter and Batch Processor toolbox. Basically only the set of binarised images are useful in feature detection and extraction.

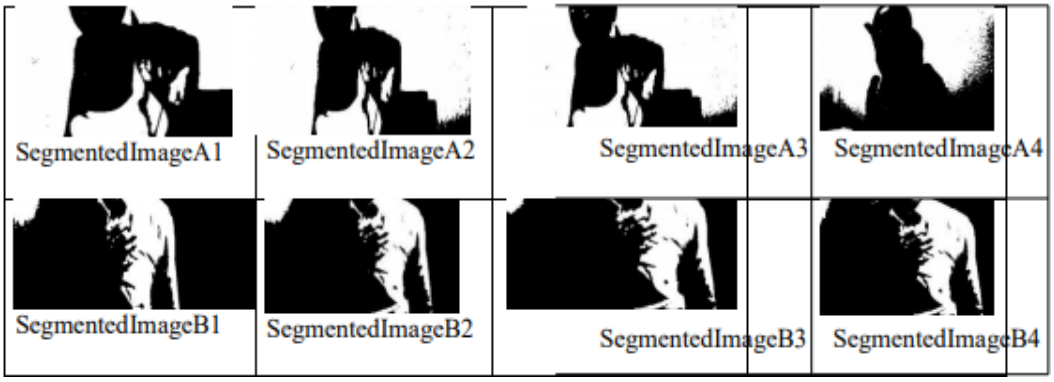

Figure 2.3: Segmented ImageSet of Coloured Images

The vertices of an ROI outline may be positioned anywhere with respect to the array of image pixels, so the same Rectangular ROI superimposed on the pixel array may appear.

- **UFL and classification of several images**:

  The identification of interest points present within the space of an image is important in the determination of the image's ROI, therefore the method being proposed in this paper maximizes the number of interest points detected within a sample image through the use of the combination of FAST corner detector and SURF detector

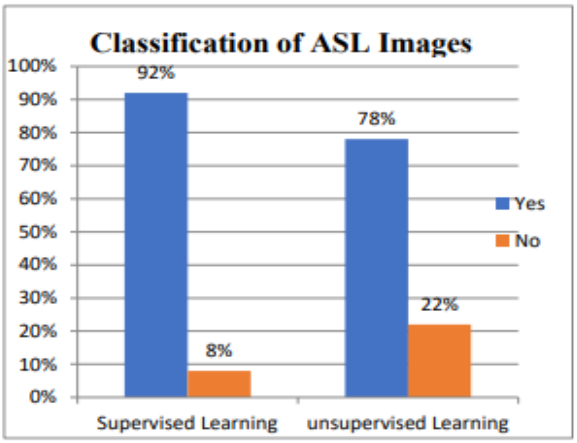

Figure 2.4: Classification of ASL Images

- **Text and Speech synthesis of classified images**:

  Text-to-Speech (TTS) refers to the ability of computers to read text aloud. A TTS Engine converts written text to a phonemic representation, and then converts the

phonemic representation to waveforms that can be output as sound. Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware. After the successful classification of these features, the important task is to generate appropriate text and speech output for every input image using MATLAB Speech Synthesizer[6].

2.2.3 Results

Sample images of different ASL signs were collected using the Kinect sensor using the image acquisition toolbox on MATLAB. About five hundred (500) data samples (with each sign count five and ten (5-10)) were collected as the training data. The reason for this is to make the algorithm very robust for images of the same database in order to reduce the rate of misclassification.

The study findings show that American Sign Language (ASL) is commonly used in Nigeria by the hearing impaired hence; five hundred (500) ASL images were collected as training set. From the collection of images, a database of forty-nine (49) different signs was used. Having subjected the set of images to batch segmentation, features of each signs were detected and extracted from specific bounding-box of Region of Interest (ROI) to aid supervised learning. The combination FAST and SURF with a KNN of 10 also showed that unsupervised learning classification could determine the best matched feature from the existing database. In turn, the best match was converted to text as well as speech. The introduced system achieved a 92% accuracy of supervised feature learning and 78% of unsupervised feature learning.

2.3 Conversion of Sign Language into Text

Sign Language is the most natural and expressive way for the hearing impaired people. People, who are not deaf, never try to learn the sign language for interacting with the deaf people. This leads to isolation of the deaf people. But if the computer can be programmed in such a way that it can translate sign language to text format, the difference between the normal people and the deaf community can be minimized. Indian sign language (ISL) uses both hands to represent each alphabet and gesture. ISL alphabets are derived from British Sign Language (BSL) and French Sign Language (FSL). When compared with ASL, Indian Sign Language relies on both hands and thus, an ISL recognition system is more complex[3].

Deaf and Dumb people rely on sign language interpreters for communications. A real time Sign Language Recognition system was designed and implemented to recognize 26 gestures from the Indian Sign Language by hand gesture recognition system for text generation. The signs are captured by using web cam. This signs are processed for feature extraction using some colour model. The extracted features are compared by using pattern matching algorithm. In order to calculate the sign recognition, the features are compared with testing database. Finally, recognized gesture is converted into text. This system provides an

opportunity for a deaf-dumb people to communicate with non-signing people without the need of an interpreter.

ASL vocabulary dictionary contains thousands of sign justlike words. It is very easy to get two completely different signs mixed up which leads to bad miscommunication. For example, the sign for "chocolate" and "cleve land" are similar, and they definitely don't mean the same thing, or even close. It is very hard to follow when a conversation has and something gets mixed up.

Related works:
A various hand gestures were recognized with different methods by different researchers in which were implemented in different fields. The recognition of various hand gestures were done by vision based approaches, data glove based approaches, soft computing approaches like Artificial Neural Network, Fuzzy logic, Genetic Algorithm and others like PCA, Canonical Analysis, etc. The recognition techniques are divided into three broad categories such as Hand segmentation approaches, Feature extraction approaches and Gesture recognition approaches.

2.3.1 LDA Approach

The Generalization of the Fisher's linear discriminant(FLD) is known as Linear discriminant analysis (LDA). LDA mainly used in statistics, pattern recognition and machine learning. It is used to find a linear combination of features that characterizes or separates two or more classes of objects or events. The LDA and FLD are used linear classifier. Its combination also used for dimensionality reduction before later classification[7].

LDA is also closely resembles to principal component analysis (PCA) and factor analysis. Both PCA and factor analysis is linear combinations of variables and they describe the data in a better manner. LDA explains to model the difference between the classes of data. PCA cannot consider the difference in class but factor analysis builds the feature combinations based on differences rather than similarities. There is a difference between Discriminant analysis and factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made. In LDA, the measurements made on independent variables for each observation are continuous quantities. Discriminant correspondence analysis is used to deal with categorical independent variables in LDA.

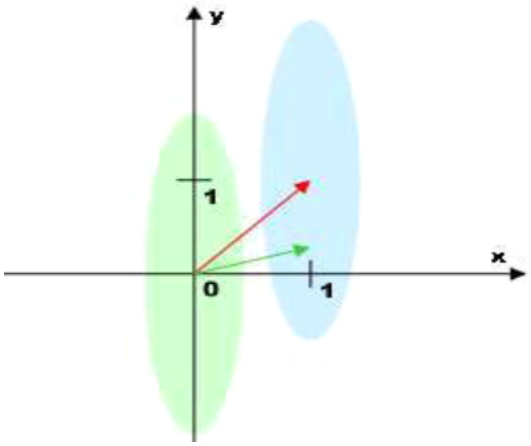

Figure 2.5: LDA Approach

The above figure 2.9 explains about the LDA approach.

- Compute the d-dimensional mean vectors for the different classes from the dataset
- Compute the scatter matrices (between class and within-class scatter matrix)
- Compute the eigenvectors (e1, e2, ..., ed) and corresponding eigenvalues (1, 2, ..., d) for the scatter matrices
- Sort the eigenvectors by decreasing Eigen values and choose k eigenvectors with the largest Eigen values to form a d×k-dimensional matrix W (where every column represents an eigenvector)
- Use this d×k eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the equation Y = X × W (where X is an n×d-dimensional matrix; the ith row represents the ith sample, and Y is the transformed n×k-dimensional matrix with the n samples projected into the new subspace)

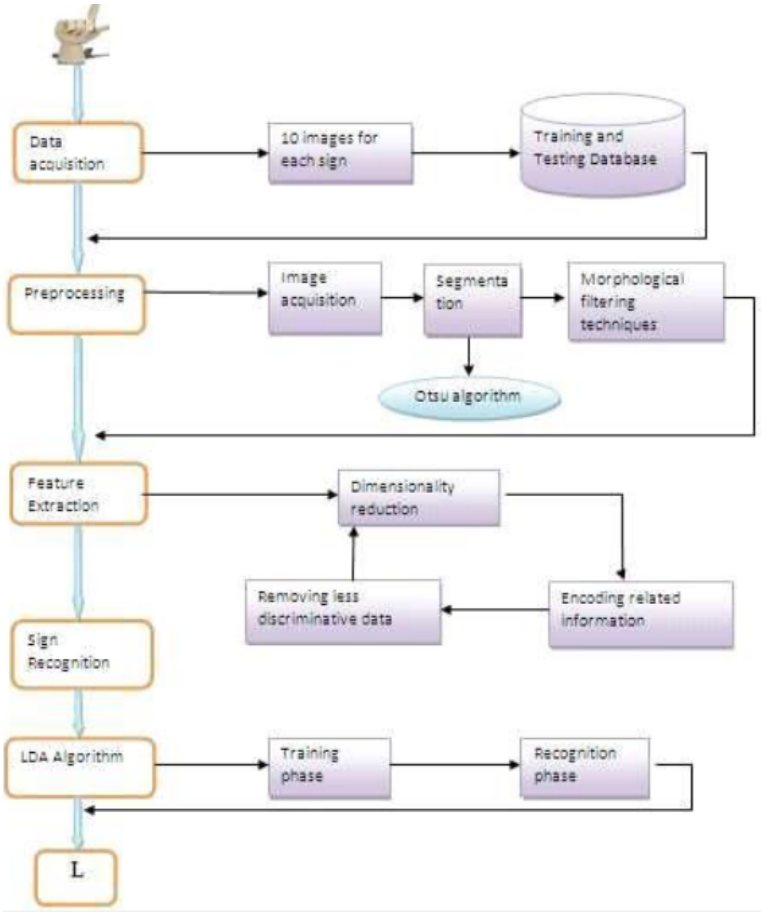

Figure 2.6: System Overview of Indian Sign Language

2.3.2 Data Acquisition

To achieve a high accuracy for sign recognition in sign language recognition system, 10 images will be taken for each 26 signs. These images are included in training and testing database. The captured image at a distance is adjusted by the signer to get the required image clarity

2.3.3 Pre-Processing

Pre-processing consist image acquisition, segmentation and morphological filtering methods.

