

**Department of Electronics & Communication Engineering,  
MANIT Bhopal**



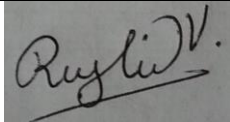
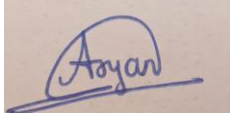
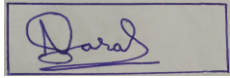

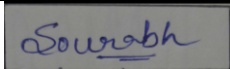
**Minor Project Report (EC-326)**

**Batch 2018 - 22**

**Group No.: 17**

**Project Title: -**

Multilingual audio to sign language and sign  
language to multilingual audio conversion  
using CNN and ANN  
(Gesture Speech)

Serial No.	Scholar No.	Name	Signature
1	181114081	Rushil Verma	
2	181114026	Aryan Gurjar Banke	
3	181114045	Niket Kumar Saraf	
4	181114092	Dayasagar Sahu	
5	181114008	Sourabh Singh	

**Title of the proposed project:**

Multilingual audio to sign language and sign language to multilingual audio conversion using CNN and ANN

**Name and signature of the Faculty Mentor:**

Faculty Mentor: Dr. Manish Kashyap

Signature-

## **Index**

Serial no.	Topic	Page No.
1	Introduction	4
2	Objective of the project	7
3	Research	8
4	Software Components	19
5	Results & conclusion	24
6	Future scope & advancements	24
7	References	25
8	Program / coding of your project	26

## **Introduction**

Special abled people (Deaf) have used sign languages throughout history. One of the earliest written records of a sign language is from the fifth century BC, in Plato's *Cratylus*, where Socrates says: "If we hadn't a voice or a tongue, and wanted to express things to one another, wouldn't we try to make signs by moving our hands, head, and the rest of our body, just as dumb people do at present?"

Sign languages do not have any linguistic relation to the spoken languages of the lands in which they arise. The correlation between sign and spoken languages is complex and varies depending on the country more than the spoken language. For example, Australia, Canada, New Zealand, the UK, and the US all have English as their dominant language, but American Sign Language (ASL), used in the US and English-speaking Canada, is derived from French Sign Language while the other three countries use varieties of British, Australian and New Zealand Sign Language, which is unrelated to ASL.

Indo-Pakistani Sign Language (IPSL) is the predominant sign language in South Asia, used by at least several hundred thousand deaf signers. The Deaf communities of the Indian subcontinent are still struggling for IPSL to gain the status of sign language as a minority language. Though sign language is used by many deaf people in the Indian subcontinent, it is not used officially in schools for teaching purposes. In 2005, the National Curricular Framework (NCF) gave some degree of legitimacy to sign language education, by hinting that sign languages may qualify as an optional third language choice for hearing impaired students. NCERT in March 2006 published a chapter on sign language in a class III textbook, emphasizing the fact that it is a language like any other and is "yet another mode of communication." The aim was to create healthy attitudes towards the differently abled.

Although discussion of sign languages and the lives of deaf people is extremely rare in the history of South Asian literature, there are a few references to deaf people and gestural communication in texts dating from antiquity. Symbolic hand gestures known as mudras have been employed in religious contexts in Hinduism, Buddhism and Zoroastrianism for many centuries, although these religious traditions have often excluded deaf people from participation in a ritual or religious membership. In addition, classical Indian dance and theatre often employs stylized hand gestures with particular meanings

Sign language is a visual language and consists of 3 major components:

- Fingerspelling: used to spell words letter by letter
- Word level sign vocabulary: Used for the majority of communication
- Non-manual features: Facial expression and tongue, mouth, and body position

This project is primarily based on First part of the sign language Fingerspelling.

This poster is taken from

Indian Sign Language Research and Training Centre (ISLRTC)

Department of Empowerment of Persons with Disabilities,

Divyanga Ministry of Social Justice & Empowerment,

Government of India

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P
Q	R	S	T
U	V	W	X
	Y	Z	



### Indian Sign Language Research and Training Centre (ISLRTC)

Department of Empowerment of Persons with Disabilities, (Divyangjan)

Ministry of Social Justice & Empowerment (Govt. of India)

A-91, 1<sup>st</sup> floor, Nagpal Business Tower, Okhla Phase II, New Delhi - 110020  
Tel: 011-26387558/59, Email: islrtnewdelhi@gmail.com, Website: islrtnic.in

ISLRTC is an autonomous organization of the Government of India established under Societies Registration Act 1860, Registration Number S/1440/2016 under the aegis of Department of Empowerment of Persons with Disabilities (Divyangjan) under Ministry of Social Justice and Empowerment, Government of India, New Delhi.

