VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

PROJECT REPORT

on

"SIGNAL TO SPEECH"

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION ENGINEERING

For the academic year 2019-2020

Submitted by

S PRAVEEN	**(1CR16EC138)**
S SHREYAS	**(1CR16EC140)**
SAMRIDHI SINGH	**(1CR16EC146)**
URVI R	**(1CR16EC207)**

Under the guidance of

Mr. CHETAN H
Associate Professor
Dept. of ECE, CMRIT

2019-2020
Department Of Telecommunication Engineering
CMR INSTITUTE OF TECHNOLOGY, Bangalore - 560 037

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people whose proper guidance and encouragement has served as a beacon and crowned my efforts with success. We take an opportunity to thank all the distinguished personalities for their enormous and precious support and encouragement throughout the duration of this seminar.

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bangalore** for providing us an opportunity to carry out our project work.

We have great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

With profound sense of gratitude, we acknowledge the guidance and support extended by **Mr. R. Elumalai,** HoD, and **Mr. Chetan H**, Associate Professor, Department of Electronics & Communication Engineering, CMRIT, Bangalore. Their incessant encouragement and invaluable technical support have been of immense help in realizing this project work. Their guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hard work.

We also extend our thanks to the faculties of ECE Department who directly or indirectly encouraged us throughout the course of project work.

We also thank our parents and friends for all their moral support they have given us during the completion of this work.

CERTIFICATE

This is to certify the Project work entitled “**Signal to Speech**”, carried by the following bonafide students of **CMR Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of **Bachelor of Engineering in Electronics & Communication Engineering** of the **Visvesvaraya Technological University, Belagavi-590018** during the academic year 2019-20. This is certified that all the corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements prescribed for the said degree.

1. S Praveen	(1CR16EC138)
2. S Shreyas	(1CR16EC140)
3. Samridhi Singh	(1CR16EC146)
4. Urvi R	(1CR16EC207)

Signature of Guide	**Signature of HoD**	**Signature of Principal**
Mr. Chetan H	**Mr. R. Elumalai**	**Dr. Sanjay Jain**
Asst. Professor	**Dept. of ECE, CMRIT**	**CMRIT**
Dept. of ECE, CMRIT		

External Viva

Name of the Examiner with date	**Signature**

1.

ABSTRACT

Human beings interact with each other to convey their ideas, thoughts and experiences to people around them. But this is not true in the case of deaf-mute people. Sign language communication is possible for deaf-mute people without the means of acoustic sounds. The main goal behind this project is to build a system for recognizing the signs, which would help to bridge out the communication gap between deaf-mute people and normal people.

Interaction with each other helps human beings to share their intention, belief, understanding and experience to individuals around them. This is not true in the case of people with hearing or speech disability. Communication for a hearing or speech impaired person is achieved through sign language.
A gesture recognition system is proposed that can detect and perceive the hand gestures of the dead-mute and produce an audio output. This system can help to bridge out the communication gap between deaf-mute and normal people by giving a digital tongue to deaf-mute.

A gesture recognition system is proposed using which the image (hand gesture) is captured using openCV with the help of Raspberry Pi Camera and Raspberry pi then handles the tensorflow library which detects and recognizes gestures. CNN was utilized for picture classification and detection. Based on examination with the prepared model, the output is given through raspberry pi. This method has a great accuracy compared to the above methods.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Hand motion acknowledgement framework got an extraordinary consideration over the last few years owing to its manifold applications and its capability to interact efficiently with machines via human computer interaction. The fundamental point of modelling hand signal acknowledgement framework is to make a feature cooperation amidst human beings and computers where the apprehended motions can be used for managing a bot or passing on essential data.

Motions can be static (pose or certain posture) which require less computational unpredictability or dynamic (succession of stances) which are progressively intricate however reasonable for constant conditions. Various strategies have been proposed for getting data vital for acknowledgment signals framework.

The greater part of the analysts grouped motion acknowledgment framework into for the most part three stages subsequent to gaining the info image from camera(s), recordings or even information glove instrumented gadget. These means are: Extraction Method, highlights estimation and extraction, and order or acknowledgment. There are different ways of achieving this goal using different technologies.We used different technology to obtain the similar result and compare their performance and outputs.

In simple words,the gesture is captured with the help of a camera,and is transmitted to a microcontroller,where the data is processed and further sent to the server where it can be detected and recognized and compare with the data obtained from trained model.This will basically bridge out the communication gap between deaf-mute and normal people.