The above figure 2.9 explains about the LDA approach.

- **Image acquisition**
  This is the first step of pre-processing. This is the process of sensing of an image. So in an image acquisition, image is sensed by "illumination". It will also involve pre-processing such as scaling. In image acquisition the image will be taken from database.

- **Segmentation**

  Segmentation is the process in which image is converted into small segments so that the more accurate image attribute can be extracted. If the segments are properly autonomous (two segments of an image should not have any identical information) then representation and description of image will be accurate and while taking rugged segmentation, the result will not be accurate. Here the Segmentation of hands is carried out to separate object and the background. Otsu algorithm is used for segmentation purpose. The segmented hand image is represented certain features. The following figure 4 shows the segmented of hand image.

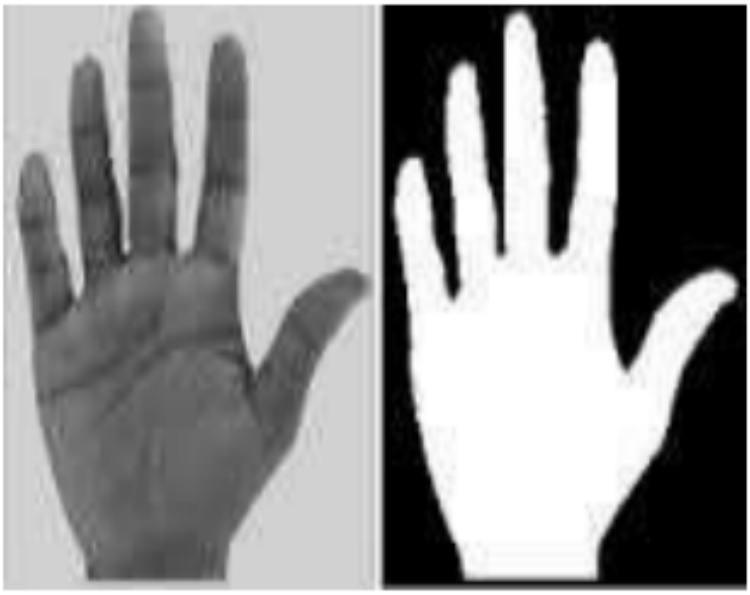

Figure 2.7: Segmented Image

- **Morphological Filtering**

  The image components are extracted by Morphological Filtering tools which are useful for representation and description of shape. Definitely the output of this process is image attribute. The following figure 5 shows the filtered form of segmented image.

  The features extracted from the segmentation operation used for gesture recognition. The smooth contour is obtained by removing the noise from the images with Morphological filtering techniques. The pre-processing operation is done on the stored database.

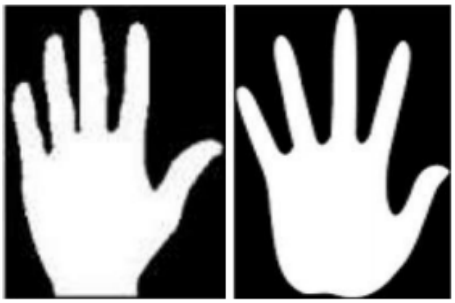

Figure 2.8: Morphological Filtered Image

2.3.4 Feature Extraction

The reduction of data dimensionality by encoding related information in a compressed representation and removing less discriminative data is called as Feature extraction Technique. Feature extraction is vital to gesture recognition performance. Therefore, the selection of which features to deal with and the extraction method are probably the most significant design decisions in hand motion and gesture recognition development. Here principal component is used as main features

Figure 2.9: Feature extraction method

2.3.5 Sign Recognition

Sign recognition using LDA is a dimensionality reduction technique based on extracting the desired number of principal components of the multi-dimensional data. The gesture recognition using LDA algorithm that involves two phases.

- **Training Phase** Each gesture is represented as a column vector in the training phase. These gesture vectors are then normalized with respect to average gesture. Next, the algorithm finds the eigenvectors of the covariance matrix of normalized gestures by

using a speed up technique that reduces the number of multiplications to be performed. The corresponding gesture space projections were obtained by the eigenvector matrix then multiplied by each of the gesture vectors.

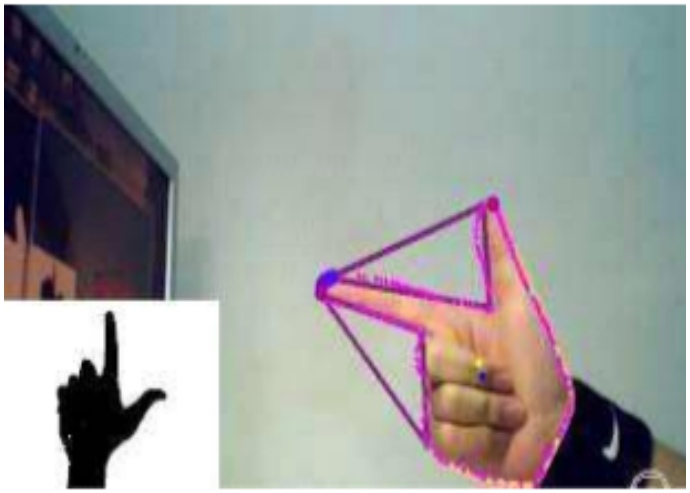

Figure 2.10: Dimensionality reduction

- **Recognition Phase** In the recognition phase, a subject gesture is normalized with respect to the average gesture and then projected onto gesture space using the eigenvector matrix. Lastly, Euclidean distance is computed between this projection and all known projections. The minimum value of these comparisons is selected for recognition during the training phase. Finally, recognized sign is converted into appropriate text and voice which is displayed on GUI.

2.4 Sign Language Interpreter using Image Processing and Machine Learning

Speech impairment is a disability which affects one's ability to speak and hear. Such individuals use sign language to communicate with other people. Although it is an effective form of communication, there remains a challenge for people who do not understand sign language to communicate with speech impaired people.Sign language is an important part of life for deaf and mute people. They rely on it for everyday communication with their peers. A sign language consists of a well-structured code of signs, and gestures, each of which has a particular meaning assigned to it. They have their own grammar and lexicon. It includes a mixture of hand positioning, shapes and movements of the hand. The people who know sign language can communicate with each other efficiently.hearing impaired people. People, who are not deaf, never try to learn the sign language for interacting with the deaf people. This leads to isolation of the deaf people. But if the computer can be programmed in such a way that it can translate sign language to text format, the difference between the normal people

and the deaf community can be minimized.Here this proposed system is able to recognize the various alphabets of Indian Sign Language; this will reduce the noise and give accurate result[4].

However, when it comes to communicating with people who don't understand sign language it causes a lot of problems. Deaf and mute people may find it difficult to express themselves in such situations because not everyone understands sign language. The aim is to develop an application which will translate sign language to English in the form of text and audio, thus aiding communication with sign language. The application acquires image data using the webcam of the computer, then it is pre-processed using a combinational algorithm and recognition is done using template matching. The translation in the form of text is then converted to audio. The database used for this system includes 6000 images of English alphabets. We used 4800 images for training and 1200 images for testing. The system produces 88% accuracy. Here discusses the implementation of a system which translates Indian Sign Language gestures to its English language interpretation.

2.4.1 Proposed Method

The proposed system consists of two main stages: (1) Segmentation of hand (2) Recognition of hand sign. The block diagram shows the working of the proposed system. The features of a hand are an important criterion for the classifier to differentiate between the hand gestures. These characteristics must be able to adapt to different hand and gestures by different people. In this system, we have used histograms of oriented gradients (HOG) as a feature descriptor. It is better than other descriptors because it can adapt to changing illuminations and rotation of objectives. It does not consider an image as a whole, but divides it into smaller cells and then for the pixels within the cells edge or gradient direction histogram is calculated. This approach creates a bin, and clubs the histograms of different samples based on magnitude and angle. In the proposed system, we are first segmenting the hand using YCbCr color space and then processing the image through HOG and then provide it to the model. We trained the SVM classifier using 5000 images and developed a model.

2.4.2 Dataset

We have created a dataset of 26 English language alphabets in Indian Sign Language. Each sign gesture is performed by two different individuals with different hand structure in varied lighting conditions. The videos were recorded on a camera and then each video was broken down frame by frame to images and adjusted to 100 frames and then augmented to get about 250 images for each sign. The data was then divided into 4800 images for training and 1200 images for testing

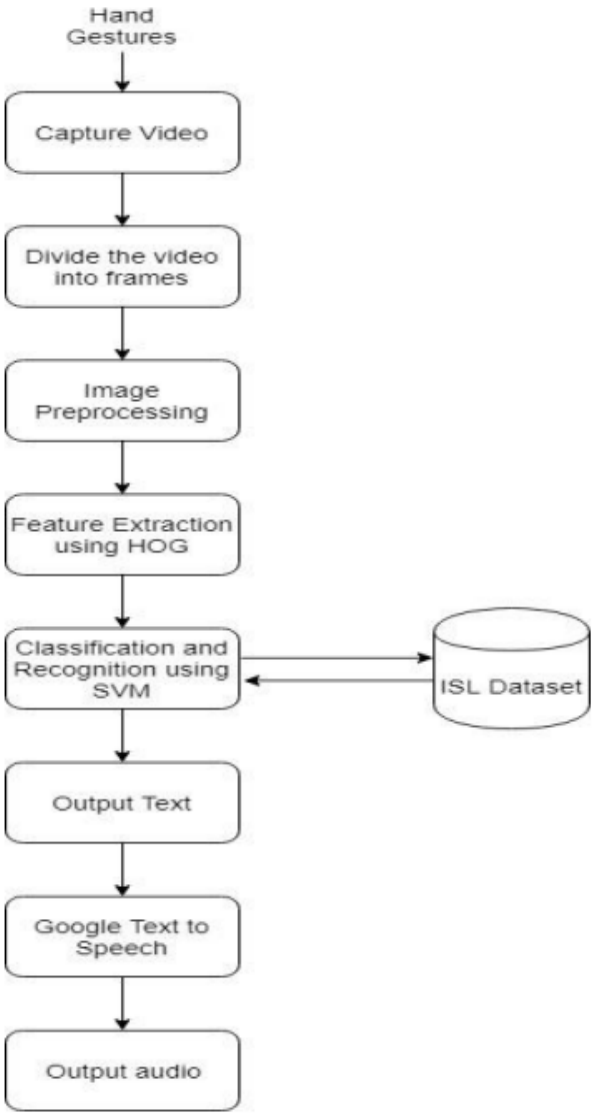

Figure 2.11: Sign language interpreter flowchart

- **Image preprocessing and edge detection**
  Before performing feature extraction, the images must be processed in such a way that only the useful information is considered and the redundant, distracting noise and superficial data are neglected. The images are first converted to 100 * 100 pixel size for faster computations. The image is then converted to grayscale and finally transform into a binary image. Simultaneously, skin color is detected using YCbCr model. Finally, edge detection is performed using Canny edge detector. The process is illustrated in figure below.

