SIGN LANGUAGE TRANSLATOR USING AI

A MINI PROJECT REPORT

Submitted by

SURYA PRAKASH.K	**19104079**
VIJAYA KUMAR.G	**19104087**
YEGAPPAN.S	**19104090**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ERODE SENGUNTHAR ENGINEERING COLLEGE

(An Autonomous Institution)

PERUNDURAI, ERODE – 638 057

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report **"SIGN LANGUAGE TRANSLATOR USING AI"** is the bonafide work of "**SURYA PRAKASH.K (ES19CS79), VIJAYA KUMAR.G (ES19CS87), YEGAPPAN.S (ES19CS90),"** Who carried out the project work under my supervision. Certified further that to the best of my knowledge the reported here in does not from part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Ms.D. SATHYA M.E**,**

SUPERVISOR

Assistant Professor,

Department of Computer Science and Engineering,

Erode Sengunthar Engineering College, Erode-638 057.

SIGNATURE

Dr.G. SIVAKUMAR M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Professor and Head,

Department of Computer Science and Engineering,

Erode Sengunthar Engineering College, Erode-638 057.

Submitted for End Semester Mini Project Viva -Voce Examination held on_____.

INTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all we would like to convey our heartfelt thanks to our respected Founder, **"UDYOG RATTAN" Deivathiru J. SUDHANANDHEN, B.Sc.,** who always blessed us to give the best.

We are delighted to thank our esteemed Correspondent **Thiru.G.KAMALAMURUGAN,B.B.M.,** of Erode Sengunthar Educational Trust and gave us full support for carrying out our project in adequate time and providing us with facilities.

We have immense pleasure to thank our Secretary **Thiru. S.N.THANGARAJU,B.E., M.B.A.,** of Erode Sengunthar Educational Trust who gave us full support and guide in all the endeavours of our project.

We would like to express our thanks to our Principal **Dr.V.VENKATACHALAM, B.E.,M.S.,M.Tech.,Ph.D.,** for forwarding us to do our project and offering adequate duration in completing our project.

We render our sincere thanks to, **Dr.G.SIVAKUMAR,M.E.,Ph.D., Professor and Head**, Department of Computer Science and Engineering for his anchoring support in doing this project.

We are heartly extend our thanks to our project guide **Ms.D.SATHYA M.E, Assistant Professor** for her support, constant supervision and as well as for providing necessary information regarding the project.

ABSTRACT

Sign languages are the primary language of many people worldwide. To overcome communication barriers between the Deaf and the hearing community, artificial intelligence technologies have been employed, aiming to develop systems for automated sign language recognition and generation. Particularities of sign languages have to be considered though sharing some characteristics of spoken languages since they differ in others. AI algorithms have paved the way for the development of various applications aiming at fulfilling the needs of deaf and hearing-impaired communities. To this end, this survey aims to provide a comprehensive review of state-of-the-art methods in sign language capturing, recognition, and translation**.** In addition, the survey presents a number of applications, while it discusses the main challenges in the field of sign language technologies. Future research direction are also proposed in order to assist prospective researchers towards further advancing the field.

Keywords: sign language recognition; sign language representation; sign language capturing; applications

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Sign language (SL) is the main means of communication between hearing-impaired people and other communities and it is expressed through manual (i.e., body and hand motions) and non-manual (i.e., facial expressions) features. These features are combined together to form utterances that convey the meaning of words or sentences . Being able to capture and understand the relation between utterances and words is crucial for the Deaf community in order to guide us to an era where the translation between utterances and words can be achieved automatically. The research community has long identified the need for developing sign language technologies to facilitate the communication and social inclusion of hearing-impaired people. Although the development of such technologies can be really challenging due to the existence of numerous sign languages and the lack of large annotated datasets, the recent advances in AI and machine learning have played a significant role towards automating and enhancing such technologies. Sign language technologies cover a wide spectrum, ranging from the capturing of signs to their realistic representation in order to facilitate the communication between hearing-impaired people, as well as the communication between hearing-impaired and speaking people. More specifically, sign language capturing involves the accurate extraction of body, hand and mouth expressions using appropriate sensing devices in marker-less or marker-based setups. The accuracy of sign language capturing technologies is currently limited by the resolution and discrimination ability of sensors and the fact that occlusions and fast hand movements pose significant challenges to the accurate capturing of signs. Sign language recognition (SLR) involves the development of powerful machine learning algorithms to robustly classify human articulations to isolated signs or continuous sentences. Current limitations in SLR lie in the lack of large annotated datasets that greatly affect the accuracy and generalization ability of SLR methods, as well as the difficulty in identifying sign boundaries in continuous SLR scenarios.