#### Our Aims & Objectives:

- ☞ To develop manpower for teaching & using ISL.
- ☞ To encourage the use of ISL in education.
- ☞ To promote research in ISL linguistics.
- ☞ To create ISL corpus and database.
- ☞ To give training to different personnels.
- ☞ To collaborate with deaf organizations.
- ☞ To upgrade the ISL.
- ☞ To conduct various courses in the field of ISL.
- ☞ To connect ISL interpreters.
- ☞ To conduct training programme, workshop, conference etc.
- ☞ To support the ISL interpreting services.

#### Resources



#### RCI Approved Courses

- ☞ Diploma in Indian Sign Language Interpretation (DISLI)
- ☞ Proposed - Diploma in Teaching Indian Sign Language (DTISL)

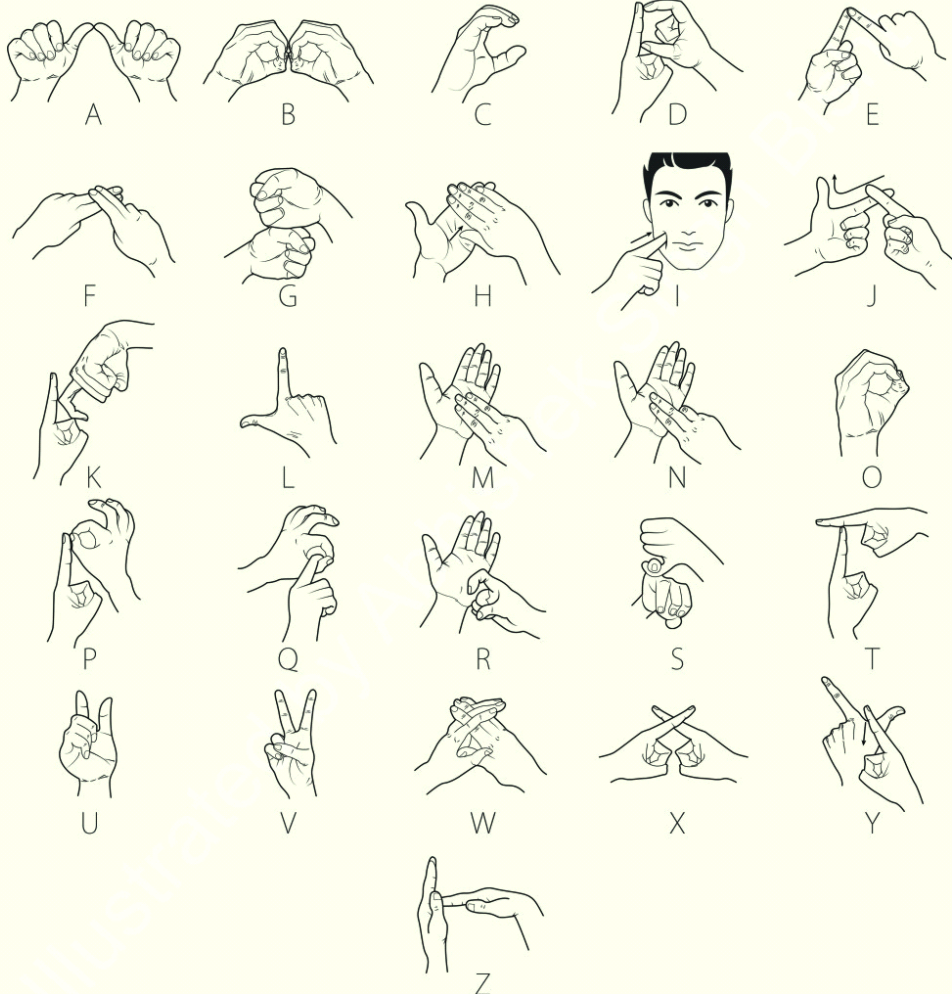


## Indian Sign Language Research and Training Centre (ISLRTC)

Department of Empowerment of Persons with Disabilities, (Divyangjan)

Ministry of Social Justice & Empowerment (Govt. of India)

### English Alphabets in Indian Sign Language



Developed by  
ISLRTC

A-91, 1st floor, Nagpal Business Tower, Okhla Phase II, New Delhi - 110020  
Tel: 011-26387558/59, Email: islrtnewdelhi@gmail.com, Website: islrtnic.in

Courtesy: <http://www.islrtnic.in/poster-manual-alphabet-isl>

## **Objective of the project**

To develop an application which could be used by especially abled person to be able to convey their hand sign or gesture language into speech and aid an ordinary person to translate speech to gesture or hand sign language in order to make the communication more fluent between a person with hearing impairment or speaking disability but has knowledge about Indian sign language. Most of the people are not able to communicate due to either the lack of knowledge of sign language to other peoples. With this as a translator an ordinary person can also be able to transform their speech into sign language which may be understood by person who has hearing impairment. Most of the programs present usually have only American sign language as their region of processing, but this application is specially designed for Indian Sign Language (ISL) Translation.