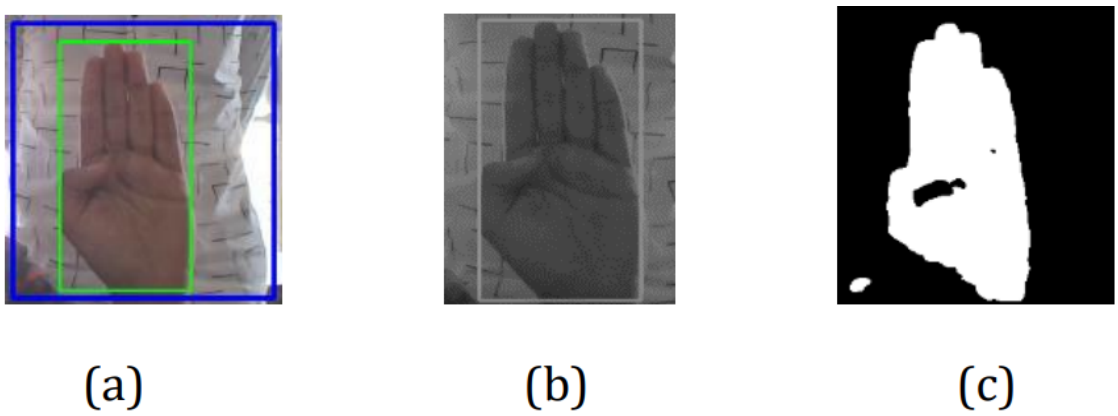

Figure 2.12: a) Original cropped image, b) Grey scale converted image, c) Skin colour detection output

- **Feature extraction**
  Extracting features from the data is a critical part of any object detection system. It can be performed using various methods such as Fourier Descriptor, Scale Invariant Feature Transform (SIFT) or Principle Component Analysis (PCA). Histogram of Oriented Gradients (HOG) is another successful method of feature extraction. In this paper, we have used the HOG method for feature extraction. The basic idea behind HOG is that an object or shape within an image can be represented by intensity distribution gradients or edge directions. In this method, the image is divided into small cells and a histogram of gradient is calculated for each cell. It creates a bin and combines the histogram of different samples based on magnitude and angle. For improving the accuracy, all the cells in the image are normalized so that they are not affected by variations in lighting. Finally, HOG feature vector is calculated for the entire image.

- **Template matching and sign recognition**
  The feature vector produced in the above step is fed into an image classifier. In this paper, we have used Support Vector Machine (SVM) for classification. By using SVM classifier we can maximize accuracy and avoid overfitting of data. In SVM data items are plotted in n-dimensional space where n is the number of features. Each feature is associated with a coordinate value. Then it finds a hyperplane that differentiates the classes. The model is saved for real-time sign language recognition.[9]

- **Text to Speech**
  We have used Google's Text to Speech API for transforming the sign language into audio. It is one of the best text to speech API available. Unlike other TTS APIs, this API generates human-like voice. The sign language is interpreted using the above steps and then the result is fed to text to speech function which converts it to audio. In this

system, we can see and hear the sign language translation at the same time which makes it very convenient to use[10].

2.4.3 Results

In this system, still hand image frame is captured using a webcam. These frames are processed to get enhanced features. Then feature extraction and classification algorithms are used to translate the sign language into English text. This translation is converted to speech using text to speech API. The system has implemented using the above algorithms to get the final output. The proposed model is evaluated by a dataset containing 26 signs from two different people. The results show the overall accuracy of the system to be 88%. Our future research will work towards implementing this model on a mobile application.

2.5 Sign Language Translator using Machine Learning

Machine learning provides a versatile and robust environment to work on. The machine learning subject also eliminates the need for the coder to write updates whenever a new sign is read, this will be done by the machine itself. This system aims to get the deaf and dumb people more involved to communicate and the idea of a camera-based sign language recognition system that would be in use for converting sign language gestures to text (English) and then to regional. A product that is versatile and robust has to be found. Here, the objective is to design a solution that is intuitive and simple which simplifies the communication for the majority of people with deaf and dumb. There are many methods to convert the sign language which often use Kinect as the basic system to get the inputs and work on them for conversion. Kinect methods are complicated in so many aspects.The discussing approach will be simple. Here, used simpler ways to capture the inputs and process them. We have used common and easily available libraries in our system.[8]

The main objective of this project is to recognizing the gestures and displaying the correspondent word. The first phase involves capturing the gesture using a webcam along with pose estimation library.The webcam captures the image and image is processed with pose estimation algorithm in tensor-flow utility. The skeleton mapped on the image is the result of the pose estimation library. The skeleton obtained provides the values for creating the data set; the data set is a collection of the values of the coordinates of the end points of the skeleton. These values are labelled accordingly and are appended to the machine for predicting when the input is taken.

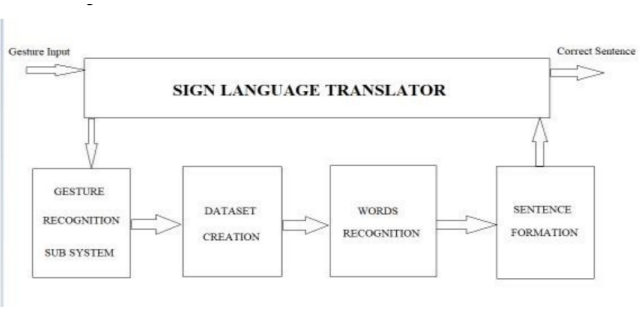

Figure 2.13: Block diagram of the SLR model

2.5.1 Extraction of gestures:

Capturing signs from real world and translating them is the core objective of this work. The real-world signs are read using a webcam which captures both static and moving images of the objects in front of it. The deaf and dumb person who is signing is made to stand in front of the webcam and the image captured from this is processed with the tf-pose-estimation library to map out the skeleton of the person signing. Figure 2.2 is an example of how the skeleton is mapped on the system. Tf-pose-estimation basically sketches out a stick figure of the body. When the webcam is running the pose estimation algorithm identifies the key points on the subject's body such as elbow joints, wrist, knee joints etc and connects them as one skeleton. The key points namely the end points of the skeleton are labelled with x, y and z co-ordinates for every frame captured. As such 17 key points are identified from the pose-estimation algorithm. The value of these coordinates change for different gestures and the relative distance between the key points is different for different people (as size changes from person to person). These coordinates are the main component to form the data set for training.

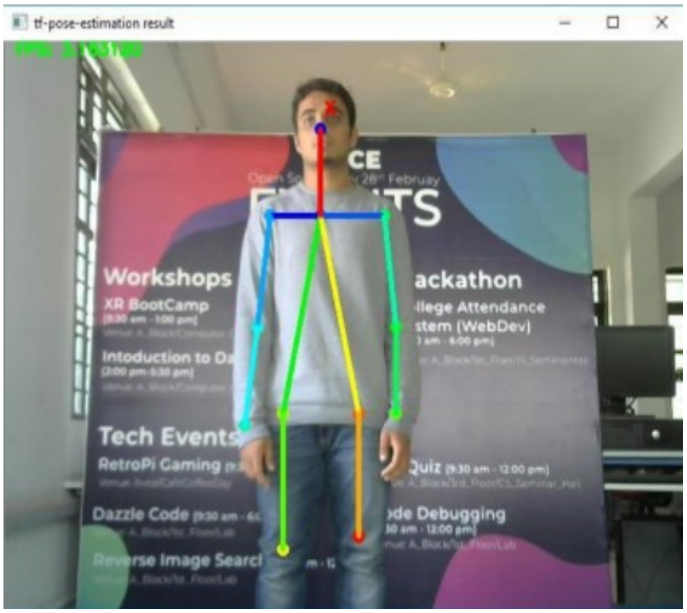

Figure 2.14: Tf-pose-estimation result

Creating Data set:

Each gesture captured has its coordinates values stored in a csv (comma separated file) file and the corresponding labels are written in another csv file. Each frame as 17 key points and each of these points has 3 coordinates; therefore, there exists 17x 3 values for each frame. This entire set of 51 values is labelled together as one gesture. We are assigning numbers for these gestures in the corresponding csv file as it is easy to handle integers while training the machine. Here '0' is the uses to label certain gestures for example in Figure 2.2 a boy standing 'Idle' is recorded figures representing the coordinates of his skeleton, and '0' is the label for this posture. Later '0' is substituted for "Idle" while displaying the result. Training the machine requires several sets of values, therefore a single person has to record many frames for a single gesture and the same gesture has to be signed by different people. Different people are required to sign for the same posture as the size of the skeleton varies from person to person.Several frames have to be recorded for a single gesture by a single person and several people have to record the same gesture to provide a better data set for training[

2.5.2 Training

The data set created is taken up in the training platform and Decision tree algorithm is used to train the machine. The set of values stored for a specific gesture is referred by the machine in its training and makes it possible to predict the gesture when the input is taken.The input values will be run through the tree and the final answer will be displayed along with its value and the corresponding label. The corresponding values for labels are then substituted with words and are displayed in the result. Every new gesture has several frames recorded for

it and trained using the decision tree algorithm. More the number of frames recorded better the efficiency of the system in predicting the gesture.