Chapter 2
LITERATURE SURVEY

Paper Name	Technology	Application	Future scope	Solutions	Drawbacks
Electronic Speaking Glove for Speechless Patients	Motion Synthesis Embedded System			Fills the communication gap in speech impaired and partially paralyzed patients.	Not waterproof Irritation and maceration of skin.
Hand gesture to speech conversion using MATLAB	Image segmentation & feature extraction algorithm Speech playback	Used by the speaking impaired people for communicating with normal people		Helps the speaking impaired person to communicate with normal people	Problematic during the low light conditions
Software Based Sign Language Converter	LabVIEW DroidCam Android App & Client Software Microsoft Speech Platform SDK 11	allows real-time translation with easy-to-use controls for the user.		Minimizing the communication gap between hearing impaired and normal people with ease.	Requires certain mobile specs without which it cannot work

Sign Language To Speech Conversion	Flex sensors Tactile sensor Accelerometer Atmega328 MCUs Arduino MCUs		convert words, phrases and simple sentences by concatenating the alphabets.	helps in reducing the ambiguity in gestures and shows improved accuracy.	Hard to fix Uncomfortable
Sixth Sense Technology: Applications and Comparison	Augmented reality, Computer-vision, Image processing, Visual information	Calculator, Viewing maps, Taking photos, Check time	Paper usage and electricity can be minimized to a greater extent	Helps in building a bridge between the physical world and the digital world.	
Sixth Sense Technology	WUW Gesture Recognition Computer Vision Voice Recognition.		In educational field, the number of hardware components could be reduced.	It is accessing data directly from machine in real time, hence linking the world and information.	This technology will affect the hardware market and will result in less revenue being generated and lower the cash inflow.

Table 1. Literature Survey

Chapter 3
MOTIVATION

Coming across the various spheres of life where this wearable technology can be applied, our major motivation for this project came from one such need. Among us, there are few people who cannot communicate by talking. The sign language can directly be converted to voice and thus enabling them to communicate verbally.

The motivation for this project is to develop a system which can detect the gestures and these gestures are widely used for conveying the information or to control the devices. Camera based solutions for gesture recognition have been widely used in numerous applications and capability to communicate through Human Computer Interaction.

Chapter 4

OBJECTIVE

The main objective behind this project is to develop a system for recognizing the signs, which would bridge out the communication gap between people with speech impaired and normal people.

The main aim of the project is to track the region of interest(ROI) with the help of a camera and detect the target object by using object detection algorithms. Here the target object is the hand and any gestures made by the hand is analysed using different technologies.

Chapter 5
METHODOLOGY

5.1. Sixth Sense Technology

Sixth Sense is basically a wearable gestural connection that upgrades the real world around us with digital data and lets us use hand signals to interface with that data. It depends on the ideas of augmented reality and has well implemented its view. Sixth Sense has incorporated these present reality objects with the advanced world. The remarkable sixth sense innovation is a mixture of numerous advancements. The thing which makes it impressive is the wonderful combination of every one of those advances and turns it into a solitary convenient and conservative item. It links technologies like hand motion acknowledgment, picture catching, handling, and control, and so forth. It superimposes the digital world on this present world.

5.2. HAAR Cascade Algorithm

HAAR cascade algorithm is a machine learning object detection algorithm which is used to identify the objects in an image or a video. In this algorithm a cascade function is trained from positive and negative images. Positive image is the one which consists of the object to be detected and the negative image is the one in which there is the remaining portion other than the object.

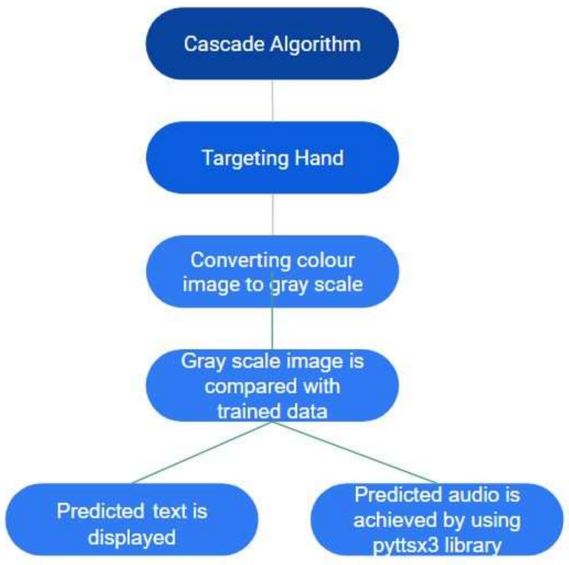

Cascade Algorithm
Targeting Hand
Converting colour image to gray scale
Gray scale image is compared with trained data
Predicted text is displayed
Predicted audio is achieved by using pyttsx3 library

Fig 5.1 Cascade algorithm flow diagram

As per the above flow diagram, Cascade algorithm targets the hand and it converts the colour image into a grayscale level. This gray scale image is compared with a trained data set and the predicted output is displayed on the screen in the form of text and the audio output is achieved using pyttsx3 library. The output obtained using this algorithm is discussed in the RESULT chapter.