## **Research**

### **Gesture to Speech:**

The steps to proceed with are these:

- Data Extraction
- Data Pre-processing
- Feature Recognition/extraction using Neural network model
- Test the ML Model
- Speech generator

#### **1. Data Extraction:**

There are broadly two ways:

- Using electronic sensors:

The use of flex sensor to measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. This may be used to analyse hand bending movements but it will not be effective as every person will not be able to use it properly and gives an extra cost for equipment.

- Using Computer Vision

Computer vision tasks include methods for acquiring, processing, analysing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Moreover, this method is more effective as it does not require overhead cost for equipment, but image processing machine like mobile, laptop, computer, etc. The complication here will be due to large variation on surrounding and noise removal from region of Interest

In order to make a dataset we used computer vision in form of camera sight as primary data collection tool

Different types present were:

- Infrared camera
- Ultraviolet light sensor
- Visible light camera

And more but to make it affordable and easy to use we stick to visible range light capturing camera

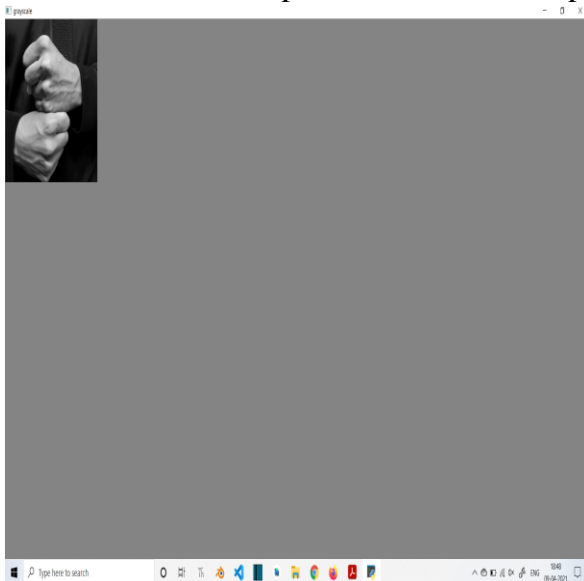


## 2. Data pre-processing:

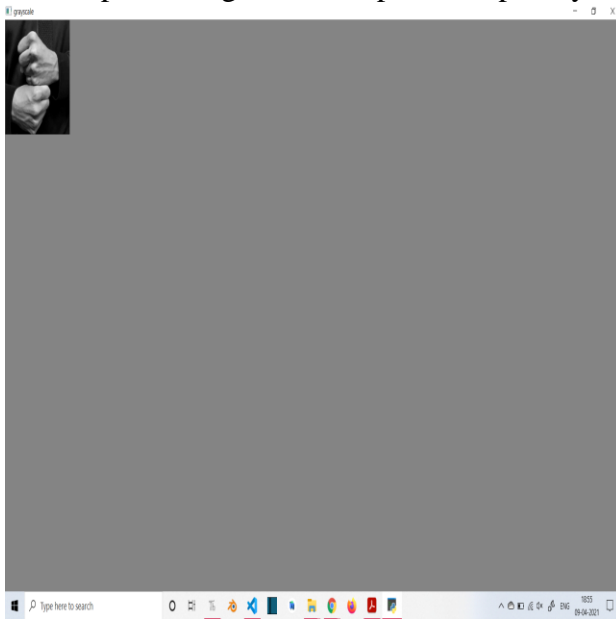
- Extract the region of Interest as whole of camera view captures most of the noise surrounding which are redundant thus cropping image to a square shaped window.



- We first of all converted the 3-channel image (RGB) into grayscale in order to decrease the operation time and complexity of processing.



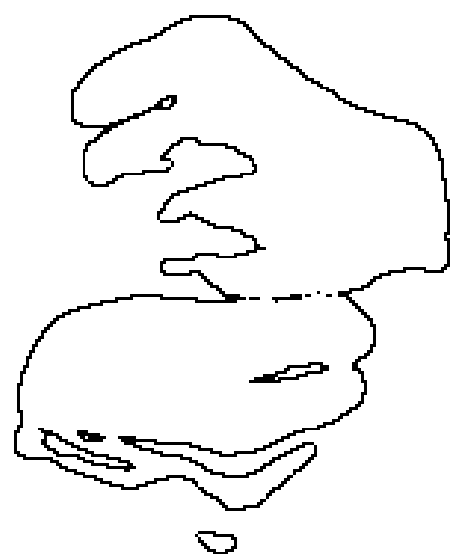
- We resized the image into 200x200 pixel image so as to further more reduce the processing time and space complexity of the algorithm.



- We introduced a gaussian filter in order to reduce the noise in image.