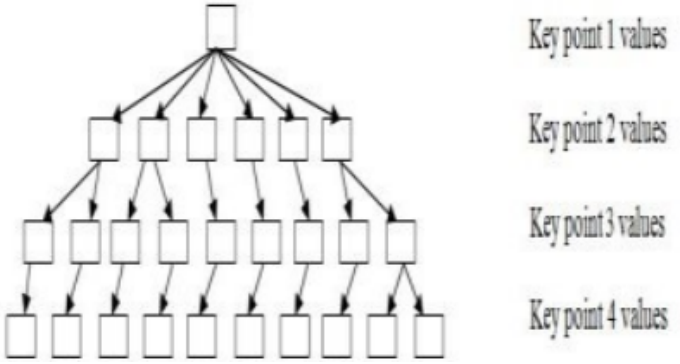

Figure 2.15: Decision Tree Values

The system, when provided with the proper gestures, gives out the corresponding words. The system can provide proper results even when there are some slight variations in gestures. There will be different kinds of variations from different kinds of persons performing the gestures. The system recognizes multiple gestures one after the after and the gives out the respective words.The requirement of machine-based sign language translator is very important in the present scenario. Even though we have found initial success in this regard, lot of work needs to be done.The main area where this can be used is in public places like ticket issuing counters, hospitals etc. This can be even used to teach the sign language to normal people. Further this can be used to take words and display the gesture for the same. Recognizing fingers will widen the training set for the machine.

CHAPTER 3

METHODOLOGY

3.1 Modules

3.1.1 Data Acquisition

The data for sign recognition are gathered from the following source GitHub. The dataset contains files in the jpeg format. All the images where then augmented into 200x200px for better classification and accuracy. The whole set of images in the dataset where divided into training and testing sets with a 60:40 ratio.

The main purpose of this is 60 percent of the dataset images where used for the training and 40 percent is used for testing and validating the accuracy. As a whole, we are using about 67000 images from the dataset for the implementation purpose. Implementation is to be carried out using a platform such as Google Colab. Python is the language used for the whole implementation process.

3.1.2 Detection and Identification Module

Out of different neural networks, CNN is being used. The built-in feature engineering present in Convolutional Neural Network makes it easier to train the classifier. This module identifies and classifies the given data. The main aim of the proposed work is to enrich the true meaning of gesture shown, by CNN using the Inception V3 architecture. The input will be the sign language gesture image which is preprocessed. Then features are extracted from the pre-processed images using the convolutional neural network loaded with the Inception V3 filter banks. Bottleneck was created for the training purpose. A bottleneck layer is a layer that contains few nodes compared to the previous layers. CNN is then used for hand gesture recognition and it will learn from the features that have been extracted from the input images. CNN is known to have less prediction timing and shows greater accuracy[11][12].

The program applies Transfer Learning using Inception v3 to train it to classify a new set of images. Transfer learning is a machine learning method that utilizes a pre-trained neural network. Inception v3 is a variation of CNN, in which the layers are stacked parallel to each other instead of on top of each other. Inception V3 architecture model trained on ImageNet images, and train a new top layer that can recognize other classes of images. The image recognition model called Inception-v3 consists of two parts:

- Feature extraction part with a convolutional neural network
- Classification of the images from the dataset

The label for each image is taken from the name of the subfolder it's in. It is the third edition of the Inception CNN model by Google, originally instigated during the ImageNet Recognition Challenge. The network here is trained for around 40 epochs. The Dataloader function is used here to automatically load the trainset and testset after applying the specified data transforms. The dataloader combines a dataset and a sampler, and makes it iterable.

3.1.3 Translation Module

The preprocessed images are fed to CNN model. The model that has already been trained generates the predicted label. All the gestures labels are assigned with a score. The label with highest score is treated to be the predicted label. The model accumulates the recognized gesture to words and they are further converted into speech using Google API both in English and Regional Language.

Convolutional Neural Network

Convolutional Neural networks are neural networks used for deep learning involving images as their data set. A convolutional network has different layers each of which serves different purpose. The neural network built here in this project consists of around 5 convolution layer, 5 fully-connected layer and 4 max-pool layer each of which serves a specific purpose.

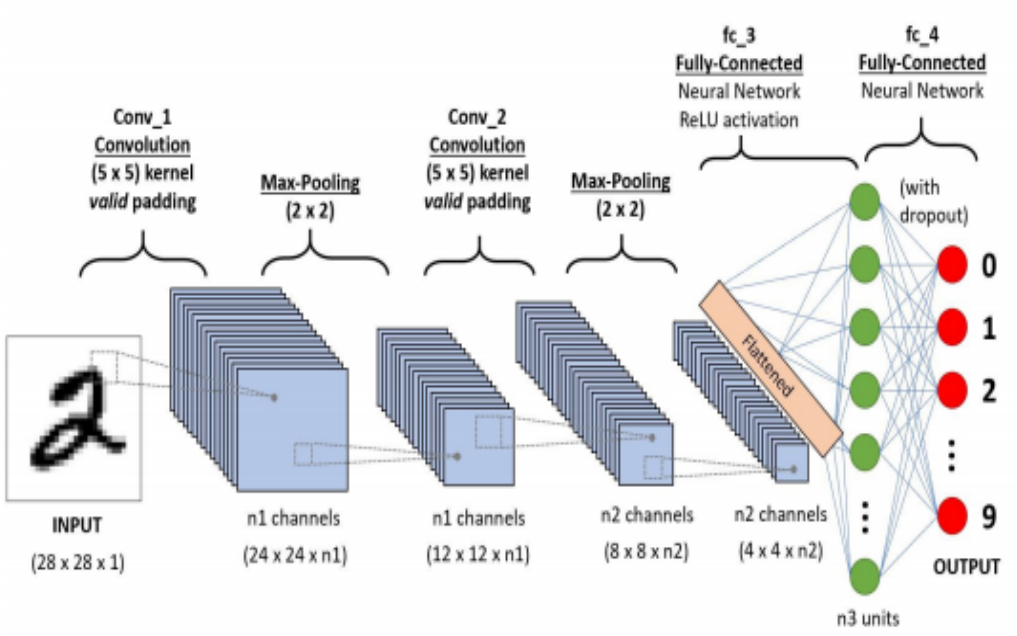

Figure 3.1: Convolutional Neural Network

- The convolution layer takes 3 parameters as input. First, the number of input channels. Here since the input image is black and white image, the number of input channel is

1. Second, the number of output channels which varies in each layer. The number output channel of the previous layer becomes the number of input channels in the next layer. Finally, the third output is the kernal size, which here is 3. Number of strides i.e the amount of movement by network's filter is by default 1. The main purpose of convolution layer is to extract high level features such as edges, from the input image.

- The Maxpool layer takes as input the parameter by which it has to reduce the size of the image each time it passes through it. Here, it is 2 implies that the size of the image after passing through each maxpool layer is halved by taking the maximum value from a set of values included in the kernel each time. The main purpose of this layer is to down-sample the input image and thus reduce the dimensionality.

- The fully connected input layers are linear layers which flattens the output of the previous layers into a single vector which is used as the input for the next stage.The first fully connected layers takes input from feature analysis and uses it to predict the label. The fully connected output layers gives the final probabilities of each class label.

- The ReLU layer in the forward function is an activation function which implements max(input,0), i.e it sets all the negative values to zero and other values are left unchanged. It stands for Rectified Linear Unit.

3.2 System Requirements and Specifications

3.2.1 Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

3.2.2 Windows 10

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1 and was released to manufacturing on July 15, 2015, and to retail on July 29, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users. Mainstream builds of Windows 10 are labeled version YYMM with YY representing the year and MM representing the month of release. For example, the latest mainstream build of Windows 10 is Version 1809. There are additional test builds of Windows 10 available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

3.2.3 Python 3.6.2

Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines. Python is distributed under an OSI-approved open source license that makes it free to use, even for commercial products.

3.2.4 Jupyter Environment

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

3.2.5 API

An application programming interface (API) is a computing interface that defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees.[1] An API can be entirely custom, specific to a component, or designed based on an industry-standard to ensure interoperability. Through information hiding, APIs enable modular programming, allowing users to use the interface independently of the implementation.

3.2.6 Google Colab

It is a free cloud service offered by Google along with its powerful GPUs. It helps in developing deep learning applications using popular libraries like Keras, Tensorflow, PyTorch and OpenCV.