5.3. TENSOR FLOW

The Hand Gesture Recognition Database is a collection of near-infra-red images of ten distinct hand gestures. In this notebook we use end-to-end deep learning to build a classifier for these images.

We'll first load some packages required for reading in and plotting the images.

In [1]:

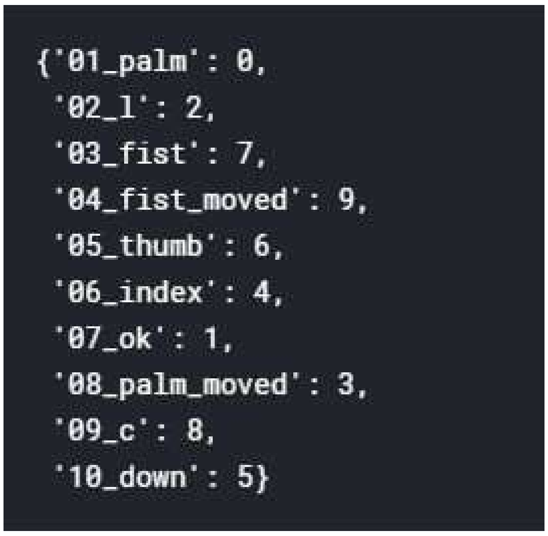
```
importnumpyasnp                    # We'll be storing our data as numpy arrays

importos                           # For handling directories

from PIL import Image              # For handling the images

importmatplotlib.pyplotasplt

importmatplotlib.imageasmpimg      # Plotting
```

As described in the Data Overview, there are 10 folders labelled 00 to 09, each containing images from a given subject. In each folder there are subfolders for each gesture. We'll build a dictionary lookup storing the names of the gestures we need to identify, and giving each gesture a numerical identifier. We'll also build a dictionary reverse lookup that tells us what gesture is associated with a given identifier.

In [2]:

```
lookup=dict()

reverselookup=dict()
```

```
count=0

for j inos.listdir('../input/leapgestrecog/leapGestRecog/00/'):

ifnotj.startswith('.'):              # If running this code locally, this is to

# ensure you aren't reading in hidden folders

lookup[j] = count

reverselookup[count] = j

count= count +1

lookup

Out[2]:
```



```
{'01_palm': 0,
 '02_l': 2,
 '03_fist': 7,
 '04_fist_moved': 9,
 '05_thumb': 6,
 '06_index': 4,
 '07_ok': 1,
 '08_palm_moved': 3,
 '09_c': 8,
 '10_down': 5}
```

Fig 5.2 Data Stored Format

Next we read in the images, storing them in x_data. We store the numerical classifier for each image in y_data. Since the images are quite large and are coming from an infra-red sensor, there's nothing really lost in converting them to grayscale and resizing to speed up the computations.

In [3]:

```
x_data= []

y_data= []

datacount=0                    # We'll use this to tally how many images are in our dataset

for i inrange(0, 10):          # Loop over the ten top-level folders

for j inos.listdir('../input/leapgestrecog/leapGestRecog/0'+str(i) +'/'):

ifnotj.startswith('.'):        # Again avoid hidden folders

count=0                        # To tally images of a given gesture

for kinos.listdir('../input/leapgestrecog/leapGestRecog/0'+

str(i) +'/'+ j +'/'):

# Loop over the images

img=Image.open('../input/leapgestrecog/leapGestRecog/0'+

str(i) +'/'+ j +'/'+ k).convert('L')

# Read in and convert to greyscale

img=img.resize((320, 120))

arr=np.array(img)

x_data.append(arr)

count= count +1

y_values=np.full((count, 1), lookup[j])

y_data.append(y_values)

datacount=datacount+ count
```

```
x_data=np.array(x_data, dtype='float32')

y_data=np.array(y_data)

y_data=y_data.reshape(datacount, 1)            # Reshape to be the correct size
```

Let's take a look at some of the pictures. Since each of the subfolders in 00 contained 200 images, we'll use the following piece of code to load one image of each gesture.