CHAPTER 2

LITERATURE SURVEY

The researches done in this field are mostly done using a glove based system. In the glove based system, sensors such as potentiometer, accelerometers etc. are attached to each of the finger. Based on their readings the corresponding alphabet is displayed. Christopher Lee and Yangsheng Xu developed a glove-based gesture recognition system that was able to recognize 14 of the letters from the hand alphabet, learn new gestures and able to update the model of each gesture in the system in online mode. Over the years advanced glove devices have been designed such as the Sayre Glove, Dexterous Hand Master and Power Glove. The main problem faced by this gloved based system is that it has to be recalibrate every time whenever a new user on the finger-tips so that the fingertips are identified by the Image Processing unit. We are implementing our project by using Image Processing. The main advantage of our project is that it is not restricted to be used with black background. It can be used with any background. Also wearing of color bands is not required in our system. By that, securities could authorize an individuals identity depending on who she is, and not what she has and what she could remember. Two main classes can be found in biometrics:

Physiological It is associated with the body shape, includes all physical traits, iris, palm print, facial features, Fingerprints, etc.

Behavioral Related to the behavioral characteristics of a person. A characteristic widely used till today is signatures. Modern methods of behavioral studies are emerging such as keystroke dynamics and voice analysis.

Deaf Mute Communication Interpreter- A Review : This paper aims to cover the various preailing methods of deaf-mute communication interpreter system. The two broad classification of the communication methodologies used by the deaf mute people are – Wearable Communication Device and Online Learning System. Under Wearable communication method, there are Glove based system, Keypad method and Handicom Touch-screen. All the above mentioned three sub- divided methods make use of various sensors, accelerometer, a suitable micro-controller, a text to speech conversion module, a keypad and a touch-screen. The need for an external device to interpret the message between a deaf mute and non-deaf-mute people can be overcome by the second method i.e., online learning system. The Online Learning System has different methods. The five subdivided methods are- SLIM module, TESSA, Wi-See Technology, SWI_PELE System and Web- Sign Technology.

An Efficient Framework for Indian Sign Language Recognition Using Wavelet Transform : The proposed ISLR system is considered as a pattern recognition technique

that has two important modules: feature extraction and classification. The joint use of Discrete Wavelet Transform (DWT) based feature extraction and nearest neighbor classifier is used to recognize the sign language. The experimental results show that the proposed hand gesture recognition system achieves maximum 99.23% classification accuracy while using cosine distance classifier.

Hand Gesture Recognition Using PCA in [3]: In this paper authors presented a scheme using a database driven hand gesture recognition based upon skin color model approach and thresholding approach along with an effective template matching with can be effectively used for human robotics applications and similar other application. Initially, hand region is segmented by applying skin color model in YCbCr color space. In the next stage thresholding is applied to separate foreground and background. Finally, template based matching technique is developed using Principal Component Analysis (PCA) for recognition.

Hand Gesture Recognition System For the Dumb People [4]: Authors presented the static hand gesture recognition system using digital image processing. For hand gesture feature vector SIFT algorithm is used. The SIFT features have been computed at the edges which are invariant to scaling, rotation, addition of noise.

An Automated System for Indian Sign Language Recognition in [5]: In this paper a method for automatic recognition of signs on the basis of shape-based features is presented. For segmentation of hand region from the images, Otsus thresholding algorithm is used, that chooses an optimal threshold to minimize the within-class variance of threshold black and white pixels. Features of segmented hand region are calculated using Hus invariant moments that are fed to Artificial Neural Network for classification. Performance of the system is evaluated on the basis of Accuracy, Sensitivity and Specificity.

Hand Gesture Recognition for Sign Language Recognition: A Review in [6]: Authors presented various method of hand gesture and sign language recognition proposed in the past by various researchers. For deaf and dumb people, Sign language is the only way of communication. With the help of sign language, these physical impaired people express their emotions and thoughts to other.

Design Issue and Proposed Implementation of Communication Aid for Deaf & Dumb People in: In this paper author proposed a system to aid communication of deaf and dumb people communication using Indian sign language (ISL) with normal people where hand gestures will be converted into appropriate text message. Main objective is to design an algorithm to convert dynamic gesture to text at real time finally after testing is done the system will be implemented on android platform and will be available as an application for smart phone and tablet pc.

Real Time Detection and Recognition of Indian and American Sign Language: Author proposed a real time vision-based system for hand gesture recognition for human computer interaction in many applications. The system can recognize 35 different hand

gestures given by Indian and American Sign Language or ISL and ASL at faster rate with virtuous accuracy. RGB-to-GRAY segmentation technique was used to minimize the chances of false detection. Authors proposed a method of improvised Scale Invariant Feature Transform (SIFT) and same was used to extract features. The system is model using MATLAB. To design and efficient user-friendly hand gesture recognition system, a GUI model has been implemented.