- Now to extract the edge map we apply adaptive threshold filter of gaussian nature



### **Gesture classification:**

- Hidden Markov Models (HMM) is used for the classification of the gestures. This model deals with dynamic aspects of gestures. Gestures are extracted from a sequence of video images by tracking the skin-colour blobs corresponding to the hand into a body–face space centred on the face of the user. The goal is to recognize two classes of gestures: deictic and symbolic. The image is filtered using a fast look–up indexing table. After filtering, skin colour pixels are gathered into blobs. Blobs are statistical objects based on the location (x,y) and the colourimetry (Y, U, V) of the skin colour pixels in order to determine homogeneous areas.
- Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation. Thus, unlike many other recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video, with a static background. The first step is to segment and label the objects of interest and to extract geometric invariants from them. Next step is the classification of gestures by using a K nearest neighbour algorithm aided with distance weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes“ classifier.
- According to paper on “Human Hand Gesture Recognition Using a Convolution Neural Network” by Hsien-I Lin , Ming-Hsiang Hsu, and Wei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct a skin model to extract the hand out of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to centre the image about it. They input this image to a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using their model they produce an accuracy of around 95% for those 7 gestures.

### 3. Feature Recognition/extraction using Neural network model:

#### Algorithm Layer:

- Apply gaussian blur filter and threshold to the frame taken with OpenCV to get the processed image after feature extraction.
- This processed image is passed to the CNN model for prediction and if a letter is detected for more than 25 frames then the letter is printed and taken into consideration for forming the word.
- Space between the words is considered using the blank symbol which is an empty frame with no sign.

#### CNN Model:

- **1st Convolution Layer:** The input picture has resolution of 200x200 pixels. It is first processed in the first convolutional layer using 64 filter weights.
- **1st Pooling Layer:** The pictures are down sampled using max pooling of 3x3 i.e., we keep the highest value in the 3x3 square of array. Therefore, our picture is down sampled.
- **2nd Convolution Layer:** Now, this output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 128 filter weights (2x2 pixels each).
- **2nd Pooling Layer:** The resulting images are down sampled again using max pool of 3x3 and is reduced to even lesser resolution of image.
- **3rd Convolution Layer:** convolutional layer using 256 filter weights (2x2 pixels each).
- **3rd Pooling Layer:** The resulting images are down sampled again using max pool of 3x3 and is reduced to even lesser resolution of image.
- **Flatten Layer:** It is used to convert the 2D pixel array into linear form in order to produce converge it into 27 class of hand signs.
- **Final layer:** The output of the 3rd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).
- **Activation Function:** We have used ReLu (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons). ReLu calculates  $\max(x, 0)$  for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time. At the last activation function, we used SOFTMAX function. It is used as the activation function in the

output layer of neural network models that predict a multinomial probability distribution. That is, SoftMax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels.

- **Pooling Layer:** We apply Max pooling to the input image with a pool size of (3, 3) with ReLu activation function. This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.
- **Dropout Layers:** The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn't perform well when given new examples. This layer "drops out" a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out.
- **Optimizer:** We have used Adam optimizer for updating the model in response to the output of the loss function. Adam combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp).

#### 4. Training and Testing:

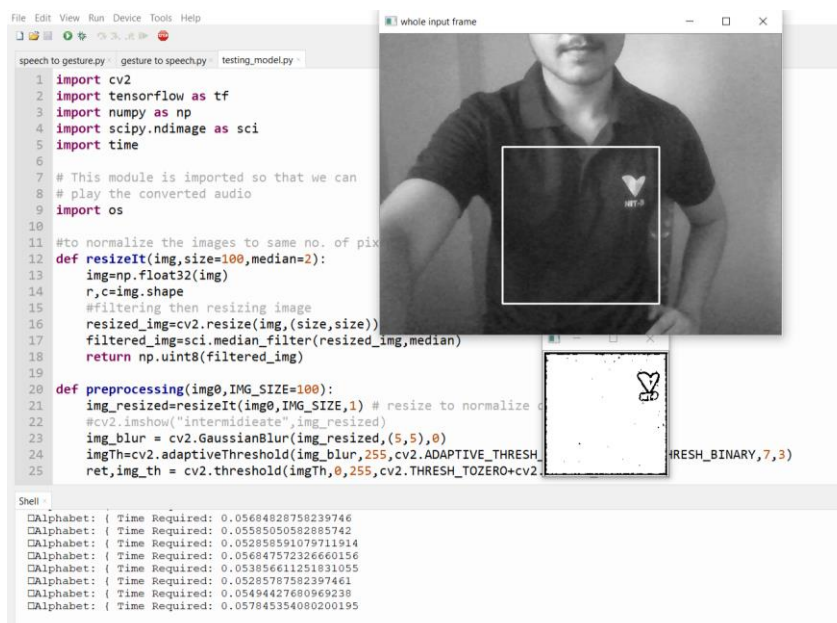
We convert our input images (RGB) into grayscale and apply gaussian blur to remove unnecessary noise. We apply adaptive threshold to extract our hand from the background and resize our images to 200x200.