In [4]:

```
from random importrandint

for i inrange(0, 10):

plt.imshow(x_data[i*200 , :, :])

plt.title(reverselookup[y_data[i*200 ,0]])

plt.show()
```

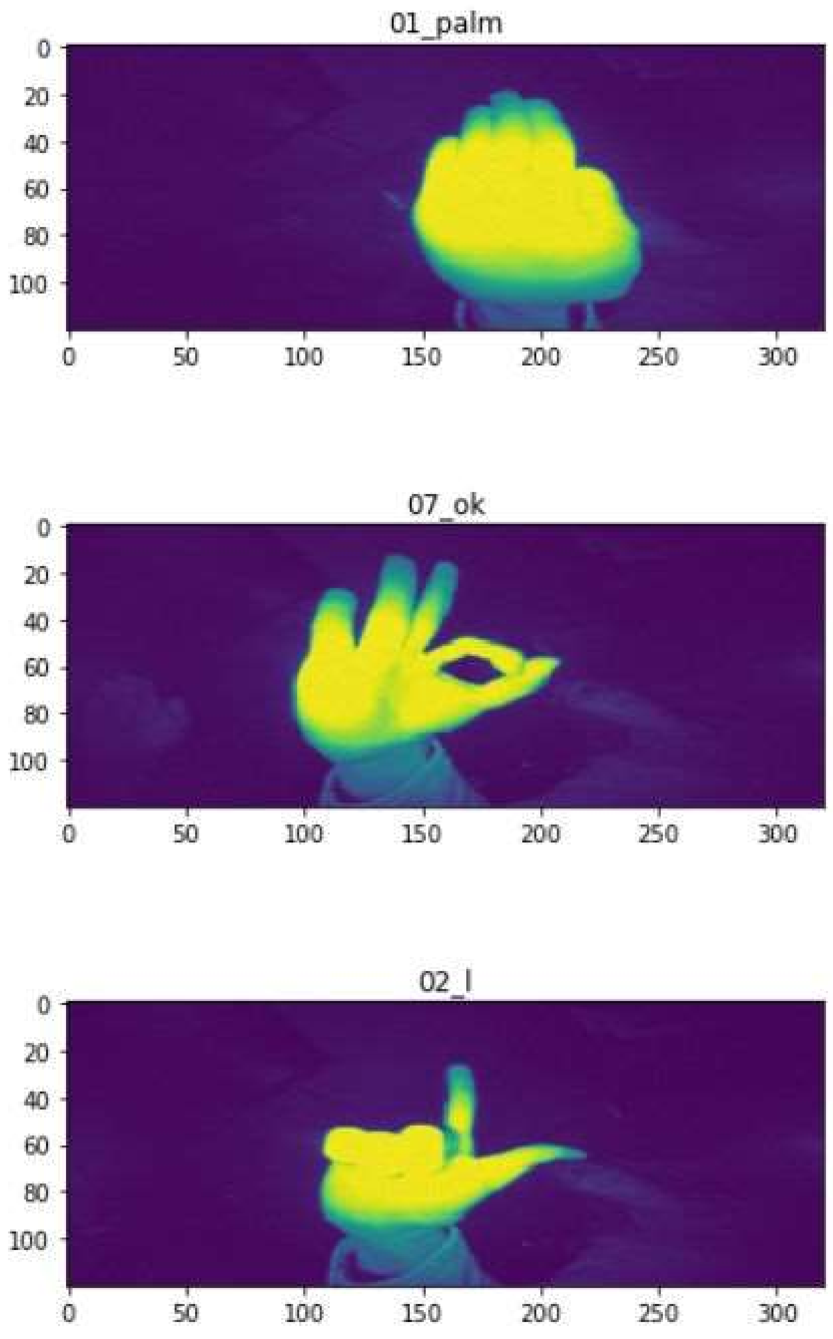


Fig 5.3 Infrared Image

The first thing to note is that this is not a difficult classification problem. The gestures are quite distinct, the images are clear, and there's no background whatsoever to worry about. If you weren't comfortable with deep learning, you could do quite well with some straight-forward feature detection -- for example the '07_ok' class could easily be detected with binary thresholding followed by circle detection.