3.3 Data Flow Diagrams

3.3.1 DFD -Level 0

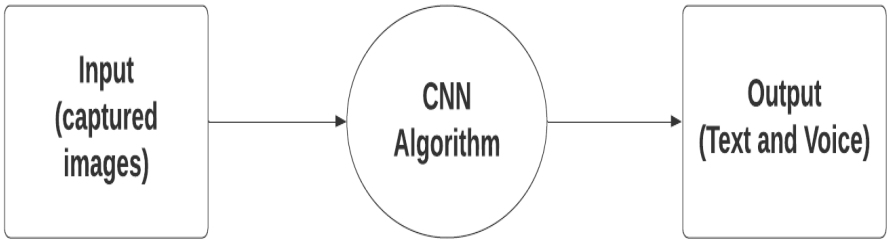

Figure 3.2: DFD Level-0

3.3.2 DFD Level-1

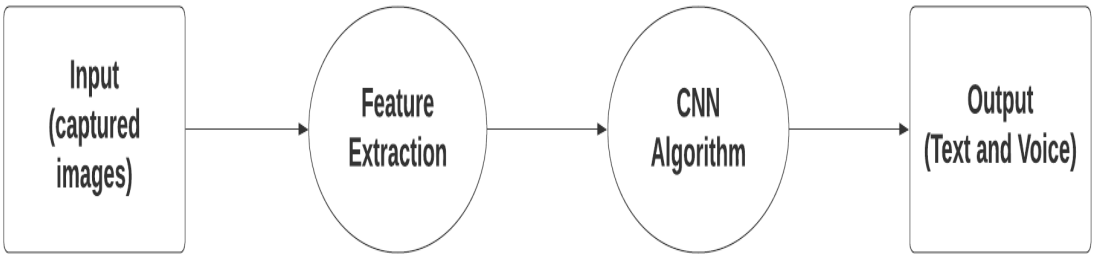

Figure 3.3: DFD Level-1

3.3.3 DFD Level-2

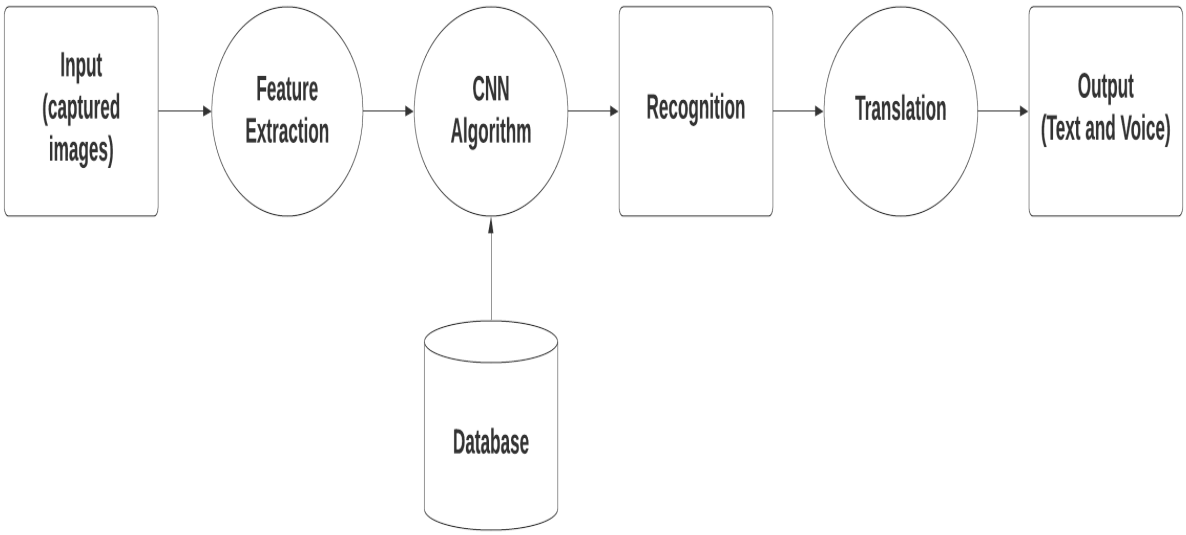

Figure 3.4: DFD Level-2

3.4 Implementation

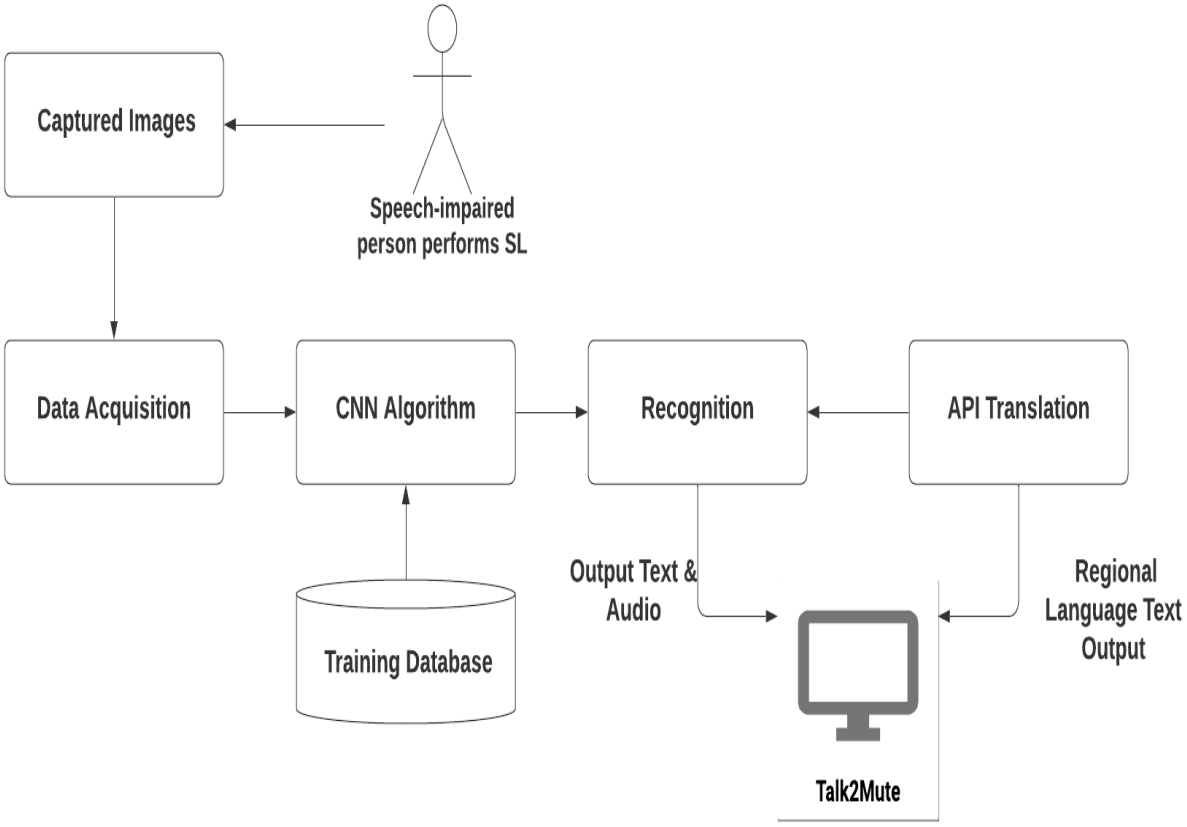

Figure 3.5: Architectural Diagram

3.4.1 Start with Google Colab

Colab is ideal for everything from improving your Python coding skills to working with deep learning libraries, like PyTorch, Keras, TensorFlow, and OpenCV. One can create notebooks in Colab, upload notebooks, store notebooks, share notebooks, mount your Google Drive. One can import most of their favorite directories, upload personal Jupyter Notebooks, upload notebooks directly from GitHub, upload Kaggle files, download your notebooks, and do just about everything else

3.4.2 Import Libraries

Used libraries are listed below:

- numpy
- hashlib
- random
- struct
- matplotlib
- opencv
- tensorflow

3.4.3 Upload the zip file

```
from google.colab import files
uploaded = files.upload()
```

3.4.4 Read the Uploaded Zip File

```
import zipfile
import io
data = zipfile.ZipFile(io.BytesIO(uploaded['filename'])
data.extractall()
```

3.4.5 Loading the Data

```
if__name__=='__main__':
  parser=argparse.ArgumentParser()
  parser.add_argument(
    '--image_dir',
    type=str,
```

```
        default='/content/drive/MyDrive/project/DataISL/train',
        help='path to folders of labeled images.'
        )
```

While loading the dataset, first the directory path in colab is to be specified. Here the image is resized to 200x200 so that the uniformity is maintained.

3.4.6 Training the model

```
  !python3 /content/train.py \
--bottleneck_dir=logs/bottlenecks \
--how_many_training_steps=2000 \
--model_dir=inception \
--summaries_dir=logs/training_summaries/basic \
--output_graph=logs/trained_graph.pb \
--output_labels=logs/trained_labels.txt \
--image_dir=/content/drive/MyDrive/project/dataset
```

We are using inception V3. These include things like tensor names and their sizes. if you want to adapt this script to work with another model you will need to update these to reflect the values in the network you are using. Data URL:'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz"

Bottleneck creation

```
def get_or_create_bottleneck(sess, image_lists, label_name,
index, image_dir,
               category, bottleneck_dir,
               jpeg_data_tensor,
               bottleneck_tensor):
```

The snippet of code retrieves or calculates bottleneck values for an image. If a cached version of the bottleneck data exists on-disk,return that,otherwise calculate the data and save it to disk for future use.

```
 label_lists = image_lists[label_name]
sub_dir = label_lists['dir']
sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
ensure_dir_exists(sub_dir_path)
bottleneck_path = get_bottleneck_path(image_lists,
```

```
label_name, index, bottleneck_dir, category)
if not os.path.exists(bottleneck_path):
with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
did_hit_error = False
try:
    bottleneck_values = [float(x) for x in
    bottleneck_string.split(',')]
except ValueError:
    print('Invalid float found, recreating bottleneck')
    did_hit_error = True
if did_hit_error:
    create_bottleneck_file(bottleneck_path,
    image_lists, label_name,
    index, image_dir, category, sess,
    jpeg_data_tensor,bottleneck_tensor)
    with open(bottleneck_path, 'r') as bottleneck_file:
        bottleneck_string = bottleneck_file.read()
    bottleneck_values = [float(x) for x in
    bottleneck_string.split(',')]
return bottleneck_values
```

Returns Numpy array of values produced by the bottleneck layer for the image. Allow exceptions to propagate here since they shouldn't happen after a fresh creation.

Training

Add the new layer that we'll be training.

```
(train_step, cross_entropy, bottleneck_input,
ground_truth_input, final_tensor) =
add_final_training_ops(len(image_lists.keys()),
       FLAGS.final_tensor_name,
       bottleneck_tensor)

if do_distort_images:
    (train_bottlenecks,
        train_ground_truth) =
        get_random_distorted_bottlenecks(
```

```
            sess, image_lists, FLAGS.train_batch_size,
            'training',FLAGS.image_dir,
            distorted_jpeg_data_tensor,
            distorted_image_tensor,
            resized_image_tensor, bottleneck_tensor)
    else:
        (train_bottlenecks,
            train_ground_truth, _) =
            get_random_cached_bottlenecks(
            sess, image_lists, FLAGS.train_batch_size,
            'training',
            FLAGS.bottleneck_dir, FLAGS.image_dir,
            jpeg_data_tensor,bottleneck_tensor)
```

Run the training for as many cycles as requested on the command line. Get a batch of input bottleneck values, either calculated fresh every time with distortions applied, or from the cache stored on disk.

```
validation_summary, validation_accuracy = sess.run(
    [merged, evaluation_step],
    feed_dict={bottleneck_input:
    validation_bottlenecks,
    validation_ground_truth})
    print('Step: %d, Train accuracy: %.4f%%,
    Cross entropy: %f,
    Validation accuracy: %.1f%% (N=%d)' % (i,
    train_accuracy * 100, cross_entropy_value,
    validation_accuracy * 100,
    len(validation_bottlenecks)))
```

Feed the bottlenecks and ground truth into the graph, and run a training step. Capture training summaries for TensorBoard with the ‘merged‘ op. After that run a validation step and capture training summaries for TensorBoard with the ‘merged‘ op.

```
test_bottlenecks, test_ground_truth, test_filenames=
(get_random_cached_bottlenecks(sess, image_lists,
'testing',LAGS.bottleneck_dir,
FLAGS.image_dir,jpeg_data_tensor,
bottleneck_tensor))
```

```
    test_accuracy, predictions = sess.run(
        [evaluation_step, prediction],
        feed_dict={bottleneck_input: test_bottlenecks,
        ground_truth_input: test_ground_truth})
    print('Final test accuracy = %.1f%% (N=%d)' % (
        test_accuracy * 100, len(test_bottlenecks)))
```

We've completed all our training, so run a final test evaluation on some new images we haven't used before.

