A PROJECT REPORT

Sign Language Recognition for Deaf and Mute People

Submitted by

1. Mitesh Makwana (16CP016)
2. Manav Prajapati (16CP032)
3. Sahil Hada (16CP045)

BACHELOR OF TECHNOLOGY

In

Computer Engineering

Birla Vishvakarma Mahavidhyalaya

An Autonomous Institution

Vallabh Vidyanagar, 388120

ANAND, GUJARAT

Faculty Guides:

Dr. Narendra M. Patel

Prof. Mosin I. Hasan

Gujarat Technological University, Ahmedabad

CERTIFICATE

This is to certify that Project work embodied in this report titled **"Sign Language Recognition for Deaf-Mute people**" was carried out by (**ID NO: 16CP016 – Mitesh Makwana, ID NO: 16CP032 - Manav Prajapati And ID NO: 16CP032 - Sahil Hada**) at **Birla Vishvakarma Mahavidhyalaya An Autonomous Institution** and **Code 007** for partial fulfillment of Bachelor of Technology degree in Computer Engineering to be awarded by Gujarat Technological University. This work has been carried out under my guidance and supervision and it is up to my satisfaction.

Date:

Place:

Dr. Narendra M. Patel	**Prof. Mosin Hasan**	**Dr. Darshak Thakore**
Project Guide	**Project Guide**	**Head of Department**

COMPLIANCE CERTIFICATE

This is to Declare that project work embodied in this Project titled "**Sign-Language Recognition for Deaf-Mute people**" was carried by (**ID NO:16CP016 –Mitesh Makwana, ID NO: 16CP032- Manav Prajapati and ID NO: 16CP032- Sahil Hada**) at **Birla Vishvakarma Mahavidyalaya an Autonomous Institution** and **Code 007** for partial fulfillment of Bachelor of Technology degree to be awarded by Gujarat Technological University. We declare that, the current report and work does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques quotations or any other materials from the work of other people included in our report, published. Failure to any infringement and copyright rules, we own all the responsibilities and university has right to withdraw degree at any point of time.

Date:

Place:

Mitesh Makwana **Manav Prajapati** **Sahil Hada**

16CP016 **16CP032** **16CP045**

REPORT APPROVAL CERTIFICATE

This is to certify that research work embodied in this dissertation titled "**Sign Language Recognition for Deaf-Mute people**" was carried out by (**ID NO:16CP016 –Mitesh Makwana, ID NO: 16CP032- Manav Prajapati and ID NO: 16CP032- Sahil Hada**) at **Birla Vishvakarma Mahavidyalaya & Code 007** is approved for the degree of Bachelor of Technology in Computer Engineering by Gujarat Technological University.

Date:

Place:

Examiner Sign

DECLARATION OF ORIGINALITY

We hereby Declare that we are the sole authors of this report and that neither any part of this report nor the whole of the report has been submitted for a degree to any other University or Institution.

We Declare that, to the best of our knowledge, the current report does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations or any other material from the work of other people included in our report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copyrighted material that surpasses the boundary of fair dealing within the meaning of the Indian Copyright (Amendment) Act 2012, we certify that we have obtained a written permission from the copyright owner(s) to include such material(s) in the current report and have included copies of such copyright clearances to our appendix.

We declare that this is a true copy of report, including any final revisions, as approved by report review committee.

We have checked write up of the present report using anti-plagiarism database and it is in allowable limit. Even though later on in case of any complaint pertaining of plagiarism, we are sole responsible for the same and we understand that as per UGC norms, University can even revoke Master of Engineering degree conferred to the student submitting this report.

Date:

Mitesh Makwana **Manav Prajapati** **Sahil Hada**

16CP016 **16CP032** **16CP045**

Acknowledgment

I hereby would like to take this opportunity to thank all persons who has involved generously in helping me and assisting me.

We would like to convey our deepest gratitude towards our supervisor, Dr. Narendra M. Patel and Professor Mosin Hasan for his support and supervision, and for the valuable knowledge that he shared with us. We would thankful to Dr. Narendra M. Patel who give us Dataset for our project.

We would like to thank my friends and juniors who have helped me to complete the thesis work successfully. We would like to convey appreciation to our family members, for their encouragement and support.

Abstract

Every human being listen, sees and react to surrounding. But there are some unlucky out there whom cannot so this. This deaf and mute people use sign language to communicate with each other. The sign language is known by this community but not by the normal lay man. So it is hard for this people to communicate with the lay man who do not know the sign language. It is also a challenge for normal person to communicate with this community because they do not know this language. In this project, the objective is make the system for deaf and mute people so that whenever they make any sign it gives the meaning of the sign in normal language in real time. This helps the deaf and mute to communicate with normal people. For methodology to solve this problem there are many ways and this CNN approach is most reliable in real time. The image from camera is captured and predicted in real time. The original color image is converted into HSV image and given to the CNN model as an input. We have train our model for 36 classes including alphabets and numbers. This trained model will predict the character accordingly. The character is predicted almost every second in real time. We combine 25 prediction results to predict actual result so it gives more accurate answer. There are some signs which very similar so they are complex to predict. We got around 70% accuracy in our system.

List of Figures

List of Tables

PLAGIARISM SCAN REPORT

Words	945	Date	October 21,2019
Characters	5915	Exclude Url	

5% Plagiarism	95% Unique	2 Plagiarized Sentences	39 Unique Sentences

PLAGIARISM SCAN REPORT

Words	954	Date	October 21,2019
Characters	5816	Exclude Url	

14% Plagiarism	86% Unique	6 Plagiarized Sentences	37 Unique Sentences

PLAGIARISM SCAN REPORT

Words	920	Date	October 21,2019
Characters	5730	Exclude Url	

42% Plagiarism	58% Unique	19 Plagiarized Sentences	26 Unique Sentences

List of Symbols, Abbreviations and Nomenclature

CNN: Convolution neural network

CV: Computer vision

SRS: Software requirement specification

ASL: American Sign Language

ISL: Indian sign language

ML: Machine learning

RMSProp: Root Mean Square Propagation

SGD: Stochastic Gradient Descent

HSV: Hue, Saturation and Value

PIL: Python Imaging Library

Table of Contents

1.Introduction

Each individual utilize language to communicate with others. The listening to disabled individuals likewise utilize language to communicate among themselves. Sign language is basically utilized by deaf-mute people to communicate with each other, developed by deaf communities. Sign language enable the deaf-mute people to communicate with normal people. Normal people don't know this language so it is hard to communicate with each other. Deaf and mute people face trouble while communicating with normal people because normal people do not know the sign language.