A Review on Feature Extraction for Indian and American Sign Language in [9]: Paper presented the recent research and development of sign language based on manual communication and body language. Sign language recognition system typically elaborate three steps preprocessing, feature extraction and classification. Classification methods used for recognition are Neural Network (NN), Support Vector Machine (SVM), Hidden Markov Models (HMM), Scale Invariant Feature Transform (SIFT), etc.

Sign Pro-an Application Suite for Deaf and Dumb. in [10]: Author presented application that helps the deaf and dumb person to communicate with the rest of the world using sign language. The key feature in this system is the real time gesture to text conversion. The processing steps include: gesture extraction, gesture matching and conversion to speech. Gesture extraction involves use of various image processing techniques such as ISSN No.: 2454- 2024 (online) International Journal of Technical Research & Science pg. 433 www.ijtrs.com www.ijtrs.org Paper Id: IJTRS-V2-I7-005 Volume 2 Issue VII, August 2017 @2017, IJTRS All Right Reserved histogram matching, bounding box computation, skin color segmentation and region growing. Techniques applicable for Gesture matching include feature point matching and correlation-based matching. The other features in the application include voicing out of text and text to gesture conversion.

Offline Signature Verification Using Surf Feature Extraction and Neural Networks Approach: In this paper, off-line signature recognition & verification using neural network is proposed, where the signature is captured and presented to the user in an image format.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

- Internet Connection
- 100MB Hard Disk Space
- 1GB RAM Memory
- HD 720(1200*720)

3.2 SOFTWARE REQUIREMENTS

- Windows 7, Windows 8, Windows 10 or Linux 2.6.8 or greater
- Python 3.7 or Greater
- Jupiter Notebook
- OpenCV
- Tensor Flow

CHAPTER 4

DOMAIN & TECHNOLOGY

ARTIFICIAL INTELLIGENCE (AI)

Artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks as, for example, discovering proofs for mathematical theorems or playing chess with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition.

All but the simplest human behaviour is ascribed to intelligence, while even the most complicated insect behaviour is never taken as an indication of intelligence. What is the difference? Consider the behaviour of the digger wasp, Sphex. When the female wasp returns to her burrow with food, she first deposits it on the threshold, checks for intruders inside her burrow, and only then, if the coast is clear, carries her food inside. The real nature of the wasp’s instinctual behaviour is revealed if the food is moved a few

inches away from the entrance to her burrow while she is inside: on emerging, she will repeat the whole procedure as often as the food is displaced. Intelligence conspicuously absent in the case of Sphex must include the ability to adapt to new circumstances.

There are a number of different forms of learning as applied to artificial intelligence. The simplest is learning by trial and error. For example, a simple computer program for solving mate-in-one chess problems might try moves at random until mate is found. The program might then store the solution with the position so that the next time the computer encountered the same position it would recall the solution. This simple memorizing of individual items and procedures known as rote learning is relatively easy to implement on a computer. More challenging is the problem of implementing what is called generalization. Generalization involves applying past experience to analogous new situations. For example, a program that learns the past tense of regular English verbs by rote will not be able to produce the past tense of a word such as jump unless it previously had been presented with jumped, whereas a program that is able to generalize can learn the "add ed" rule and so form the past tense of jump based on experience with similar verbs.

CONVOLUTIONAL NEURAL NETWORK

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm a Convolutional Neural Network.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification

algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

TENSORFLOW

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java.[This flexibility lends itself to a range of applications in many different sectors.TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools,

libraries, and community resources that lets researchers push the state-ofthe-art in ML, and gives developers the ability to easily build and deploy ML-powered applications.

TensorFlow provides a collection of workflows with intuitive, highlevel APIs for both beginners and experts to create machine learning models in numerous languages. Developers have the option to deploy models on a number of platforms such as on servers, in the cloud, on mobile and edge devices, in browsers, and on many other JavaScript platforms. This enables developers to go from model building and training to deployment much more easily. NUMPY

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

NumPy is often used along with packages like SciPy (Scientific Python) and Mat−plotlib (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

CHAPTER 5

WORKING PRINCIPLE

ALGORITHM

Step 1: Take picture of the hand to be tested using a webcam.

Step 2: Convert the image into Grayscale.

Step 3: Convert the Grayscale image into a binary image. Set a threshold so that the pixels that are above certain intensity are set to white and those below are set to black.

Step 4: From the binary image, we generate the coordinates of the image.

Step 5: These coordinates are then compared with stored co-ordinates in the database for the purpose of output generation using pattern matching technique.

Step 6: We need to use a pattern matching algorithm for this purpose.

Step 7: If the pattern is matched, the alphabet corresponding to the image is displayed .