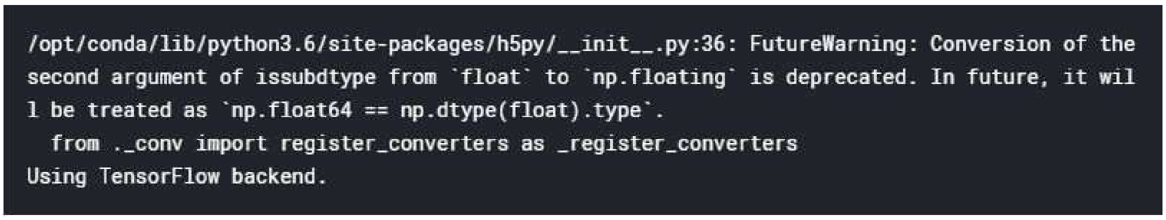
Moreover, the gestures consistently occupy only about 25% of the image, and all would fit snugly inside a square bounding box. Again if you're looking to do basic feature detection, an easy first step would be to write a short script cropping everything to the relevant 120 x 120 square.

But the point of this notebook is to show how effective it is to just throw a neural network at a problem like this without having to worry about any of the above, so that's what we're going to do.

At the moment our vector y_data has shape (datacount, 1), with y_data[i,0] = j if the ith image in our dataset is of gesture reverselookup[j]. In order to convert it to one-hot format, we use the keras function to_categorical:

In [5]:

```
importkeras

fromkeras.utilsimportto_categorical

y_data=to_categorical(y_data)
```



```
/opt/conda/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the
second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it wil
l be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Fig 5.4 Fitting

Our set of images has a shape (datacount, 120, 320). Keras will be expecting another slot to tell it the number of channels, so we reshape x_data accordingly. We also rescale the values in x_data to lie between 0 and 1.

In [6]:

```
x_data=x_data.reshape((datacount, 120, 320, 1))

x_data/=255
```

We need a cross-validation set and a test set, and we'll use the sklearn package to construct these. In order to get an 80-10-10 split, we call train_test_split twice, first to split 80-20, then to split the smaller chunk 50-50. Note that we do this after the rescaling step above, to ensure that our train and test sets are coming from the same distribution.

In [7]:

```
fromsklearn.model_selectionimporttrain_test_split

x_train,x_further,y_train,y_further=train_test_split(x_data,y_data,test_size=0.2)

x_validate,x_test,y_validate,y_test=train_test_split(x_further,y_further,test_size=0.5)
```

Now it's time to build our network. We'll use keras.

In [8]:

```
fromkerasimport layers

fromkerasimport models
```

Since our images are big (we chose not to do any cropping) and the classification problem looks quite easy, we're going to downsample fairly aggressively, beginning with a 5 x 5 filter with a stride of 2. Note we have to specify the correct input shape at this initial layer, and keras will figure it out from then on. We won't worry about padding since it's clear that all the useful features are well inside the image. We'll continue with a sequence of convolutional layers followed by max-pooling until we arrive at a small enough image that we can add a fully-connected layer. Since we need to classify between 10 possibilities, we finish with a softmax layer with 10 neurons.

In [9]:

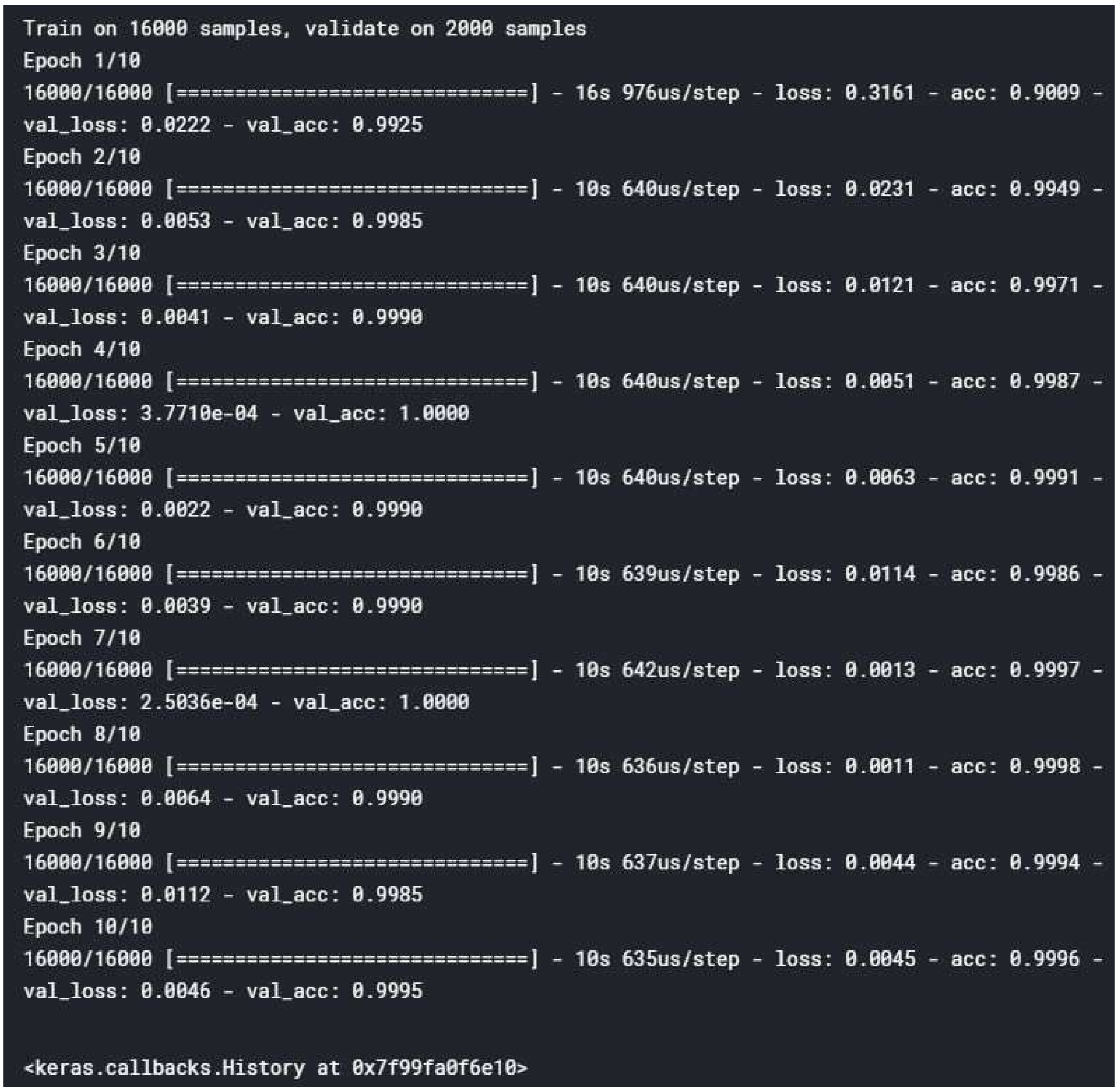
```
model=models.Sequential()

model.add(layers.Conv2D(32, (5, 5), strides=(2, 2), activation='relu', input_shape=(120,
320,1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dense(10, activation='softmax'))
```

Finally, we fit the model.

In [10]:

```
model.compile(optimizer='rmsprop',

loss='categorical_crossentropy',

metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10, batch_size=64, verbose=1,
validation_data=(x_validate, y_validate))
```

Out[10]:



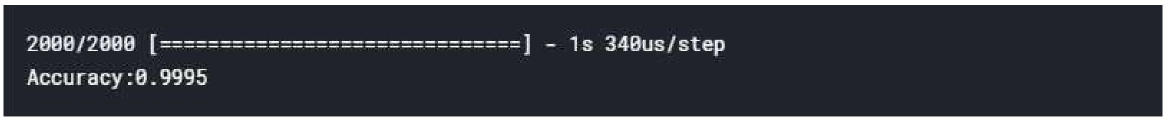
```
Train on 16000 samples, validate on 2000 samples
Epoch 1/10
16000/16000 [==============================] - 16s 976us/step - loss: 0.3161 - acc: 0.9009 -
val_loss: 0.0222 - val_acc: 0.9925
Epoch 2/10
16000/16000 [==============================] - 10s 640us/step - loss: 0.0231 - acc: 0.9949 -
val_loss: 0.0053 - val_acc: 0.9985
Epoch 3/10
16000/16000 [==============================] - 10s 640us/step - loss: 0.0121 - acc: 0.9971 -
val_loss: 0.0041 - val_acc: 0.9990
Epoch 4/10
16000/16000 [==============================] - 10s 640us/step - loss: 0.0051 - acc: 0.9987 -
val_loss: 3.7710e-04 - val_acc: 1.0000
Epoch 5/10
16000/16000 [==============================] - 10s 640us/step - loss: 0.0063 - acc: 0.9991 -
val_loss: 0.0022 - val_acc: 0.9990
Epoch 6/10
16000/16000 [==============================] - 10s 639us/step - loss: 0.0114 - acc: 0.9986 -
val_loss: 0.0039 - val_acc: 0.9990
Epoch 7/10
16000/16000 [==============================] - 10s 642us/step - loss: 0.0013 - acc: 0.9997 -
val_loss: 2.5036e-04 - val_acc: 1.0000
Epoch 8/10
16000/16000 [==============================] - 10s 636us/step - loss: 0.0011 - acc: 0.9998 -
val_loss: 0.0064 - val_acc: 0.9990
Epoch 9/10
16000/16000 [==============================] - 10s 637us/step - loss: 0.0044 - acc: 0.9994 -
val_loss: 0.0112 - val_acc: 0.9985
Epoch 10/10
16000/16000 [==============================] - 10s 635us/step - loss: 0.0045 - acc: 0.9996 -
val_loss: 0.0046 - val_acc: 0.9995