Communication through signing is a very organized non-verbal language. Communication with typical individuals is a major impediment for them since not every ordinary people comprehend their gesture-based communication. To beat this issue, sign language recognition system is expected to assist the deaf and mute people to communicate with normal people. The deaf and mute people use this system to communicate with a normal person.

Each country has its own sign language. This community communicate with this local sign language. We have analyzed that there are some previous works are done in some sign language like American Sign Language (ASL) and Malaysian Sign Language(MSL). There is an Indian Sign Language which used by local communities of our country. We have seen that there is a very little work is done in this field. So we have encouraged to do the same. This all sign languages are different in almost all ways. Generally, ASL is done through only one hand where ISL is have almost all alphabet of two hands. This leads us to make a new system to recognize the ISL.

In India, Indian Sign Language (ISL) is the communication via gestures that is generally utilized by the deaf community. Hence, the design of the sign language recognition system will be based on the ISL, in order to suit the local people's environment.

2.Literature Review

[1] Fareed Mohd has given the glow approach to recognize the Malaysian sign language. They use accelerometer and tilt sensor to get actual position of the hand. Tilt sensor give 3D orientation and accelerometer give motion direction. This method does not use any machine learning and used preprogrammed Arduino. These data is given to Arduino Leonardo which will eventually output the meaning of gesture. This is an accurate approach but it required dedicated glove system which not reliable in real world.

[2] Lakshman, Sudharsana and Gokul have done American sign language recognition using open cv and machine learning. Open cv is used for image processing and CNN is used for predicting the gesture. Each image of the gesture or alphabet is given to CNN which will give the output label. This is comparatively easy, less accurate and more realistic approach. This method is not properly working in low light condition and some skin colored background.

[3] Sanil and Sameer have done their work on Indian sign language character recognition. They do image preprocessing using open cv and train model by CNN. This is similar kind of approach what describe above. OpenCV is used for image capturing and preprocessing. This reprocessed image is given to the CNN model for prediction. They got around 53% accuracy. The main thing here is, this is for Indian sign language because they have built their own dataset. The Indian sign dataset is hard to get. We also use this dataset for our project. The limitation of this project is same as above since both use same technology.

3.Design Approach

3.1 Specification

Convolution Neural Network:

Here we use CNN with 3 Convolution Layers followed by 3 Max Pooling Layer. After this there is a Flattening Layer which is followed by Fully connected layer. At the output layer there are 36 nodes because there are 36 classes to predict (0-9 and A-Z).

The Convolution Neural Network Specification

Property	Value
Convolution Layer (Number of feature maps)	32, 64, 128
Convolution Layer (kernel size)	3, 3, 2
Max Pooling Layer (pool size)	(2, 2)
Fully Connected Layer	128 nodes
Output Layer	36 nodes
Activation used	Softmax
Hyper parameters	
Learning rate	0.01
Batch size	1
No. of epochs	10
Other Information	
Machine learning toolkit	Tensorflow

Table: 1: Convolution neural network representation

Computer Vision:

Table No 2: Computer vision Specifications:

Property	Values
Region of interest	220 X 220 pixel
Skin Color Detection	Lower_HSV[108,23,82] – Higher_HSV[179,255,255]
Noise reduction	Blur, Filter2D, Dilate functions used
Data Augmentation	Rotation_range = 5 Width_shift_range = 0.2 Height_shift_range = 0.2 Rescale = 1/255

OpenCV Hand Masking:

Firstly, the frame is captured from the camera of the user. In every frame, there is ROI (Region of Interest) which is highlighted using Green colored rectangle box. Now user has to do their Signs in that ROI. Now let's understand how hand masking is done. Now we have set the HSV skin color range [108, 23, 82] – [179, 255, 255]. Now we take every pixel and compare the pixel color value. If the pixel color value is in the range, then it converts the color to white pixel color and if pixel color is not in range then it converts to Black color pixel.

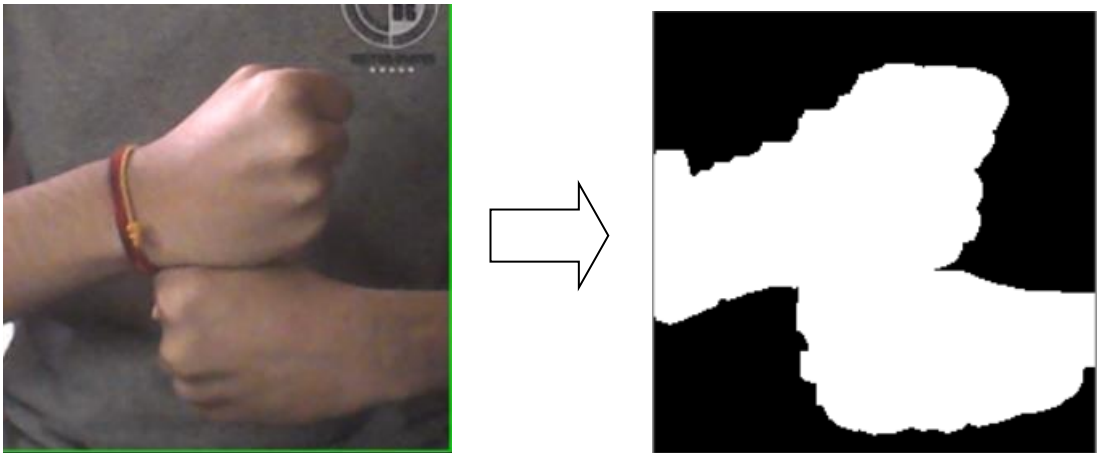

Figure 1.1: Original Hand Image **Figure 1.2: Hand Mask Image**

Figure 1: Figures of OpenCV Masks

3.2 Analysis and implementation strategy

There are many approaches to implement this project. Using machine learning we can implement it. Some potential ways to implementation are following.
There are mainly two types to implement the sign language. First is to do prediction of alphabet of sign language and second is prediction of gesture like, time, what, food etc. This people generally use gesture based sign language but to implement this system is very complex. Since there is no dataset for these gestures, it is very hard to build the model which predict the gesture.