LIST OF MODULES

- Sign Language Capturing
- Datasets
- Continuous Sign Language Recognition Datasets
- Isolated Sign Language Recognition Datasets
- Sign Language Recognition

- Sign Language Translation

Sign Language Capturing

Sign language capturing involves the recording of sign gestures using appropriate sensor setups. The purpose is to capture discriminative information from the signs that will allow the study, recognition and 3D representation of signs at later stages. Moreover, sign language capturing enables the construction of large datasets that can be used to accurately train and evaluate machine learning sign language recognition and translation algorithms.

Datasets

Datasets are crucial for the performance of methodologies regarding sign language recognition, translation and synthesis and as a result a lot of attention has been drawn towards the accurate capturing of signs and their meticulous annotation. The majority of the existing publicly available datasets are captured with visual sensors and are presented below.

Hello Dataset

Hello Dataset

Fig-5.1

Hello.XML

```
<annotation>
	<folder>collectedimages</folder>
	<filename>hello.4a1db1ba-b4ea-11ec-8b5e-9c7bef6ca9d7.jpg</filename>
	<path>C:\Users\yegappan\RealTimeObjectDetection\Tensorflow\workspace\images\co
llectedimages\hello.4a1db1ba-b4ea-11ec-8b5e-9c7bef6ca9d7.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>640</width>
		<height>480</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>Hello</name>
		<pose>Unspecified</pose>
		<truncated>1</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>344</xmin>
			<ymin>161</ymin>
			<xmax>640</xmax>
			<ymax>480</ymax>
		</bndbox>
	</object>
</annotation>
```

Yes Dataset

Yes Dataset

Fig-5.2

Yes.XML

<annotation>
<folder>collectedimages</folder>
<filename>yes.99bf5693-b4ea-11ec-ad7a-9c7bef6ca9d7.jpg</filename>
<path>C:\Users\yegappan\RealTimeObjectDetection\Tensorflow\workspace\images\collectedimages\yes.99bf5693-b4ea-11ec-ad7a-9c7bef6ca9d7.jpg</path>
<source>
<database>Unknown</database>
</source>
<size>
<width>640</width>
<height>480</height>

```
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Yes</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>269</xmin>
            <ymin>230</ymin>
            <xmax>557</xmax>
            <ymax>443</ymax>
        </bndbox>
    </object>
</annotation>
```

No Dataset

No Dataset

Fig-5.3

No.XML

```
<annotation>
	<folder>collectedimages</folder>
	<filename>no.b204f256-b4ea-11ec-b64b-9c7bef6ca9d7.jpg</filename>
	<path>C:\Users\yegappan\RealTimeObjectDetection\Tensorflow\workspace\images\collectedimages\no.b204f256-b4ea-11ec-b64b-9c7bef6ca9d7.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>640</width>
		<height>480</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>No</name>
		<pose>Unspecified</pose>
		<truncated>1</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>1</xmin>
			<ymin>248</ymin>
			<xmax>297</xmax>
			<ymax>480</ymax>
		</bndbox>
	</object>
</annotation>
```