3.4.7 Testing The Trained Model

```
image_path = sys.argv[1]
image_data = tf.gfile.FastGFile(image_path, 'rb').read()
label_lines = [line.rstrip() for line
            in tf.gfile.GFile("logs/trained_labels.txt")]
with tf.gfile.FastGFile("logs/trained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')
with tf.Session() as sess:
    softmax_tensor = sess.graph.get_tensor_by_name
    ('final_result:0')

 predictions = sess.run(softmax_tensor, \
          {'DecodeJpeg/contents:0': image_data})
 top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('%s (score = %.5f)' % (human_string, score))
```

The above section of code first read the image data. Load label file, strips off carriage return. Feed the image data as input to the graph and get first prediction. Sort to show labels of first prediction in order of confidence.

CHAPTER 4

RESULTS & DISCUSSION

4.1 Sign Language Recognition

The figures following shows the learning rate and training status of the work during implementation. The trained was completed within a time span between 7-8 hours.

Bottleneck was created in the training phase. A bottleneck layer is a layer that contains fewer nodes compared to the previous layers. It can be used to obtain a representation of the input with reduced dimensionality.

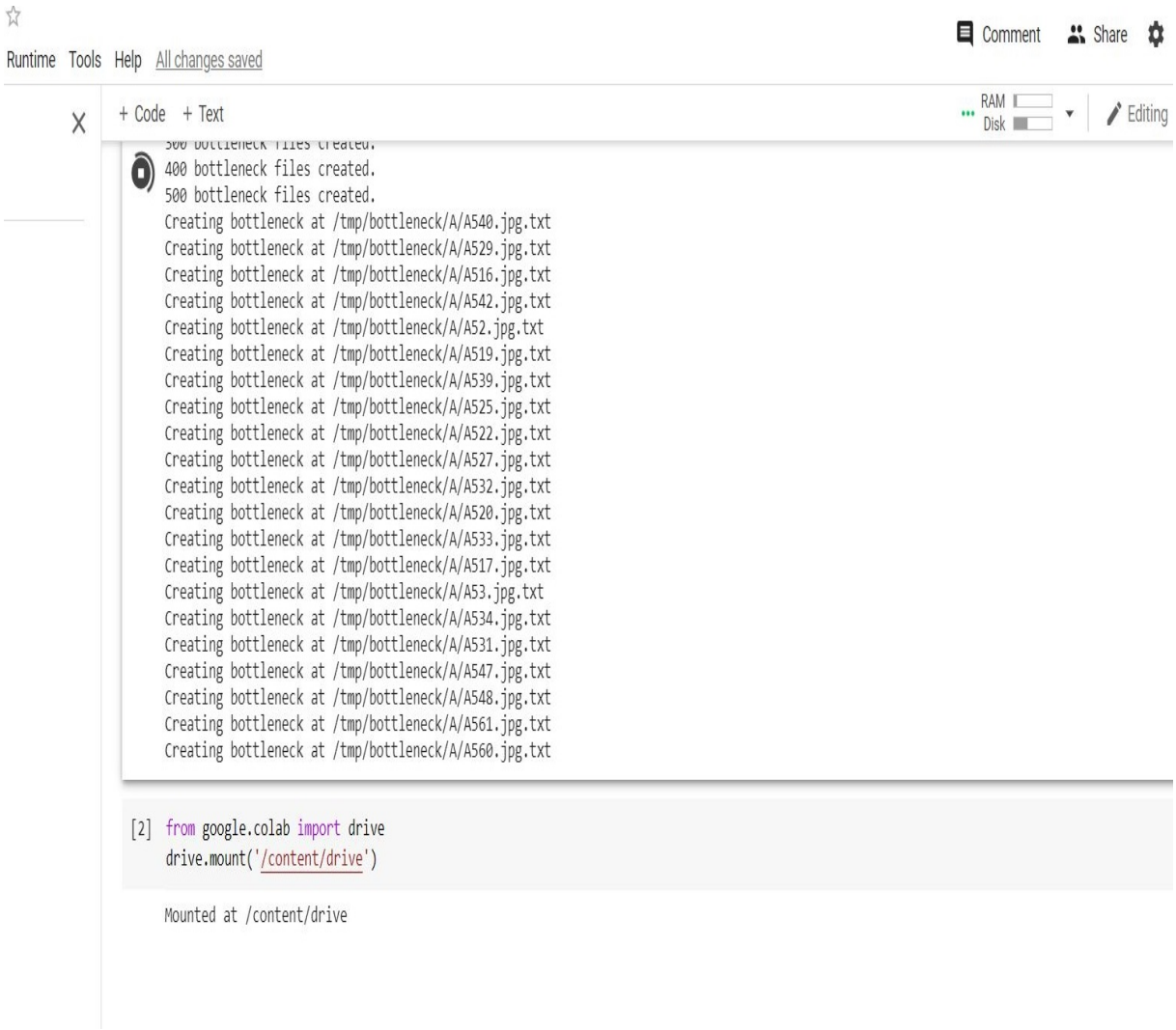

Figure 4.1: Bottleneck creation

4.1.1 Gesture Classification Results

This section works on the feature extraction and classification of the images. The feature extraction part is done with a convolutional neural network. The classification part is done using fully-connected and softmax layers. After the digital image is processed and classified. It undergoes traning and testing phase. After training and testing system will give out an output score for each image which will be a value. The accuracy of each alphabet can be found out. The image with the highest accuracy value is predicted as the output

```
[3] Step: 2400, Train accuracy: 93.0000%, Cross entropy: 0.517844, Validation accuracy: 97.0% (N=100)
    Step: 2500, Train accuracy: 97.0000%, Cross entropy: 0.358664, Validation accuracy: 92.0% (N=100)
    Step: 2600, Train accuracy: 93.0000%, Cross entropy: 0.501720, Validation accuracy: 96.0% (N=100)
    Step: 2700, Train accuracy: 97.0000%, Cross entropy: 0.332119, Validation accuracy: 96.0% (N=100)
    Step: 2800, Train accuracy: 99.0000%, Cross entropy: 0.374474, Validation accuracy: 94.0% (N=100)
    Step: 2900, Train accuracy: 98.0000%, Cross entropy: 0.336657, Validation accuracy: 95.0% (N=100)
    Step: 3000, Train accuracy: 93.0000%, Cross entropy: 0.429749, Validation accuracy: 98.0% (N=100)
    Step: 3100, Train accuracy: 98.0000%, Cross entropy: 0.319861, Validation accuracy: 94.0% (N=100)
    Step: 3200, Train accuracy: 96.0000%, Cross entropy: 0.326997, Validation accuracy: 100.0% (N=100)
    Step: 3300, Train accuracy: 97.0000%, Cross entropy: 0.286140, Validation accuracy: 97.0% (N=100)
    Step: 3400, Train accuracy: 98.0000%, Cross entropy: 0.344163, Validation accuracy: 94.0% (N=100)
    Step: 3500, Train accuracy: 96.0000%, Cross entropy: 0.354169, Validation accuracy: 94.0% (N=100)
    Step: 3600, Train accuracy: 96.0000%, Cross entropy: 0.297877, Validation accuracy: 99.0% (N=100)
    Step: 3700, Train accuracy: 98.0000%, Cross entropy: 0.300955, Validation accuracy: 99.0% (N=100)
    Step: 3800, Train accuracy: 100.0000%, Cross entropy: 0.225708, Validation accuracy: 98.0% (N=100)
    Step: 3900, Train accuracy: 96.0000%, Cross entropy: 0.331665, Validation accuracy: 98.0% (N=100)
    Step: 4000, Train accuracy: 95.0000%, Cross entropy: 0.288417, Validation accuracy: 99.0% (N=100)
    Step: 4100, Train accuracy: 97.0000%, Cross entropy: 0.272574, Validation accuracy: 99.0% (N=100)
    Step: 4200, Train accuracy: 98.0000%, Cross entropy: 0.271904, Validation accuracy: 93.0% (N=100)
    Step: 4300, Train accuracy: 97.0000%, Cross entropy: 0.266485, Validation accuracy: 98.0% (N=100)
    Step: 4400, Train accuracy: 98.0000%, Cross entropy: 0.294051, Validation accuracy: 100.0% (N=100)
    Step: 4500, Train accuracy: 99.0000%, Cross entropy: 0.237912, Validation accuracy: 99.0% (N=100)
    Step: 4600, Train accuracy: 100.0000%, Cross entropy: 0.260003, Validation accuracy: 97.0% (N=100)
    Step: 4700, Train accuracy: 97.0000%, Cross entropy: 0.237173, Validation accuracy: 96.0% (N=100)
    Step: 4800, Train accuracy: 98.0000%, Cross entropy: 0.251166, Validation accuracy: 98.0% (N=100)
    Step: 4900, Train accuracy: 97.0000%, Cross entropy: 0.241150, Validation accuracy: 99.0% (N=100)
    Step: 4999, Train accuracy: 97.0000%, Cross entropy: 0.256516, Validation accuracy: 96.0% (N=100)
    Final test accuracy = 97.8% (N=3077)
    WARNING:tensorflow:From <ipython-input-3-909b914abb6f>:848: convert_variables_to_constants (from tensorflow.python.framework.graph_util_impl)
    Instructions for updating:
```

Figure 4.2: Training Results

```
[ ] !python3 classify.py A81.jpg

WARNING:tensorflow:From classify.py:13: FastGFile._init_ (from tensorflow.python.platform.gfile) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
WARNING:tensorflow:From classify.py:18: The name tf.gfile.GFile is deprecated. Please use tf.io.gfile.GFile instead.

WARNING:tensorflow:From classify.py:22: The name tf.GraphDef is deprecated. Please use tf.compat.v1.GraphDef instead.

WARNING:tensorflow:From classify.py:26: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

a (score = 0.70958)
e (score = 0.07966)
b (score = 0.04266)
s (score = 0.03838)
j (score = 0.02038)
i (score = 0.01834)
t (score = 0.01634)
x (score = 0.01555)
o (score = 0.01215)
m (score = 0.01034)
n (score = 0.00917)
y (score = 0.00688)
z (score = 0.00579)
f (score = 0.00561)
w (score = 0.00151)
k (score = 0.00150)
r (score = 0.00103)
d (score = 0.00101)
u (score = 0.00085)
l (score = 0.00065)
g (score = 0.00059)
space (score = 0.00051)
c (score = 0.00035)
p (score = 0.00027)
```

Figure 4.3: Testing Accuracy

Here, the final values for each alphabet is obtained based on the test dataset. The programming language used for implementing the model is Python.The main goal is to identify the hand shown sign language gestures by the deaf and dumb people.Thus helping the person to get a faster and efficient way to communicate with the rest of the world.