We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above.

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

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

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



```
File Edit View Run Device Tools Help
speech to gesture.py gesture to speech.py testing_model.py
1 import cv2
2 import tensorflow as tf
3 import numpy as np
4 import scipy.ndimage as sci
5 import time
6
7 # This module is imported so that we can
8 # play the converted audio
9 import os
10
11 #to normalize the images to same no. of pixels
12 def resizeIt(img,size=100,median=2):
13     img=np.float32(img)
14     r,c=img.shape
15     #filtering then resizing image
16     resized_img=cv2.resize(img,(size,size))
17     filtered_img=sci.median_filter(resized_img,median)
18     return np.uint8(filtered_img)
19
20 def preprocessing(img0,IMG_SIZE=100):
21     img_resized=resizeIt(img0,IMG_SIZE,1) # resize to normalize
22     #cv2.imshow("intermediate",img_resized)
23     img_blur = cv2.GaussianBlur(img_resized,(5,5),0)
24     imgTh=cv2.adaptiveThreshold(img_blur,255,cv2.ADAPTIVE_THRESH
25     ret,img_th = cv2.threshold(imgTh,0,255,cv2.THRESH_TOZERO+cv2.THRESH_BINARY,7,3)
26
27 Shell
28
29 [Alphabet: ( Time Required: 0.05684828758239746
30 [Alphabet: ( Time Required: 0.05585050582885742
31 [Alphabet: ( Time Required: 0.052858591079711914
32 [Alphabet: ( Time Required: 0.056847572326660156
33 [Alphabet: ( Time Required: 0.053856611251831055
34 [Alphabet: ( Time Required: 0.05285787582397461
35 [Alphabet: ( Time Required: 0.05494427680969238
36 [Alphabet: ( Time Required: 0.057845354080200195
```

A blank input screen is displayed showing '{' as output to represent blank space and time required to process the frame





## Speech to gesture:

### Steps:

- **Speech acquisition:** It is done through audio input device(microphone).
- **Speech Recognition:** The speech is saved in local device then converted into text or character format using speech recognition API, Python mainly supports following Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text
- **Display:**
  - a. The good quality images of ISL alphabets corresponding to every speech alphabet is stored on local device (which could be later converted to an online source).
  - b. This character format is then fed to algorithm which displays corresponding graphic representation (image) of the ISL alphabet.
  - c. OpenCV library is used to display the image file over the display in order to be perceived by eyes.

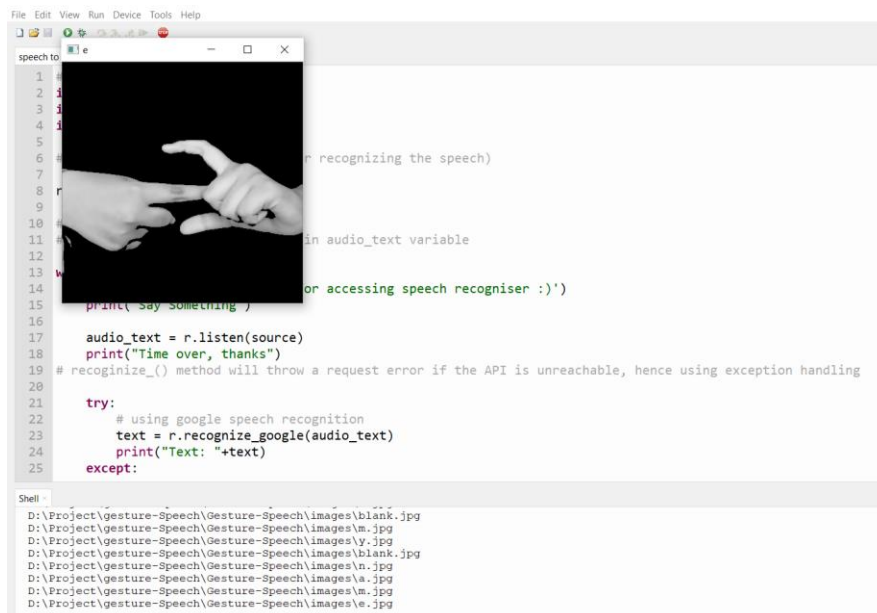
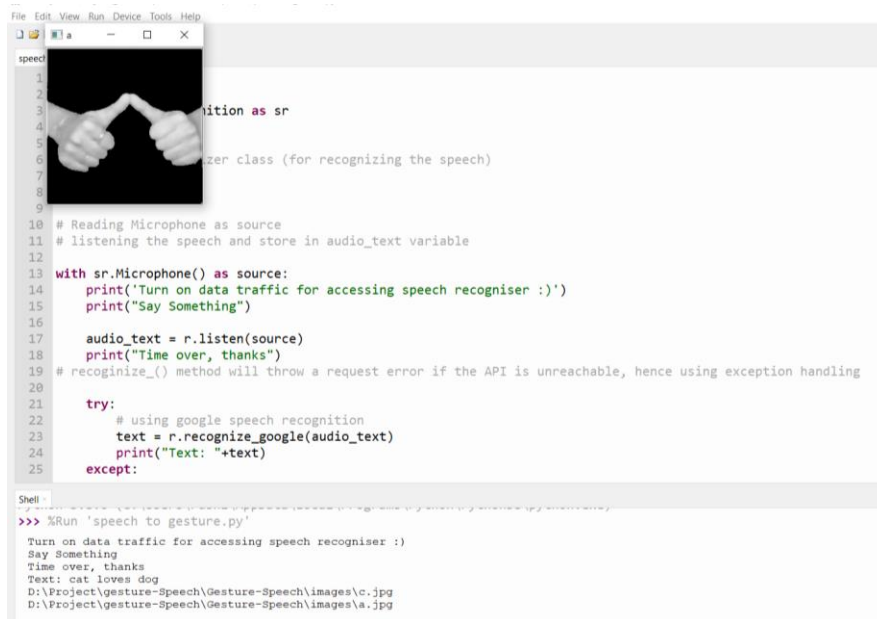