Peace Dataset

Peace Dataset

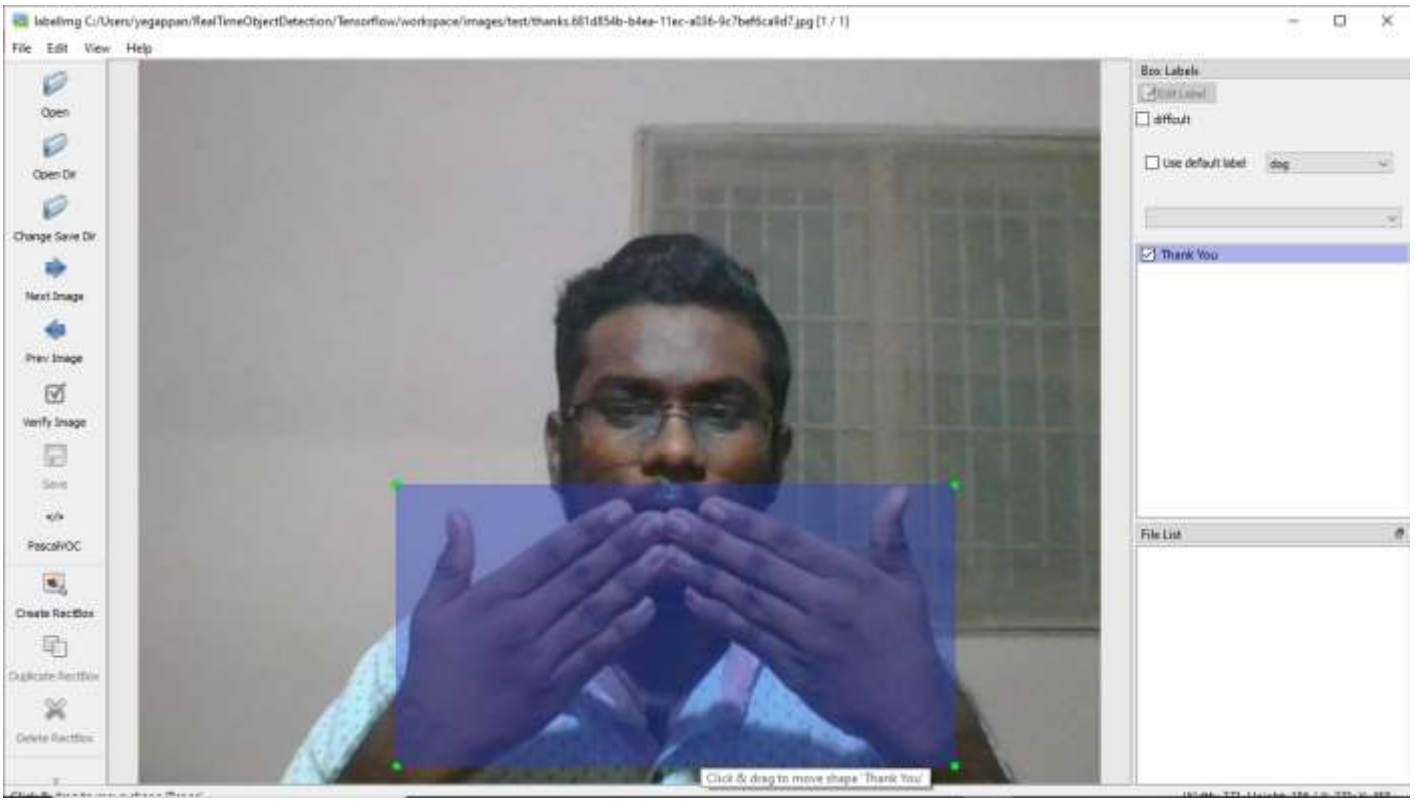
Fig-5.4

Peace.XML

```
<annotation>
	<folder>collectedimages</folder>
	<filename>peace.0875c2d2-b4eb-11ec-9c61-9c7bef6ca9d7.jpg</filename>
	<path>C:\Users\yegappan\RealTimeObjectDetection\Tensorflow\workspace\images\collectedimages\peace.0875c2d2-b4eb-11ec-9c61-9c7bef6ca9d7.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>640</width>
		<height>480</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
```

```
        <object>
                <name>Peace</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>208</xmin>
                        <ymin>120</ymin>
                        <xmax>484</xmax>
                        <ymax>478</ymax>
                </bndbox>
        </object>
</annotation>
```

Thankyou Dataset



Thankyou Dataset

Fig-5.5

Thankyou.XML

```
<annotation>
	<folder>collectedimages</folder>
	<filename>thanks.681d854b-b4ea-11ec-a036-9c7bef6ca9d7.jpg</filename>
	<path>C:\Users\yegappan\RealTimeObjectDetection\Tensorflow\workspace\images\co
llectedimages\thanks.681d854b-b4ea-11ec-a036-9c7bef6ca9d7.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>640</width>
		<height>480</height>
		<depth>3</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>Thank You</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>172</xmin>
			<ymin>282</ymin>
			<xmax>543</xmax>
			<ymax>468</ymax>
		</bndbox>
	</object>
</annotation>
```

Continuous Sign Language Recognition Datasets

Continuous sign language recognition (CSLR) datasets contain videos of sequences of signs instead of individual signs and are more suitable for developing real-life applications. Phoenix-2014 [27] is one of the most popular CSLR dataset with recordings of weather forecasts in German sign language. All videos were recorded with 9 signers at a frame rate of 25 frames per second. The dictionary has 1081 unique glosses and the dataset contains 5672 videos for training, 540 videos for validation and 629 videos for testing. The same authors created an updated version of Phoenix-2014, called Phoenix-2014-T [28], with spoken language translations, which makes it appropriate for both CSLR and sign language translation experiments. It contains 8257 videos from 9 different signers performing 1088 unique signs and 2887 unique words. Although all recordings are performed in a controlled environment, Phoenix-2014 and Phoenix-2014-T are both challenging datasets with large vocabularies and varying number of samples per sign with a few signs having a single sample. Similarly, BSL-1K [29] contains video recordings from British news broadcasts, along with automatically extracted annotations from provided subtitles. It is a large database with 273,000 samples from 40 signers that is also used for sign language segmentation. Another notable dataset is CSL [30,31] that contains Chinese words widely used in daily communication. The dataset has 100 sentences with signs that were performed from 50 signers. The recordings are performed in a lab with predefined conditions (i.e., background, lighting). The vocabulary size is 178 words that are performed multiple times, resulting in high recognition results achieved by SLR methods. GRSL [15] is another CSLR dataset of Greek sign language that is used in home care services, which contains multiple modalities, such as RGB, depth and skeletal joints. On the other hand, GSL [17] is a large Greek sign language dataset created to assist communication of Deaf people with public service employees. The dataset was created with a RealSense D435 sensor that records both RGB and depth information. Furthermore, it contains both continuous and isolated sign videos from 15 predefined scenarios. It is recorded on a laboratory environment, where each scenario is repeated five consecutive times.