We have chosen sign language recognition for the alphabet not gestures. But still we can do gesture recognition of sign language using some following ways.

1: If we have dataset of this gestures in form of small videos we can do 3D convolution neural network technique to train our model. In this technique, it is similar to 2D CNN but the one extra dimension is time. This method works well but the main problem in this method is, it is time and space consuming. The time taken in training is too much.

2: Here we can also use CNN and RNN both. The output of last convolution layer is given to the RNN as sequence. This is known as Long-term Recurrent Neural Network(LRNN).

We have chosen to recognize the alphabet of sign language. To do so, we use convolution neural network to train our model. But to implement this there are some other options also which are described shorty as follows.

1: Using motion sensor, gyro sensor. These sensors will give actual position and motion of hand. After getting this information we can predict which gesture is performed by these people. This required knowledge of various sensors and a dedicated product for implementation which we are not planning so we do not choose this method.

2: Using Microsoft Kinect. Kinect is a device which give position of hand and motion. So from this data we can predict which motion is done by the person. But the problem with this method is we required Microsoft Kinect at user side to recognize the sign which not possible for all users out there. So, we do not use this method to implement our project.

3: With computer vision. In this method, the image is captured from camera and convert it into hsv image and give it to model which will perform prediction. We choose this method because this involve very less amount of hardware and very less complex for implementation. Although this method is less accurate than other method but it is easy to implement and easy to access because everyone has smart phone which have camera in it.

Computer Vision:

Computer vision is a discipline that studies how to reconstruct, interrupt and understand a 3d scene from its 2d images, in terms of the properties of the structure present in the scene. "opencv" is an open source python library that provide this functionality. We use this to capture the photo and preprocess before passing it to the CNN model. Which functionalities we use from this library are following.

1: Capture the frame. We capture the frame from real time. This frame image is then preprocessed.

2: Region of interest. This is the region in which the user has to do sign. Whatever the sign is done in this region will be going forward. The sign outside this region is not considering. This process is called masking.

3: Skin color detection. Whichever pixels are in range of skin color given above are made white and other become black. This step make image black and white which known as hsv image.

4: Noise reduction. We discard the spreaded white pixels and make them black. We put white pixel instead of black if the black pixels are surrounded by lots of white pixels. This will give nice close image which have less noise in it.

Convolution Neural Network:

The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. In Deep learning, a convolutional Neural

Network is a class of deep neural network, most commonly applied to analyzing visual imagery. Convolutional Neural Network has had Ground breaking results over the past decade in a variety of fields related to pattern recognition; from image processing to voice recognition. Convolutional neural network is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm.

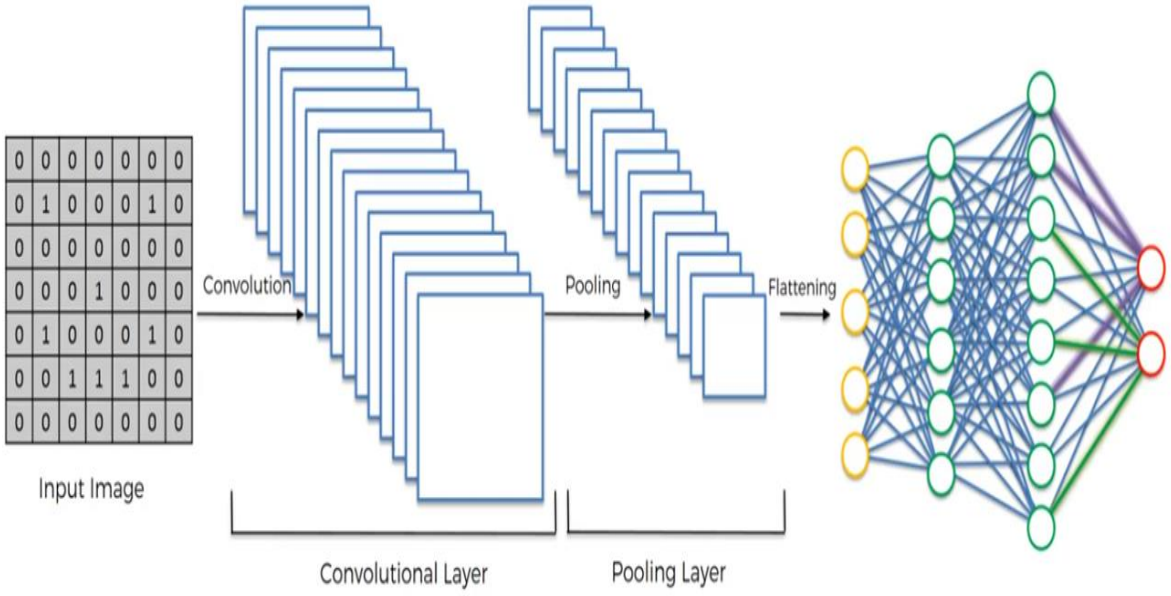

Figure 2: Architecture of Convolution Neural Networks

CNN Model Layers:

1. **CNN- Convolution Layer**

When programming a CNN, each convolutional layer within a neural network should have the following attributes:

- Input is a tensor with shape (number of images) x (image width) x (image height) x (image depth).
- Convolutional kernels whose width and height are hyper-parameters, and whose depth must be equal to that of the image. Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.

2. CNN- Max Pooling

Pooling layers reduce the dimensions of the data by combining the outputs. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer.

3. CNN- Flattening

Flattening creating a vector of pooled Feature map.