<keras.callbacks.History at 0x7f99fa0f6e10>
```

Fig 5.5 Training model

At this point we would typically graph the accuracy of our model on the validation set, and choose a suitable number of epochs to train for to avoid overfitting. We might also consider introducing dropout and regularisation. However, we can see we're getting perfect accuracy on the validation set after just one or two epochs, so we're pretty much done. Let's quickly confirm that this is carrying through to the test set:

In [11]:

```
[loss, acc] = model.evaluate(x_test,y_test,verbose=1)

print("Accuracy:"+str(acc))
```



```
2000/2000 [==============================] - 1s 340us/step
Accuracy:0.9995
```

Fig 5.6 Accuracy

You'll get slightly different numbers each time you run it but you should be getting between 99.9 and 100% accuracy. Great!

Chapter 6
RESULT

The following output was obtained using the HAAR cascade algorithm.

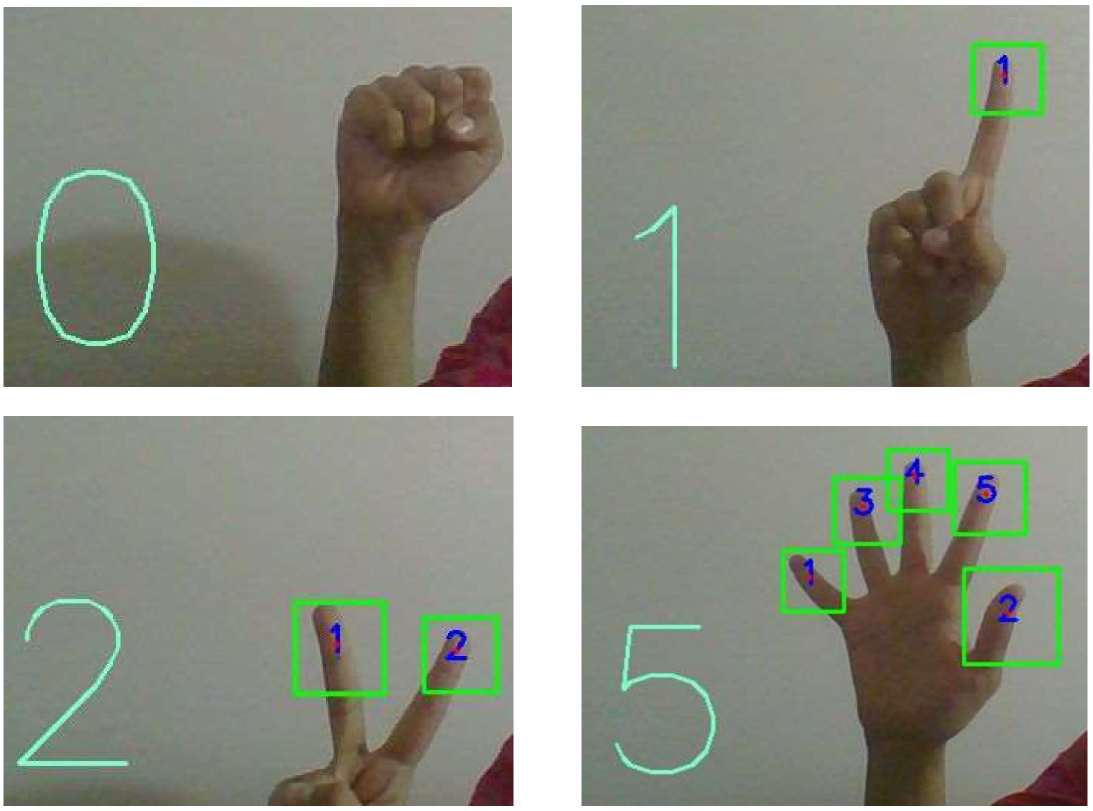


Fig 6.1 Number recognition using cascade algorithm

In this algorithm, the targeted object which is the human hand here, is recognized and a square box is formed around it. The result is displayed on the screen and audio output of the same is also achieved.

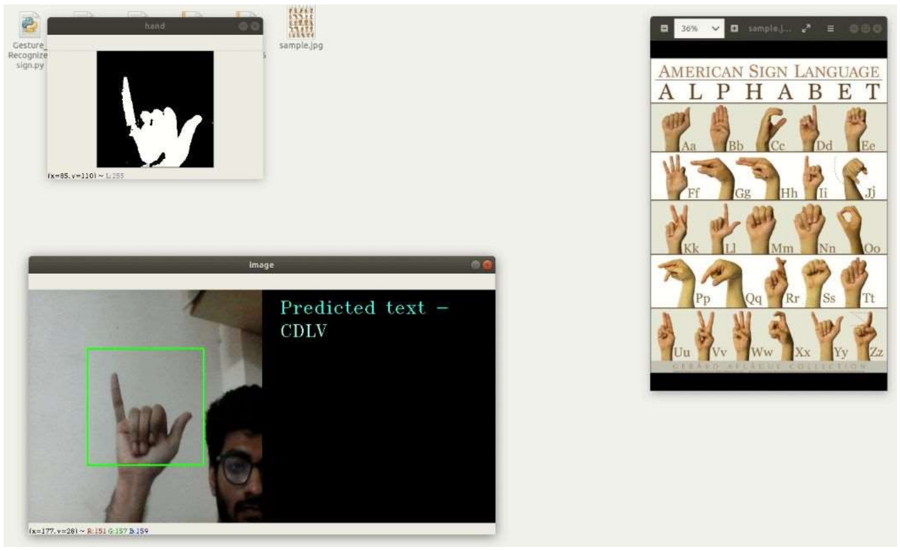

Predicted text –
CDLV

Fig 6.2 Gesture recognition using Tensorflow

In the above Fig 6.2, The hand gesture is recognised using the tensorflow library. A square box is formed around the targeted object, which is the hand here. A speech output is produced. Tensorflow library provides better accuracy when compared to the HAAR cascade algorithm.

6.1. CONCLUSION

This project introduced a flexible methodology for gesture Recognition. At first, the sixth sense approach was utilized as it has inbuilt features to perform the required task,but owing to equipment limitations,requirement of ceaseless power supply and cost of the item ,due to the following drawbacks we then proceeded with the Haar cascade algorithm.