Isolated Sign Language Recognition Datasets

Isolated sign language recognition (ISLR) datasets are important for identifying and learning discriminative features for sign language recognition. CSL-500 is the isolated version of CSL but it contains 500 unique glosses

performed from the same 50 signers. CSLR methods usually adopt this dataset for feature learning prior to finetuning on the CSL dataset. MS-ASL is another widely employed ISLR dataset with 1000 unique American sign language glosses. It contains recordings collected from YouTube platform from 222 signers with a large variance in background settings, which makes this dataset suitable for training complex methods with strong representation capabilities. Similarly, WASL is an ISLR dataset with 2000 unique American sign glosses performed by 119 signers. The videos have different background and illumination conditions, which makes it a challenging ISLR benchmark dataset. On the other hand, AUTSL is a Turkish sign language dataset captured under various indoor and outdoor backgrounds, while LSA64 is an Argentinian sign language dataset that includes 3200 videos, in which 10 non-expert subjects execute 5 repetitions of 64 different types of signs. LSA64 is a small and relatively easy dataset, where SLR methods achieve outstanding recognition performance. Finally, IsoGD is a gesture recognition dataset that consists of 47,933 RGB-D videos performed by 21 different individuals and contains 249 gesture labels. Although IsoGD is a gesture recognition dataset, its large size and challenging illumination and background conditions allows the training of highly accurate ISLR methods.

Sign Language Recognition

Sign language recognition (SLR) is the task of recognizing sign language glosses from video streams. It is a very important research area since it can bridge the communication gap between hearing and Deaf people, facilitating the social inclusion of hearing-impaired people. Moreover, sign language recognition can be classified into isolated and continuous based on whether the video streams contain an isolated gloss or a gloss sequence that corresponds to a sentence.

Sign Language Translation

Sign Language Translation is the task of translating videos with sign language into spoken language by modeling not only the glosses but also the language structure and grammar. It is an important research area that facilitates the communication between the Deaf and other communities. Moreover, the SLT task is more challenging compared to CSLR due to the additional linguistic rules and the representation of spoken languages. SLT methods are usually evaluated using the bilingual evaluation understudy (BLEU) metric [92]. BLEU is a translation quality score that evaluates the correspondence between the predicted translation and the ground truth text. More specifically, BLEU-n measures the n-gram overlap between the output and the reference sentences. BLEU-1,2,3,4 are reported to provide a clear view of the actual translation performance of a method.

Camgoz et al. in , adopted an attention-based neural machine translation architecture for SLT. The encoder consisted of a 2D-CNN and an LSTM network, while the decoder consists of word embeddings with an attention LSTM. The authors stated that the method is prone to errors when spoken words are not explicitly signed in the video but inferred from the context. Their method set the baseline performance on Phoenix-2014-T with a BLEU-4 score of 18.4. Orbay et al. in, compared different gloss tokenization methods using either 2D-CNN, 3D-CNN, LSTM or Transformer networks. In addition, they investigated the importance of using full frames compared to hand images as the first provide useful information regarding the face and arms of the signer for SLT. On the other hand, Ko et al. in [94], utilized human keypoints extracted from the video, which were then fed to a recurrent encoder-decoder network for sign language translation. Furthermore, the skeletal features were extracted with OpenPose and then normalized to improve the overall performance. Then, they were fed to the encoder, while the translation was generated from the attention decoder. Differently, Zheng et al. in [95], used a preprocessing algorithm to remove similar and redundant frames of the input video and increase the processing speed of the neural network without losing information. Then, they employed an SLT architecture that consisted of a 2D-CNN, temporal convolutional layers and bidirectional GRUs. Their method was able to deal with long videos that have long-term dependencies, improving the translation quality. Zhou et al. in [62], proposed a multi-modal framework for CSLR and SLT tasks. The proposed method used 2D-CNN, 1D convolutional layers and several BLSTMs and learned both spatial and temporal dependencies between different modalities. The proposed method achieved a BLEU-4 score of 23.65 on the test set of Phoenix-2014-T. However, due to the multi-modal cues, this method is very computationally heavy and requires several hours of training.