4. CNN- Fully Connected Layers

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

3.3 Work plan

The following work was done in each week:

- Week 1:
    - Requirement gathering
- Week 2:
    - Data collection
    - Implementation method selection
- Week 3:
    - Information gathering about Dense pose
- Week 4:
    - Initial alphabet recognition for American sign language
- Week 5:
    - Documentation of SRS
- Week 6:
    - Increase accuracy of CNN model for ASL
- Week 7:
    - Hand detection (Skin detection) using OpenCV
- Week 8:
    - Create a new CNN model for ISL
    - Try to increase accuracy for testing
- Week 9:
    - Noise reduction in hand masking
- Week 10:
    - Combine 25 predictions of frames to do accurate prediction
- Week 11:
    - Due to less amount of data we have do data augmentation
- Week 12:
    - Documentation of Sign Language Recognition project.

3.4 Tools required

Python 3.6

The python is open source object oriented programming language having rich libraries mentioned below.

Libraries

- **Numpy**

  Numpy is a library provided by python for scientific computation tasks.

  It provides versatile data structure likes array object and various derived objects along with it, it enables to perform various tasks like mathematical operations, logical, shape manipulation, sorting, selection etc.

- **TensorFlow**

  TensorFlow is a python library which provide faster algorithms to train and test machine learning models. TensorFlow is used by ML agents for training of the models.

- **Keras**

  Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

- **PIL**

  Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

4.Implementation

4.1 Methodology

The communication between the user and the system occurs as follows:

- The user makes the gestures in front of the camera which captures the Depth Image and separates the gestures from the entire scene.
- The important features such as position of hand and fingers are extracted to make decision on what is being gestured.
- The Gesture Recognition is performed using Convolutional Neural Networks, by classifying the gestures.
- The sentences are interpreted from the gesture and finally sent to the user interface which displays/speaks the sentence.

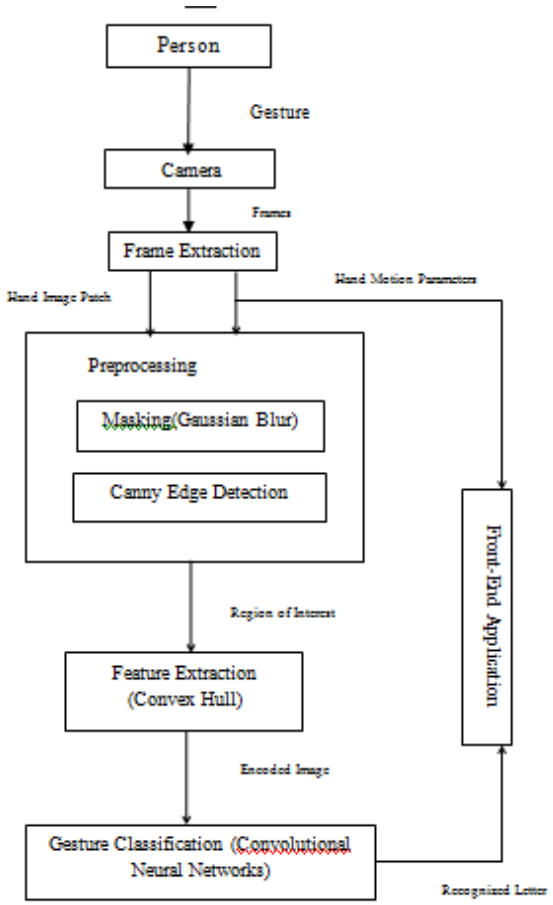

Figure 3: Block Diagram for Sign Language Recognition

A. Pre-processing of input data

Each frame of image is pre-processed by cropping it by 110x110 pixels and then Gaussian blur is applied for smoothing the image frames. The Gaussian blurred images are then further smoothened using median blur function to obtain a black and white image. These frames are then processed for obtaining the contours, which can be used to extract the features.

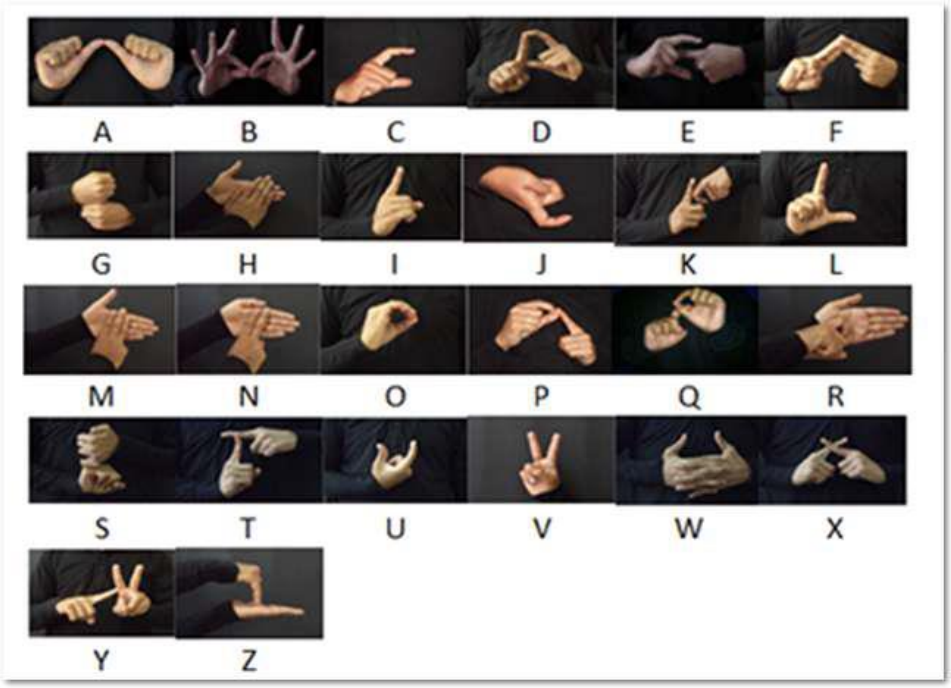

Figure 4: The ISL Manual Alphabets

g

Fig.2 shows a list of words and their labels for classification that will be used for predicting the gestures. The words will be constructed based upon the labels listed in the above table which is stored in an SQLite database.

B. Gaussian Blur

To perform a smoothing operation, we will apply a filter to our image. The most common type of filters is linear, in which an output pixel's value (i.e.) is determined as a weighted sum of input pixel values (i.e. f(i+k,j+1));

$$g(i,j) = \sum_{k,l} f(i+k, j+l) h(k,l)$$

h(k,1)is called the kernel, which is nothing more than the coefficients of the filter.