Figure displaying a frame of the speech converted text of alphabet 'E' of the detected word



```
1 # Import the SpeechRecognition module as sr
2
3 # Import the recognizer class (for recognizing the speech)
4
5 # Reading Microphone as source
6 # listening the speech and store in audio_text variable
7
8 # with sr.Microphone() as source:
9 #     print('Turn on data traffic for accessing speech recogniser :')
10 #     print("Say Something")
11
12 # audio_text = r.listen(source)
13 # print("Time over, thanks")
14 # recognize_() method will throw a request error if the API is unreachable, hence using exception handling
15
16 try:
17     # using google speech recognition
18     text = r.recognize_google(audio_text)
19     print("Text: "+text)
20 except:
```

Shell

```
>>> %Run 'speech to gesture.py'
Turn on data traffic for accessing speech recogniser :)
Say Something
Time over, thanks
Text: cat loves dog
D:\Project\gesture-Speech\gesture-Speech\images\c.jpg
D:\Project\gesture-Speech\gesture-Speech\images\a.jpg
```

Figure showing console detecting the speech and a frame is shown representing 'A' alphabet in ISL language alphabet

## Software Components

### Artificial Neural Networks:

Artificial Neural Network is a connection of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer.

**There are capable of learning and they have to be trained. There are different learning strategies:**

- Unsupervised Learning
- Supervised Learning
- Reinforcement Learning

### Convolution Neural Network:

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.

- **Convolution Layer:** In convolution layer we take a small window size [typically of length  $5 \times 5$ ] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color.
- **Pooling Layer:** We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two type of pooling:

**1. Max Pooling:** In max pooling we take a window size [for example window of size  $2 \times 2$ ], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its original Size.

**2. Average Pooling:** In average pooling we take average of all values in a window.

- **Fully Connected Layer:** In convolution layer neurons are connected only to a local region, while in a fully connected region, we connect all the inputs to neurons.
- **Final Output Layer:** After getting values from fully connected layer, we connect them to final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

## **TensorFlow:**

TensorFlow is an open-source software library for numerical computation. First, we define the nodes of the computation graph, then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

## **Keras:**

Keras is a high-level neural networks library written in python that works as a wrapper to TensorFlow. It is used in cases where we want to quickly build and test the neural network with minimal lines of code. It contains implementations of commonly used neural network elements like layers, objective, activation functions, optimizers, and tools to make working with images and text data easier.

## **OpenCV:**

OpenCV (Open-Source Computer Vision) is an open-source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAVE.

Library for performing speech recognition, with support for several engines and APIs, online and offline.

Speech recognition engine/API support:

- [CMU Sphinx](#) (works offline)
- Google Speech Recognition
- [Google Cloud Speech API](#)
- [Wit.ai](#)
- [Microsoft Bing Voice Recognition](#)
- [Houndify API](#)
- [IBM Speech to Text](#)
- [Snowboy Hotword Detection](#) (works offline)

Meta

License: BSD License (BSD)

Author: [Anthony Zhang \(Uberi\)](#)

Tagsspeech, recognition, voice, sphinx, google, wit, bing, api, houndify, ibm, snowboy

Maintainers

[Anthony.Zhang](#)

### **Finger spelling sentence formation Implementation:**

1. Whenever the count of a letter detected exceeds a specific value and no other letter is close to it by a threshold we print the letter and add it to the current string (In our code we kept the value as 50 and difference threshold as 20).
2. Otherwise we clear the current dictionary which has the count of detections of present symbol to avoid the probability of a wrong letter getting predicted.
3. Whenever the count of a blank (plain background) detected exceeds a specific value and if the current buffer are empty no spaces are detected.
4. In other case it predicts the end of word by printing a space and the current gets appended to the sentence below.