CHAPTER 6

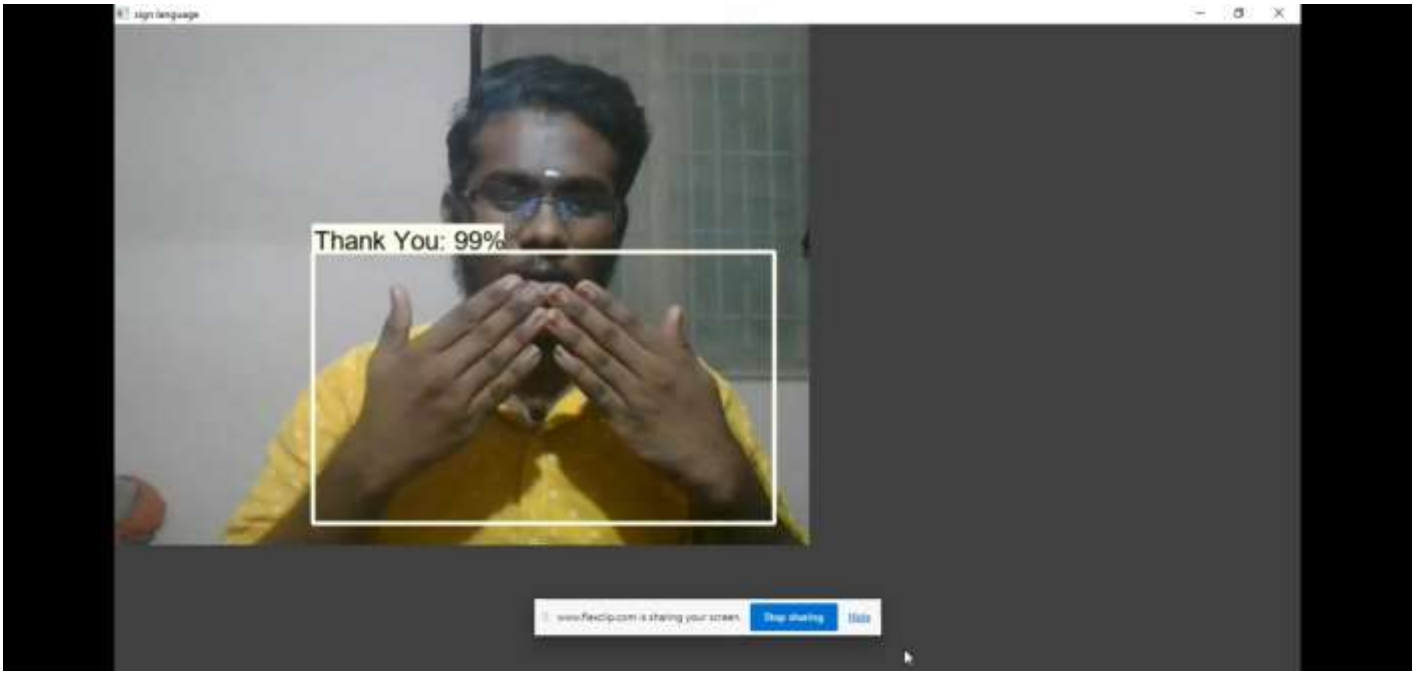
EXPERIMENT & RESULT

Results

Hello Sign (Fig 6.1**)**

Yes Sign (Fig 6.2)

No Sign (Fig 6.3)



Thank You Sign (Fig 6.4)

CHAPTER 7

SOURCE CODE

7.1 PROGRAM

7.1.1 Real Time Image Capture

```
import cv2
import os
import time
import uuid
IMAGES_PATH = 'Tensorflow/workspace/images/collectedimages'
labels = ['hello','thanks', 'yes', 'no', 'iloveyou','peace']
number_imgs = 15
for label in labels:

    !mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}
    cap = cv2.VideoCapture (0)
    print('Collecting images for {}'.format(label))
    time.sleep (10)
    for imgnum in range(number_imgs):
        ret, frame = cap.read()
    imgname=os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
          cv2.imwrite(imgname, frame)
          cv2.imshow('frame', frame)
          time.sleep(3)

          if cv2.waitKey(1) & 0xFF == ord('q'):
             break
    cap.release()
```

7.1.2 Sign Language Detection

 # 0. Setup Paths

```
WORKSPACE_PATH = 'Tensorflow/workspace'
SCRIPTS_PATH = 'Tensorflow/scripts'
APIMODEL_PATH = 'Tensorflow/models'
ANNOTATION_PATH = WORKSPACE_PATH+'/annotations'
IMAGE_PATH = WORKSPACE_PATH+'/images'
MODEL_PATH = WORKSPACE_PATH+'/models'
```

```
PRETRAINED_MODEL_PATH=WORKSPACE_PATH+'/pre-trained-models'
CONFIG_PATH = MODEL_PATH+'/my_ssd_mobnet/pipeline.config'
CHECKPOINT_PATH = MODEL_PATH+'/my_ssd_mobnet/'
```