It helps to visualize a filter as a window of coefficients sliding across the image. Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array.

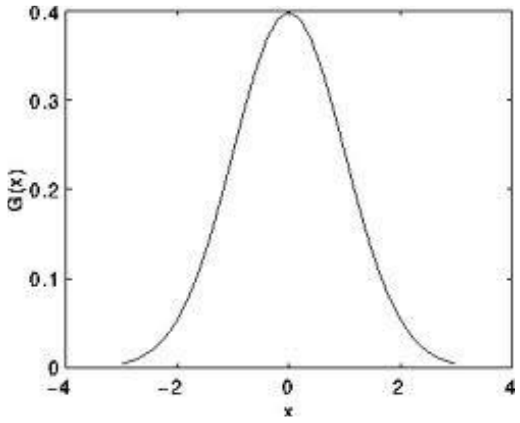

Figure 5: Example of 1D Gaussian curve

A 2D Gaussian can be represented as:

$$G_0(x,y) = Ae^{\left(\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}\right)}$$

Figure 6: Photos of implementation original, skin dataset and HSV images after preprocessing.

4.2 Steps of Implementation

Following are the steps of our project model creation:

Step 1. Install all the necessary libraries required in project.

- **a.** Numpy
- **b.** Keras
- **c.** PIL
- **d.** TensorFlow
- **e.** OpenCV
- **f.** Joblib (for saving model)

Step 2. Firstly import Sequential Class from keras.model to work with CNN.

Step 3. Now we have to import layers of CNN like Convolution Layer, MaxPooling layer, Flattening layer, Dense layer for fully connected layer.

```
# Importing all the libraries
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Initialising the CNN
classifier = Sequential()
```

Step 4. Now we have to initialize the CNN hence we will create 'classifier' model. Now add Conv2D layer followed by MaxPooling2D. At the end add Flattening layer Now the output of Flattening layer will be given to Fully Connected Layer.
At the end, add output layer. Here, we have 36 classes to classify (0-9 and A-Z). Hence, we have 36 nodes in output layer. Now all the layers are added. So, compile the model.

```
# Step 1 - Convolution
classifier.add(Conv2D(32, kernel_size=3, activation='relu', input_shape=(110, 110, 1)))

# Step 2 - Max Pooling
classifier.add(MaxPooling2D(pool_size = (2,2)))

# Adding extra convolution layers
classifier.add(Conv2D(64, kernel_size=3, activation='relu'))
classifier.add(MaxPooling2D(pool_size = (2,2)))
classifier.add(Conv2D(128, kernel_size=2, activation='relu'))
classifier.add(MaxPooling2D(pool_size = (2,2)))

# Step 3 - Flatten
classifier.add(Flatten())

# Step 4 - Fully Connected Layer
classifier.add(Dense(128, activation='relu'))
classifier.add(Dropout(0.3))
classifier.add(Dense(36, activation='softmax'))

# Compile the Model
classifier.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

Step 5. Now comes the part of Data Augmentation. To get the functionality of data augmentation, import ImageDataGenerator class from keras.preprocessing.image library.

Step 6. Here we have to specify the different Data Augmentation techniques like rotation, width shift, height shift, rescale value, etc.

```
# Keras Image Preprocessing
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
            rotation_range=5,
            width_shift_range=0.2,
            height_shift_range=0.2,
            rescale=1/255,
            validation_split=0.2
        )

test_datagen = ImageDataGenerator(rescale=1./255)
```

Step 7. Here to apply data augmentation to our dataset, we use flow_from_directory() function. This function takes several argument like directory path, target size, batch size, color mode, etc.

```
train_generator = train_datagen.flow_from_directory(
        'ISL Gestures DataSet',
        target_size=(110, 110),
        batch_size=1,
        color_mode='grayscale',
        class_mode='categorical',
        subset='training')

validation_generator = train_datagen.flow_from_directory(
        'ISL Gestures DataSet',
        target_size=(110, 110),
        batch_size=1,
        color_mode='grayscale',
        class_mode='categorical',
        subset='validation')
```

Step 8. Now fit the model using classifier.fit_generator() function. This also takes several argument like steps_per_epoch, total number of epochs, etc. At the end save the model using Joblib.dump() function.

```
classifier.fit_generator(
        train_generator,
        steps_per_epoch=len(train_generator.filenames),
        epochs=10,
        validation_data=validation_generator,
        validation_steps=len(train_generator.filenames))
```

Following are the steps of our project hand masking and prediction:

Step 1. Firstly, import cv2, numpy, joblib, and Image from PIL library.

```
# Import the Model
import joblib
model = joblib.load('ISL-CNN-Model2')
alpha = {}
count = 0
```

Step 2. Load the previously saved model using joblib.load()

Step 3. To capture the video use OpenCV's VideoCapture() function and continuously read frames until Break key is pressed. Here in our case Break key is Esc key.

Step 4. Take Region of Interest and give it preprocess() function which gives hand masked image in return.

```
while True:
    ret, frame = cap.read()
    roi = frame[200:420, 200:420]
    cv2.rectangle(frame, (200,200), (420,420), (0,255,0), 0)
```

Step 5. Now this masked image is given to trained model, which will give predicted output.

```
classes = model.predict_classes(mask3)
if(classes>9):
    char = chr(classes-10+ord('A'))
    if char in alpha.keys():
        alpha[char] += 1
    else:
        alpha[char] = 1
    #cv2.putText(frame,chr(classes-10+ord('A')), (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
else:
    char = str(classes)
    if char in alpha.keys():
        alpha[char] += 1
    else:
        alpha[char] = 1
    #cv2.putText(frame,str(classes), (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)

cv2.imshow('Original', frame)
cv2.imshow('Mask', mask)
count += 1
```

Step 6. To increase accuracy of predicted result, we take maximum occurrence of Predicted character of last 25 frames.