### **Autocorrect Feature:**

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

### **Training and Testing:**

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

We feed the input images after pre-processing to our model for training and testing after applying all the operations mentioned above.

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

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

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

### **Challenges Faced:**

There were many challenges faced by us during the project.

The first issue we faced was of dataset. We wanted to deal with raw images and that too square images as CNN in Keras as it was a lot more convenient working with only square images. We couldn't find any existing dataset for that hence we decided to make our own dataset.

Second issue was to select a filter which we could apply on our images so that proper features of the images could be obtained and hence then we could provide that image as input for CNN model. We tried various filter including binary threshold, canny edge detection, gaussian blur etc. but finally we settled with gaussian blur filter with thresholding over it.

Third issue was, we tried to extract the features directly from the images, but the problem arise was that it is getting adapted to the particular dataset only and doesn't give output for any other dataset. So, to fix this we used Adaptive Gaussian Thresholding to extract the edge map which finally fixed the issue and now it works can work on different dataset.

More issues were faced relating to the accuracy of the model we trained in earlier phases which we eventually improved by increasing the input image size and also by improving the dataset.

## **Results:**

We have achieved an accuracy of 95.8% in our model using only layer 1 of our algorithm, and using the combination of layer 1 and layer 2 we achieve an accuracy of 98.0%. Most of the research papers focus on using devices like Kinect for hand detection. In they build a recognition system for Flemish sign language using convolutional neural networks and Kinect and achieve an error rate of 2.5%. In a recognition model is built using hidden Markov model classifier and a vocabulary of 30 words and they achieve an error rate of 10.90%. In they achieve an average accuracy of 86% for 41 static gestures in Japanese sign language. Using depth sensors map achieved an accuracy of 99.99% for observed signers and 83.58% and 85.49% for new signers. They also used CNN for their recognition system. One thing should be noted that our model doesn't uses any background subtraction algorithm while some of the models present above do that. So, once we try to implement background subtraction in our project the accuracies may vary. On the other hand, most of the above.

## **Conclusion:**

In this report, a functional real time vision based Indian sign language recognition for D&M people have been developed for ISL alphabets. We achieved final accuracy of 98.0% on our dataset. This application could be used by especially abled person to be able to convey their hand sign or gesture language into speech and aid an ordinary person to translate speech to gesture or hand sign language in order to make the communication more fluent between a person with hearing impairment or speaking disability but has knowledge about Indian sign language. We are able to improve our prediction after implementing two layers of algorithms in which we verify and predict symbols which are more similar to each other. This way we are able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate.

## **Future Scope :**

- The UI of the project is more to be worked on.
- We are planning to achieve higher accuracy even in case of complex backgrounds by trying out various background subtraction algorithms.
- We are also thinking of improving the pre-processing to predict gestures in bright light conditions and extremely low light conditions with a higher accuracy without using Black Background.
- We can new model architectures of CNN like GoogleNet, etc which require less parameters to train(Reducing size of model file) but have high accurate outputs.



**References:**

- [https://www.youtube.com/watch?v=Wo5dMEP\\_BbI&list=PLQVvva0QuDcjD5BAw2DxE6OF2tius3V3&ab\\_channel=sentdex](https://www.youtube.com/watch?v=Wo5dMEP_BbI&list=PLQVvva0QuDcjD5BAw2DxE6OF2tius3V3&ab_channel=sentdex)
- [https://www.youtube.com/watch?v=WrfjUvmWzko&ab\\_channel=DigitalArtsAcademyfortheDeafLLP](https://www.youtube.com/watch?v=WrfjUvmWzko&ab_channel=DigitalArtsAcademyfortheDeafLLP)
- <https://medium.com/@gongster/how-does-a-neural-network-work-intuitively-in-code-f51f7b2c1e3f>
- <https://medium.com/@gongster/building-a-simple-artificial-neural-network-with-keras-in-2019-9eccb92527b1>
- <https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4>
- <https://contactsunny.medium.com/label-encoder-vs-one-hot-encoder-in-machine-learning-3fc273365621>