	Classification
Accuracy	97.8%
Precision	90.2%

Table 4.1: Final Prediction

4.1.2 Sign Identification & Detection

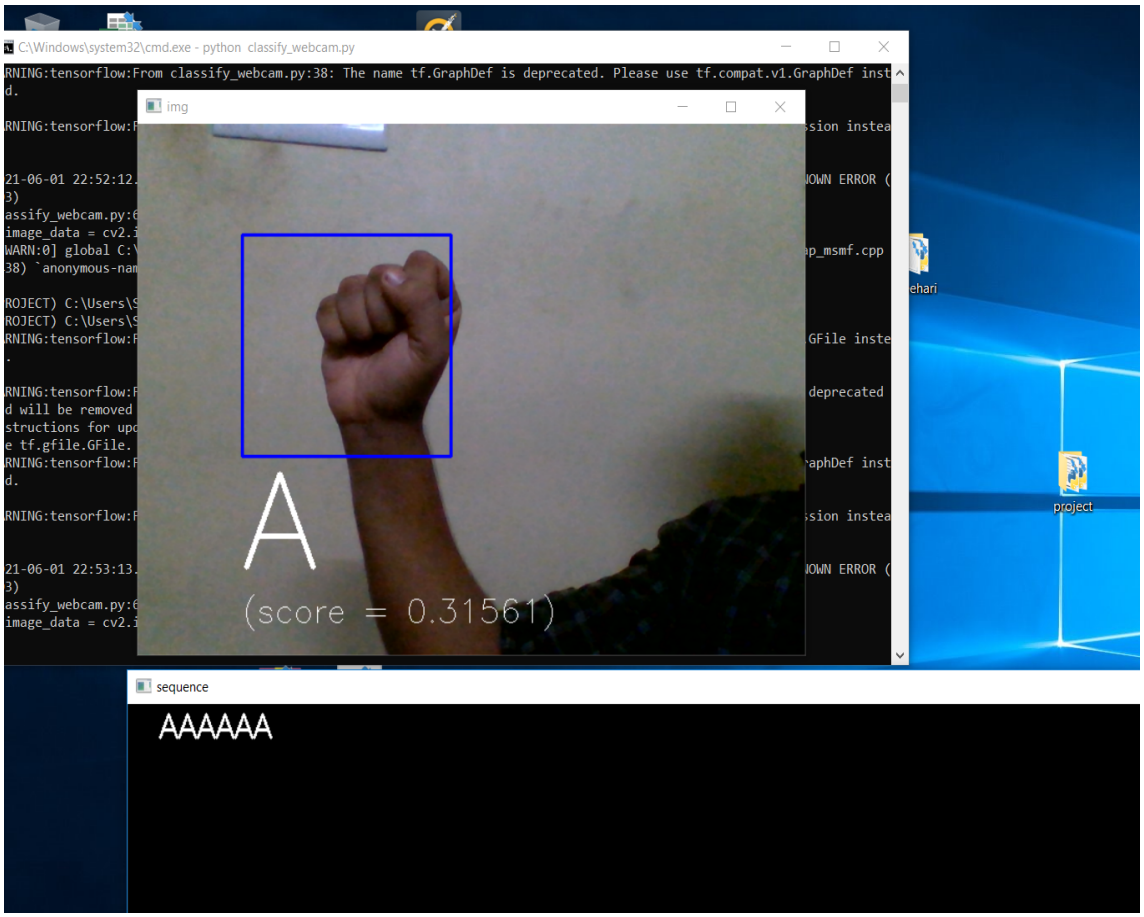

Figure 4.4: 'A' identification

The sign language gestures can be given as input through a webcam. It is important that the system should be efficient in identifying and classifying the alphabets according to their respective gestures. When you show the sign of an alphabet, it recognizes automatically, and display the corresponding text in the output text box. This section shows the results of alphabets being able to get identified by the classifier with real time input.

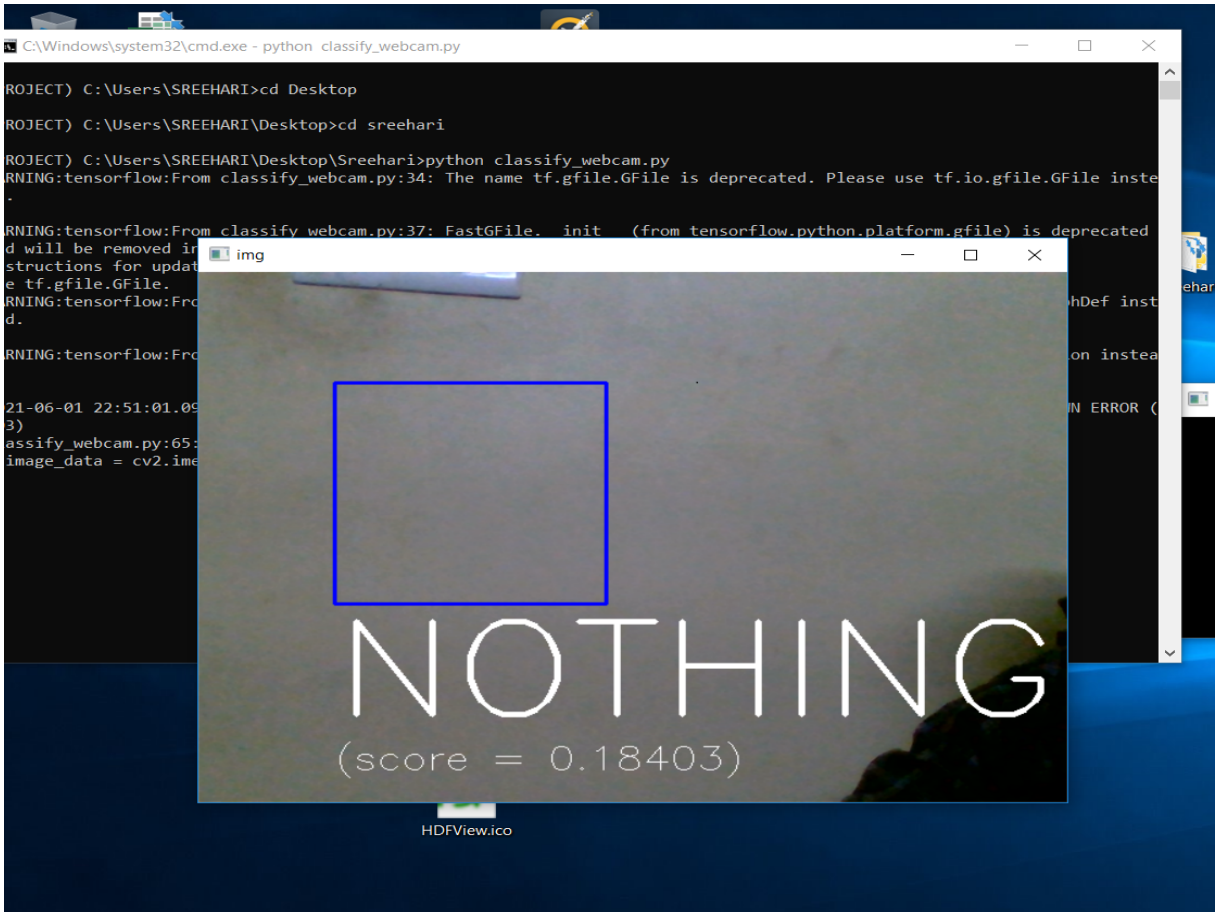

Figure 4.5: No sign detection

The system is efficient in detecting if no gestures are being shown in the region of interest. This section depicts the results when no hand is shown, system detects it and displays the message 'Nothing'. Thus the system gives an appreciable accuracy rate for both the above mentioned scenarios.

CHAPTER 5

CONCLUSION & FUTURE SCOPE

5.1 Conclusion

Deep learning techniques have been widely used in many applications such as health-care, big data analysis, etc. We use this deep learning techniques to detect and identify sign gestures, thus to overcome the limitations of the existing Sign Language Translation systems and provide an efficient solution to improve communication and support differently abled people in environments where they are unable to be accompanied by sign interpreters.

We have outlined the design of the proposed project, which aims to translates Sign Language to English and regional languages in the form of text and speech. Hence this system makes the communication between speech-impaired people and other simple and comfortable.

5.2 Future Scope

In future, the system can be improved by allowing multi-language to be displayed and converted to speech. The system can be extended to incorporate the learning of sign language for the people who wish to learn it with a visual guidance. A mobile version of the application will increase the reach to more people.

Bibliography

[1] Arsan, Taner and Ülgen, and Oğuz. *"Sign Language Converter"*. International Journal of Computer Science & Engineering Survey (IJCSES), pages 39-51.2015.

[2] Adewale, Victoria A and Olamiti, Adejoke O. *"Conversion of Sign Language to text and speech using Machine Learning techniques"*. Journal of Research and Review in Science, pages 58-65. 2018.

[3] NB, Mahesh Kumar *"Conversion of sign language into text"*. International Journal of Applied Engineering Research, pages 7154-7161. IEEE, 2018.

[4] Vedak, Omkar and Zavre, Prasad and Todkar, Abhijeet, Patil, and Manoj. *"Sign Language Interpreter using Image Processing and Machine Learning"*. International Research Journal of Engineering and Technology (IRJET), 2016.

[5] Han, Jungong and Shao, Ling and Xu, Dong and Shotton, Jamie. *"Enhanced computer vision with microsoft kinect sensor: A review"*. IEEE transactions on cybernetics, pages 1318-1334. IEEE, 2013

[6] Marques and Oge. *"Practical image and video processing using MATLAB "*. John Wiley & Sons, 2011.