```
if count == 25:
    count=0
    maxAlpha = max(alpha,key = alpha.get)
    #cv2.putText(frame,maxAlpha, (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
    alpha.clear()

cv2.putText(frame,maxAlpha, (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
```

4.2 Result

We have make our own CNN model. We have tried different number of Convolutional layers, optimizers and Number of total nodes in Fully Connected layer. The image captured by camera is given for preprocessing. The part of image which is inside the region of interest is considered other part of image is discarded. We have used various optimizers to check accuracy. We also tried different number of convolution layers and check various accuracy. 110 x 110 pixel image is given to CNN model and it predicts Alphabet.

Table 3: Result using different optimizer

No of Convolution Layers in CNN	Optimizer	Accuracy(%)
3	RMSPROP	69.11
3	SGD	82.97
3	Adam	90.23

To optimize our result further, we use one more technique as follows:

The prediction result is fluctuating many time in 1 sec. so to reduce fluctuations we do not print predicted character every time we predict. We combine 25 prediction results to predict final result. We compute prediction for each frame in real time. This prediction is added in list. Whenever 25 predictions are made we see which character is occurred maximum time in this list. We print this maximum occurring prediction as our final result. This method increases the accuracy of predicted result. This make our project more reliable in real time prediction. This method print prediction almost every second. This method removes the fluctuation of our predicted result in real time so it is more reliable in nature.

Results of Model using RMSPROP Optimizer:

The original 220 x 220 pixel image is resize to 110 x 110 pixel. This image is converted into HSV image by skin detection. Noise is reduced in this image and given to the CNN model as input. The following are the result of this using RMSProp as an optimizer for CNN. The predicted output is shown in top screen on original capturing frame.

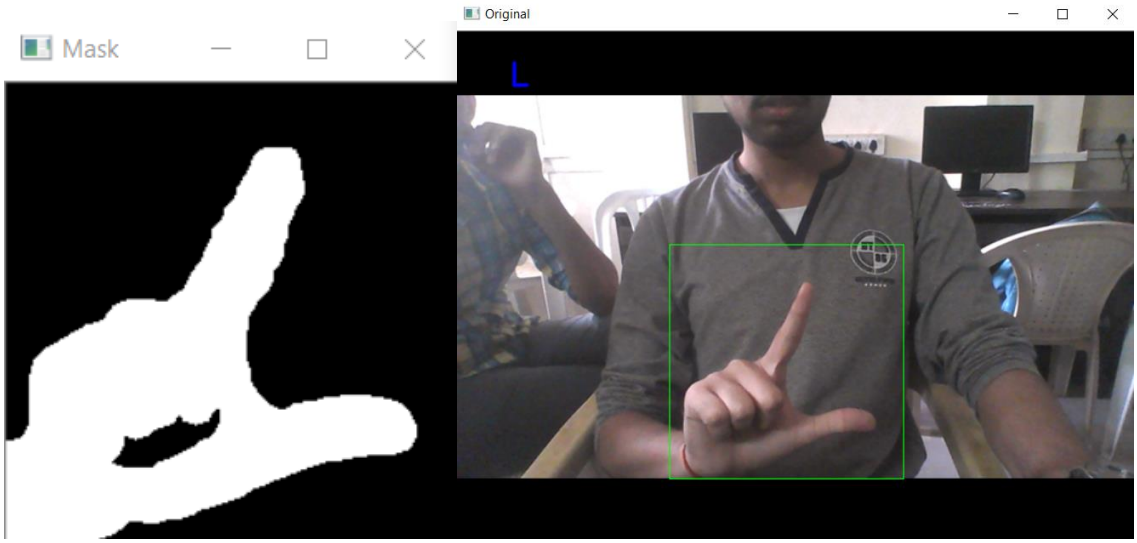

Figure 7.1: **Original Alphabet L, Predicted L**

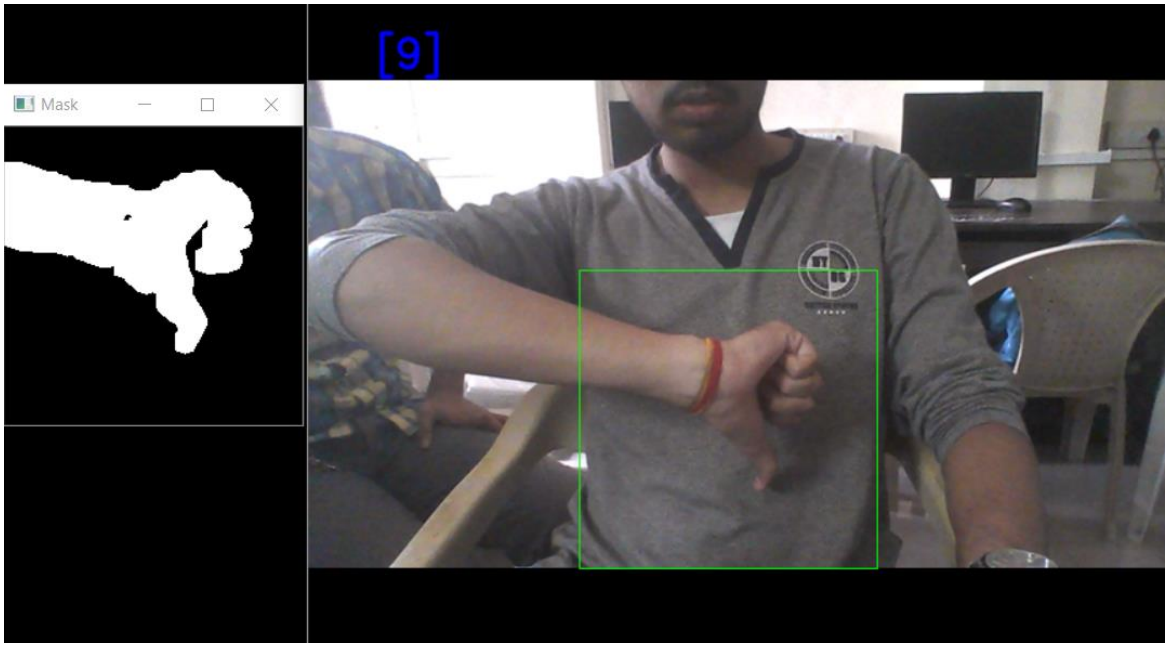

Figure 7.2: Original Digit 9, Predicted 9

Results of Model using SGD:

The original 220 x 220 pixel image is resize to 110 x 110 pixel. This image is converted into HSV image by skin detection. Noise is reduced in this image and given to the CNN model as input. The following are the result of this using SGD as an optimizer for CNN. The predicted output is shown in top screen on original capturing frame.

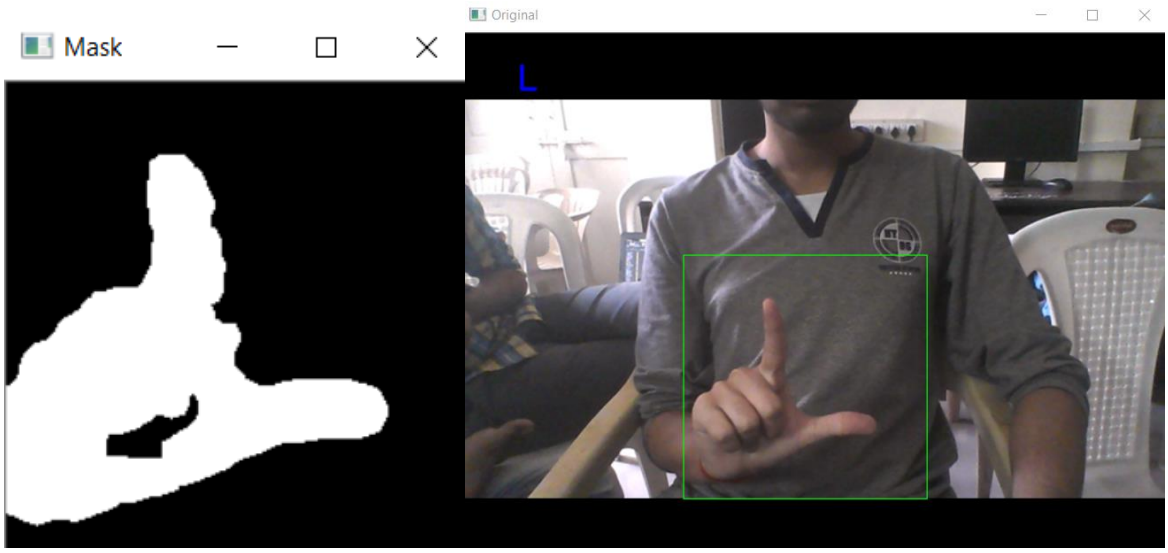

Figure 8.1: Original Alphabet L, Predicted L

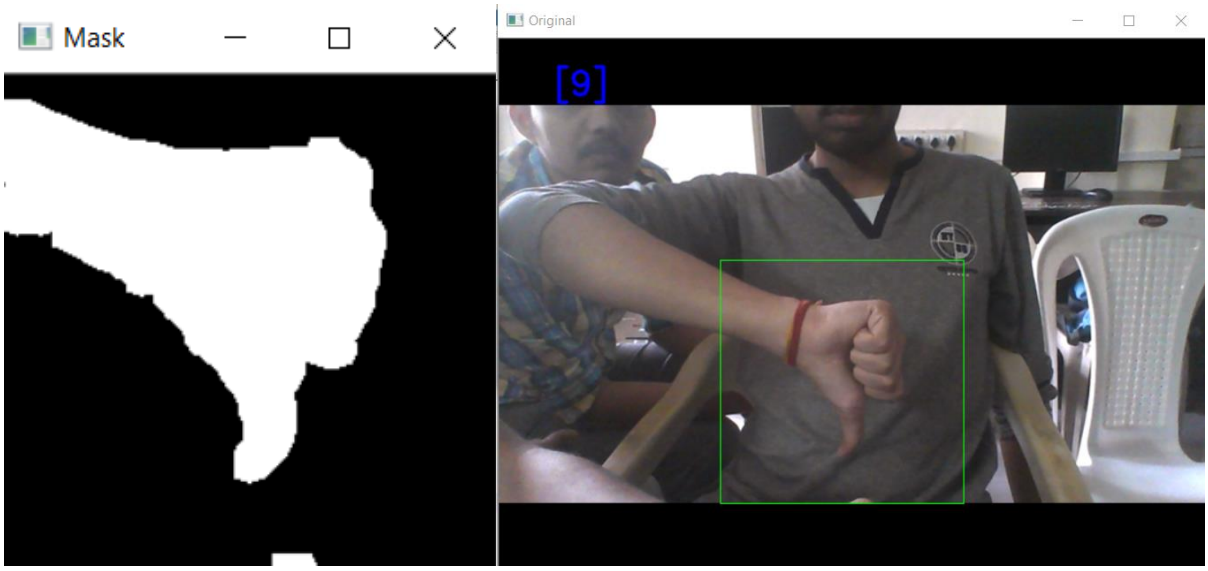

Figure 8.2: Original Digit 9, Predicted 9

Results of Model using Adam Optimizer:

The original 220 x 220 pixel image is resize to 110 x 110 pixel. This image is converted into HSV image by skin detection. Noise is reduced in this image and given to the CNN model as input. The following are the result of this using Adam as an optimizer for CNN. The predicted output is shown in top screen on original capturing frame.

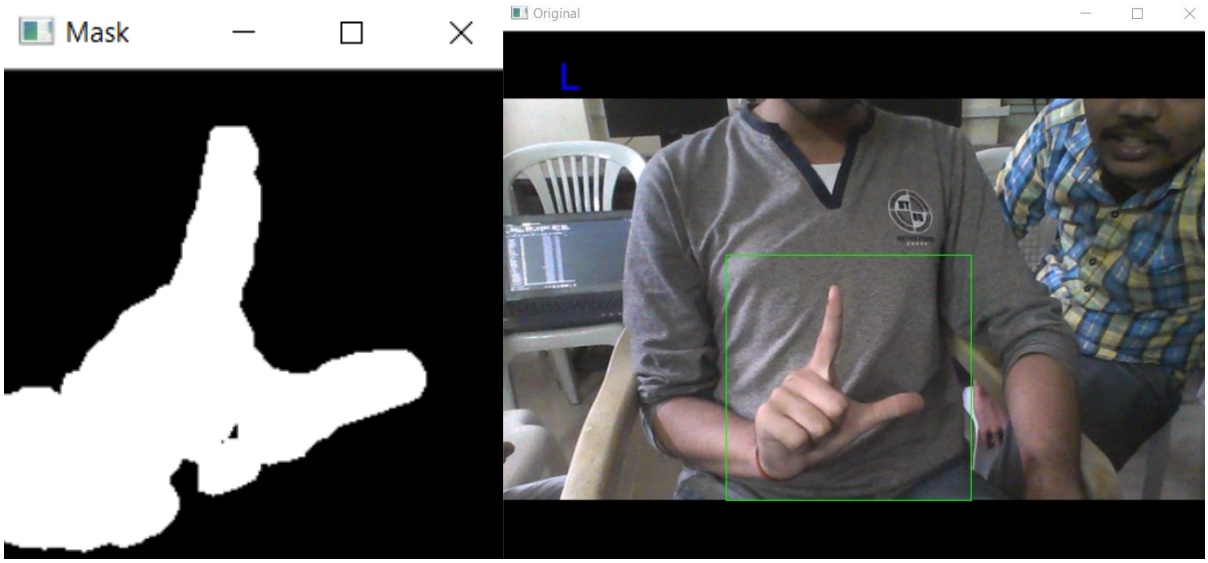

Figure 9.1: Original Alphabet L, Predicted L

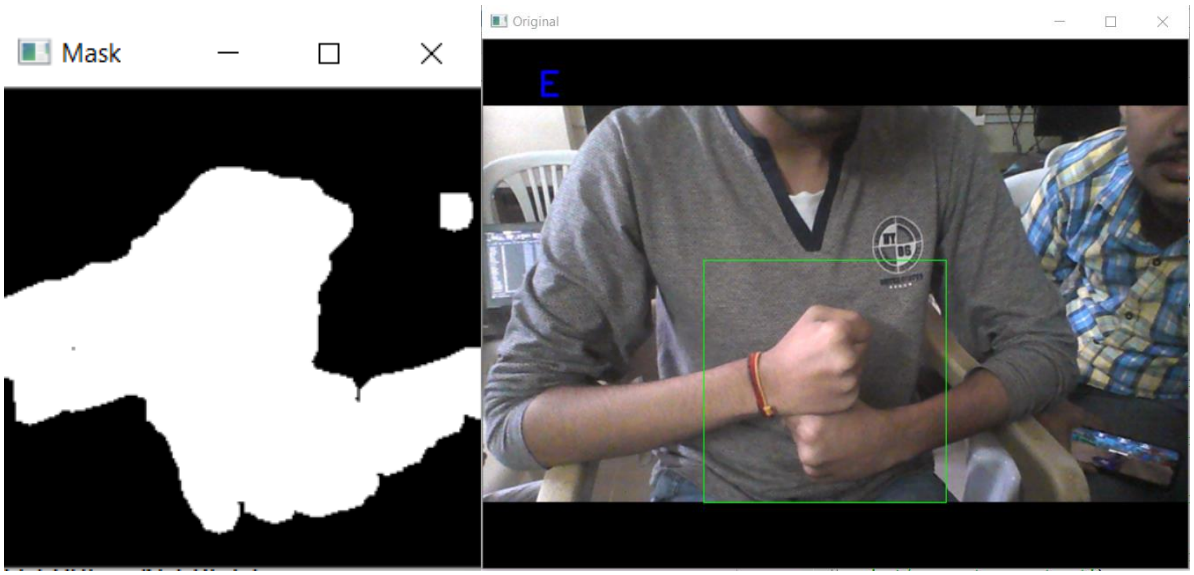

Figure 9.2: Original Alphabet G, Predicted E

5. Conclusion

The theoretical accuracy of Model using RMSProp optimizer is 69.1% while accuracy of SGD and Adam are 82.97% and 90.23% which are significantly greater. But in practical experience the best result is given by Model which uses RMSProp as optimizer. There are much inaccurate results in SGD and Adam as compared to RMSProp. As we use skin color range to mask the hand, some colors like Yellow, Red, etc. which are in range of skin color also get detected in Hand Mask. So in background we need specific colors which fall outside of skin color range. There is another limitation in project, some signs are similar in HSV color format and hence there is confusion and will not give appropriate prediction.