## **Python Code:**

**For gesture to speech:**

```
import cv2,
import tensorflow as tf
import numpy as np
import scipy.ndimage as sci
from gtts import gTTS
import time
from textblob import TextBlob

# This module is imported so that we can
# play the converted audio
import os

#to normalize the images to same no. of pixels
def resizeIt(img,size=100,median=2):
    img=np.float32(img)
    r,c=img.shape
    #filtering then resizing image
    resized_img=cv2.resize(img,(size,size))
    filtered_img=sci.median_filter(resized_img,median)
    return np.uint8(filtered_img)

def preprocessing(img0,IMG_SIZE=100):
    img_resized=resizeIt(img0,IMG_SIZE,1) # resize to normalize data size
    #cv2.imshow("intermidieate",img_resized)
    img_blur = cv2.GaussianBlur(img_resized,(5,5),0)
    imgTh=cv2.adaptiveThreshold(img_blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv
2.THRESH_BINARY,7,3)
```

```
ret,img_th = cv2.threshold(imgTh,0,255,cv2.THRESH_TOZERO+cv2.THRESH_OTSU)
```

```
#edges = cv2.Canny(img_resized,170, 300)
```

```
return img_th
```

```
def playText(text):
```

```
myobj = gTTS(text=text, lang='en' , slow=False)
```

```
if os.path.exists("audio.mp3"):
```

```
os.remove("audio.mp3")
```

```
myobj.save("audio.mp3")
```

```
# Playing the converted file
```

```
#os.system("mpg123 welcome.mp3")
```

```
from playsound import playsound
```

```
playsound('audio.mp3')
```

```
return 0
```

```
ALPHABET = [] #array containing letters to categorize
```

```
alpha = 'a'
```

```
for i in range(0, 27):
```

```
ALPHABET.append(alpha)
```

```
alpha = chr(ord(alpha) + 1)
```

```
prev=""
```

```
model = tf.keras.models.load_model("model_name.model")
```

```
cap = cv2.VideoCapture(0) #to load video file
```

```
count = 0
```

```
buffer = []
```

```
prev_time = time.time()
```

```
word = ''
```

```
sentence = ''
```

```

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    if ret==False:
        break
    # Our operations on the frame come here
    img_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    IMG_SIZE = 200
    #print(img_gray.shape)
    w, h = img_gray.shape
    top=180
    left=h-195
    down=w-50
    right=195
    img_rect = cv2.rectangle(img_gray,(right,top),(left,down),(255,0,0),2)
    img_gray = img_gray[top:down, right:left]
    #print(img_gray.shape)
    img_test = preprocessing(img_gray,IMG_SIZE)

    cv2.imshow('whole input frame', np.uint8(img_rect))
    cv2.imshow('qwerty',img_test)
    # Display the resulting frame
    #cv2.imshow('testing this',np.uint8(img_test))

    prediction = model.predict([img_test.reshape(-1, IMG_SIZE, IMG_SIZE, 1)])
    #print(img_test)

    text = ALPHABET[int(np.argmax(prediction[0]))]
    _ = os.system('cls')

    print('Alphabet: '+text+' Word: '+word+' Sentence: '+sentence+' Time Required: '+str(time.time()-prev_time))

```

```

prev_time = time.time()
no_frames = 50

count = count + 1
buffer.append(text)
print(buffer)
if (count > no_frames) :
text = max(set(buffer),key = text.count) #finding mode of buffer of letters
try:
playText(text)
except:
print("Try again!!")

count=0
buffer = []
if( text == '{'):
word = word + ' '

else:
word = word + text

playText(word)

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

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()

```

### **For gesture to speech :**

```
#import library,
import cv2
import speech_recognition as sr
import os

# Initialize recognizer class (for recognizing the speech)

r = sr.Recognizer()

# Reading Microphone as source
# listening the speech and store in audio_text variable

with sr.Microphone() as source:
    print("Turn on data traffic for accessing speech recogniser :)")
    print("Say Something")

    audio_text = r.listen(source)

    print("Time over, thanks")

    # recognize_() method will throw a request error if the API is unreachable, hence using
    exception handling

    try:
        # using google speech recognition
        text = r.recognize_google(audio_text)
        print("Text: "+text)
    except:
        print("Sorry, I did not get that, please try again")

    for elem in text:
        path = r'D:\Project\gesture-Speech\Gesture-Speech\images'
```

```
if elem == ' ':
    full_path = os.path.join(path,'blank')+'.jpg'
    print(full_path)
    img = cv2.imread(full_path ,cv2.IMREAD_GRAYSCALE)
    cv2.imshow('blank',img)
    cv2.waitKey(1000)
    cv2.destroyWindow('blank')
else:
    full_path = os.path.join(path,elem)+'.jpg'
    print(full_path)
    img = cv2.imread(full_path ,cv2.IMREAD_GRAYSCALE)
    cv2.imshow(elem,img)
    cv2.waitKey(1000)
    cv2.destroyWindow(elem)

cv2.destroyAllWindows()
```

**Full data extraction codes in python is given in Following link:**

<https://github.com/Tech-Cravers/Gesture-Speech>