[7] PYe, Jieping and Li, Qi. *"LDA/QR: an efficient and effective dimension reduction algorithm and its theoretical foundation"*. Pattern recognition, pages 851-854. Elsevier, 2004.

[8] Moryossef, Amit and Tsochantaridis, Ioannis and Aharoni, Roee and Ebling, Sarah, Narayanan, and Srini. *"Real-time sign language detection using human pose estimation"*. European Conference on Computer Vision, pages 237–248. Springer, 2020.

[9] Li, Huan and Chung, Fu-lai and Wang, and Shitong. *"A SVM based classification method for homogeneous data"*. Applied Soft Computing pages 228-235. Elsevier, 2015.

[10] Dutoit, Milos, Thierry and Cernak. *"TTSBOX: A MATLAB toolbox for teaching text-to-speech synthesis"*. International Conference on Acoustics, Speech, and Signal Processing. IEEE, 2005.

[11] Szegedy, Christian and Vanhoucke, Vincent and Ioffe, Sergey and Shlens, Jon and Wojna, Zbigniew. *"Rethinking the inception architecture for computer vision"*. Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818-2826. IEEE, 2016.

[12] Garcia, Brandon and Viesca, and Sigberto Alarcon. *"Real-time American sign language recognition with convolutional neural networks"*. Convolutional Neural Networks for Visual Recognition, pages 225-232. Stanford University, 2016.

APPENDIX A
SUMMARY OF THE RESULTS

	Classification
Accuracy	97.8%
Precision	90.2%

Table A.1: Summarization Table

APPENDIX B
SCREENSHOTS

```
def predict(image_data):

    predictions = sess.run(softmax_tensor, \
             {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    max_score = 0.0
    res = ''
    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        if score > max_score:
            max_score = score
            res = human_string
    return res, max_score

# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                   in tf.gfile.GFile("logs/trained_labels.txt")]

# Unpersists graph from file
with tf.gfile.FastGFile("logs/trained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    c = 0

    cap = cv2.VideoCapture(0)
```

Figure B.1: Code For Capturing Real Time Sign Gestures

Code For Training

```
rom __future__ import absolute_import
rom __future__ import division
rom __future__ import print_function

mport argparse
from datetime import datetime
mport hashlib
mport os.path
mport random
mport re
mport struct
mport sys
mport tarfile

mport numpy as np
rom six.moves import urllib
mport tensorflow as tf

rom tensorflow.python.framework import graph_util
rom tensorflow.python.framework import tensor_shape
rom tensorflow.python.platform import gfile
rom tensorflow.python.util import compat

LAGS = None

 These are all parameters that are tied to the particular model architecture
 we're using for Inception v3. These include things like tensor names and their
 sizes. If you want to adapt this script to work with another model, you will
 need to update these to reflect the values in the network you're using.
```

Figure B.2: (a)

```
result = {}
sub_dirs = [x[0] for x in gfile.Walk(image_dir)]
# The root directory comes first, so skip it.
is_root_dir = True
for sub_dir in sub_dirs:
    if is_root_dir:
        is_root_dir = False
        continue
    extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
    file_list = []
    dir_name = os.path.basename(sub_dir)
    if dir_name == image_dir:
        continue
    print("Looking for images in '" + dir_name + "'")
    for extension in extensions:
        file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
        file_list.extend(gfile.Glob(file_glob))
    if not file_list:
        print('No files found')
        continue
    if len(file_list) < 20:
        print('WARNING: Folder has less than 20 images, which may cause issues.')
    elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
        print('WARNING: Folder {} has more than {} images. Some images will '
              'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
    label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
    training_images = []
    testing_images = []
    validation_images = []
    for file_name in file_list:
        base_name = os.path.basename(file_name)
```

Figure B.3: (b)

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--image_dir',
        type=str,
        default='',
        help='Path to folders of labeled images.'
        )
    parser.add_argument(
        '--output_graph',
        type=str,
        default='logs/output_graph.pb',
        help='Where to save the trained graph.'
        )
    parser.add_argument(
        '--output_labels',
        type=str,
        default='logs/output_labels.txt',
        help='Where to save the trained graph\'s labels.'
        )
    parser.add_argument(
        '--summaries_dir',
        type=str,
        default='logs/retrain_logs',
        help='Where to save summary logs for TensorBoard.'
        )
    parser.add_argument(
        '--how_many_training_steps',
        type=int,
        default=5000,
```

Figure B.4: (c)

```
[3] Step: 2400, Train accuracy: 93.0000%, Cross entropy: 0.517844, Validation accuracy: 97.0% (N=100)
    Step: 2500, Train accuracy: 97.0000%, Cross entropy: 0.358664, Validation accuracy: 92.0% (N=100)
    Step: 2600, Train accuracy: 93.0000%, Cross entropy: 0.501720, Validation accuracy: 96.0% (N=100)
    Step: 2700, Train accuracy: 97.0000%, Cross entropy: 0.332119, Validation accuracy: 96.0% (N=100)
    Step: 2800, Train accuracy: 99.0000%, Cross entropy: 0.374474, Validation accuracy: 94.0% (N=100)
    Step: 2900, Train accuracy: 98.0000%, Cross entropy: 0.336657, Validation accuracy: 95.0% (N=100)
    Step: 3000, Train accuracy: 93.0000%, Cross entropy: 0.429749, Validation accuracy: 98.0% (N=100)
    Step: 3100, Train accuracy: 98.0000%, Cross entropy: 0.319861, Validation accuracy: 94.0% (N=100)
    Step: 3200, Train accuracy: 96.0000%, Cross entropy: 0.326997, Validation accuracy: 100.0% (N=100)
    Step: 3300, Train accuracy: 97.0000%, Cross entropy: 0.286140, Validation accuracy: 97.0% (N=100)
    Step: 3400, Train accuracy: 98.0000%, Cross entropy: 0.344163, Validation accuracy: 94.0% (N=100)
    Step: 3500, Train accuracy: 96.0000%, Cross entropy: 0.354169, Validation accuracy: 94.0% (N=100)
    Step: 3600, Train accuracy: 96.0000%, Cross entropy: 0.297877, Validation accuracy: 99.0% (N=100)
    Step: 3700, Train accuracy: 98.0000%, Cross entropy: 0.300955, Validation accuracy: 99.0% (N=100)
    Step: 3800, Train accuracy: 100.0000%, Cross entropy: 0.225708, Validation accuracy: 98.0% (N=100)
    Step: 3900, Train accuracy: 96.0000%, Cross entropy: 0.331665, Validation accuracy: 98.0% (N=100)
    Step: 4000, Train accuracy: 95.0000%, Cross entropy: 0.288417, Validation accuracy: 99.0% (N=100)
    Step: 4100, Train accuracy: 97.0000%, Cross entropy: 0.272574, Validation accuracy: 99.0% (N=100)
    Step: 4200, Train accuracy: 98.0000%, Cross entropy: 0.271904, Validation accuracy: 93.0% (N=100)
    Step: 4300, Train accuracy: 97.0000%, Cross entropy: 0.266485, Validation accuracy: 98.0% (N=100)
    Step: 4400, Train accuracy: 98.0000%, Cross entropy: 0.294051, Validation accuracy: 100.0% (N=100)
    Step: 4500, Train accuracy: 99.0000%, Cross entropy: 0.237912, Validation accuracy: 99.0% (N=100)
    Step: 4600, Train accuracy: 100.0000%, Cross entropy: 0.260003, Validation accuracy: 97.0% (N=100)
    Step: 4700, Train accuracy: 97.0000%, Cross entropy: 0.237173, Validation accuracy: 96.0% (N=100)
    Step: 4800, Train accuracy: 98.0000%, Cross entropy: 0.251166, Validation accuracy: 98.0% (N=100)
    Step: 4900, Train accuracy: 97.0000%, Cross entropy: 0.241150, Validation accuracy: 99.0% (N=100)
    Step: 4999, Train accuracy: 97.0000%, Cross entropy: 0.256516, Validation accuracy: 96.0% (N=100)
    Final test accuracy = 97.8% (N=3077)
    WARNING:tensorflow:From <ipython-input-3-909b914abb6f>:848: convert_variables_to_constants (from tensorflow.python.framework.graph_util_impl) :
    Instructions for updating:
```

Figure B.5: Training Result

Code For Testing

```
import tensorflow as tf
import sys
import os


# Disable tensorflow compilation warnings
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
import tensorflow as tf

image_path = sys.argv[1]

# Read the image_data
image_data = tf.gfile.FastGFile(image_path, 'rb').read()


# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                   in tf.gfile.GFile("logs/trained_labels.txt")]

# Unpersists graph from file
with tf.gfile.FastGFile("logs/trained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(softmax_tensor, \
```

Figure B.6: (a)

```
image_path = sys.argv[1]

# Read the image_data
image_data = tf.gfile.FastGFile(image_path, 'rb').read()


# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                   in tf.gfile.GFile("logs/trained_labels.txt")]

# Unpersists graph from file
with tf.gfile.FastGFile("logs/trained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')

with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(softmax_tensor, \
             {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('%s (score = %.5f)' % (human_string, score))
```

Figure B.7: (b)

```
[ ] !python3 classify.py A81.jpg

WARNING:tensorflow:From classify.py:13: FastGFile.__init__ (from tensorflow.python.platform.gfile) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
WARNING:tensorflow:From classify.py:18: The name tf.gfile.GFile is deprecated. Please use tf.io.gfile.GFile instead.

WARNING:tensorflow:From classify.py:22: The name tf.GraphDef is deprecated. Please use tf.compat.v1.GraphDef instead.

WARNING:tensorflow:From classify.py:26: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

a (score = 0.70958)
e (score = 0.07966)
b (score = 0.04266)
s (score = 0.03838)
j (score = 0.02038)
i (score = 0.01834)
t (score = 0.01634)
x (score = 0.01555)
o (score = 0.01215)
m (score = 0.01034)
n (score = 0.00917)
y (score = 0.00688)
z (score = 0.00579)
f (score = 0.00561)
w (score = 0.00151)
k (score = 0.00150)
r (score = 0.00103)
d (score = 0.00101)
u (score = 0.00085)
l (score = 0.00065)
g (score = 0.00059)
space (score = 0.00051)
c (score = 0.00035)
p (score = 0.00027)
```

Figure B.8: Testing Result