References

[1] MOHD FAREED ASYRAF BIN MOHD HASNI, “Sign Language Recognition System for Deaf and Dumb people”, June 2015.
Available:
http://eprints.utem.edu.my/18235/1/Sign%20Language%20Recognition%20System%20For%20Deaf%20And%20Dumb%20People%2024%20Pages.pdf

[2] LAKSHMAN KARTHIK RAMKUMAR, SUDHARSANA PREMCHAND, GOKUL KARTHI VIJAYAKUMAR, “Sign Language Recognition using Depth Data and CNN”, published in SSRG International Journal of Computer Science and Engineering (SSRG – IJCSE) – Volume 6 Issue 1 – January 2019.
Available: http://www.internationaljournalssrg.org/IJCSE/2019/Volume6-Issue1/IJCSE-V6I1P102.pdf

[3] SANIL JAIN, K. V. SAMEER RAJA, “Indian Sign Language Character Recognition”.
Available: https://www.cse.iitk.ac.in/users/cs365/2015/_submissions/vinsam/report.pdf

[4] NEHA V. TAVARI, P. A. V. D. Indian sign language recognition based on histograms of oriented gradient. International Journal of Computer Science and Information Technologies
5, 3 (2014), 3657–3660.
Available:
https://pdfs.semanticscholar.org/b188/e93f8aa17a6c415bb2e558ee973538510f25.pdf

[5] ABHISHEK SINGH, “Getting Alexa to Respond to Sign Language Using Your Webcam and TensorFlow.js”, 8 August 2018.
Available: https://medium.com/tensorflow/getting-alexa-to-respond-to-sign-language-using-your-webcam-and-tensorflow-js-735ccc1e6d3f

[6] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278 – 2324.
Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[7] SHIKHA SINGHAL, SHALAKHA SINGHAL, “Hand Gesture Recognition using Depth Data for
Indian Sign Language”, May 2013.
Available: http://ethesis.nitrkl.ac.in/5033/1/109EC0244.pdf