 # 1. Create Label Map

```
labels = [
    {'name':'Hello', 'id':1},
    {'name':'Yes', 'id':2},
    {'name':'No', 'id':3},
    {'name':'Thank You', 'id':4},
    {'name':'I Love You', 'id':5},
    {'name':'Peace', 'id':6}

]

with open(ANNOTATION_PATH + '\label_map.pbtxt', 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

 # 2. Create TF records

```
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x {IMAGE_PATH + '/train'} -l
{ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH +
'/train.record'}

!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x{IMAGE_PATH + '/test'} -l
{ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH + '/test.record'}
```

 # 3. Update Config for Transfer Learning

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
CONFIG_PATH = MODEL_PATH+'/'+CUSTOM_MODEL_NAME+'/pipeline.config'
config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
config
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(CONFIG_PATH, "r") as f:
    proto_str = f.read()
```

```
    text_format.Merge(proto_str, pipeline_config)
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint =
PRETRAINED_MODEL_PATH+'/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-
8/checkpoint/ckpt-0'
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= ANNOTATION_PATH +
'/label_map.pbtxt'
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] =
[ANNOTATION_PATH + '/train.record']
pipeline_config.eval_input_reader[0].label_map_path = ANNOTATION_PATH +
'/label_map.pbtxt'
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] =
[ANNOTATION_PATH + '/test.record']
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(CONFIG_PATH, "wb") as f:
    f.write(config_text)
```

 # 4. Train the model

```
print("""python {}/research/object_detection/model_main_tf2.py --model_dir={}/{} --
pipeline_config_path={}/{}/pipeline.config --
num_train_steps=20000""".format(APIMODEL_PATH,
MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAM
E))
```

 # 5. Load Train Model From Checkpoint

```
import os
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH, 'ckpt-21')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
```

```
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

 # 6. Detect in Real-Time

```
import cv2
import numpy as np
category_index =
label_map_util.create_category_index_from_labelmap(ANNOTATION_PATH+'/label_ma
p.pbtxt')
# Setup capture
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
while True:
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.5,
            agnostic_mode=False)
```

```
    cv2.imshow('sign language detection using AI',  cv2.resize(image_np_with_detections,
(800, 600)))

    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        break
detections = detect_fn(input_tensor)
from matplotlib import pyplot as plt
```

CHAPTER 8

CONCLUSION & FUTURE WORK

Our project aims to make communication simpler between deaf and dumb people by introducing Computer in communication path so that sign language can be automatically captured, recognized, translated to text and displayed it on LCD. There are various methods for sign language conversion.

In future work, proposed system can be developed and implemented using Python. Image Processing part should be improved so that System would be able to communicate in both directions i.e.it should be capable of converting normal language to sign language and vice versa. We will try to recognize signs which include motion. Moreover we will focus on converting the sequence of gestures into text i.e. word and sentences and then converting it into the speech which can be heard.

REFERENCE

1.A. Das, M. Wasif Ansari and R. Basak, (2020) " Detection Using TensorFlow and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, pp. 1-5.

2.Bragg, D., Verhoef, T., Vogler, C., Ringel Morris, M.,Koller, O., Bellard, M., Berke, L., Boudreault, P.,Braffort, A., Caselli, N., Huenerfauth, M., and Kacorri, H. (2019). Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective. In The 21st International ACMSIGACCESS Conference on Computers and Accessibility – ASSETS '19, pages 16–31, Pittsburgh, PA, USA. ACM Press.

3.Al-khazraji, S., Berke, L., Kafle, S., Yeung, P., and Huenerfauth, M. (2018). Modeling the Speed and Timing of American Sign Language to Generate Realistic Animations. In Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '18, pages 259–270, Galway, Ireland. ACM Press.

4.Camgoz, N. C., Hadfield, S., Koller, O., Ney, H., and Bowden, R. (2018). Neural Sign Language Translation.

5.Camgoz, N. C., Hadfield, S., Koller, O., and Bowden, R.(2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. In 2017 IEEE International Conference on Computer Vision (ICCV),pages 3075–3084, Venice. IEEE.

6.Joseph Redmon, S. D. (2016) "You Only Look Once), Real Time Object Detection" IEEE

7.Adamo-Villani, N. and Wilbur, R. B. (2015). ASL-Pro:American Sign Language Animation with Prosodic Elements. In Antona, M. and Stephanidis, C., editors, Universal Access in Human-Computer Interaction. Access to Interaction, volume 9176, pages 307–318. Springer International Publishing, Cham.

8.Bavelier, D., Newport, E. L., and Supalla, T. (2003). Children Need Natural Languages, Signed or Spoken. Technical report, Dana Foundation.

.