Utilizing Haar cascade algorithm the content and sound output were achieved.The fundamental issue looked here was preparing the model.As training doesn't occur constantly in Haar cascade algorithm,training the model was not a simple task.Using CNN in tensorflow defeats this trouble.

The final method was implemented using tensorflow. The image was captured using openCV ,with the help of Raspberry pi, ,and Raspberry pi then handles the tensorflow library which detects and recognizes gestures.CNN was utilized for picture classification, object recognition and detection. Based on examination with the prepared model , the output is given through raspberry pi.This method has a great accuracy compared to the above methods.

6.2. FUTURE ENHANCEMENTS:

1. Implementing this model to work on all kinds of light conditions ,which will require a flashlight along with the camera.
2. To make it easily portable by customizing a hardware which is only for this purpose(embedded systems).
3. Training the model by adding correct data for accurate results.
4. Updating and adding new features.
5. Reducing the bugs for better efficiency.

REFERENCES

1.]Zeng, Z., Gong, Q., & Zhang, J. (2019). CNN Model "Design of Gesture Recognition Based on Tensorflow Framework." 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). doi:10.1109/itnec.2019.8729185
2. Sadman Shahriar Alam, Akib Jayed Islam, Nahid Nasrin, Khandoker Tanjim Ahammad "Hand Gesture Detection Using Haar Classifier with Appropriate Skin Color, Kernel Sizing & Auto Thresholding". International Journal of Scientific & Engineering Research, Volume 8, Issue 3, March-2017 98 ISSN 2229-5518
3. "Applying the Haar-cascade Algorithm for Detecting Safety Equipment in Safety Management Systems for Multiple Working Environments" Le Tran Huu Phuc , HyeJun Jeon, Nguyen Tam Nguyen Truong * and Jung Jae Hak * Department of Chemical Engineering, Yeungnam University, Gyeongsang-si 39541, Korea; phucleth@ynu.ac.kr (L.T.H.P.); daivjeon@ynu.ac.kr (H.J.) * Correspondence: tamnguyentn@ynu.ac.kr (N.T.N.T.);jhjung@ynu.ac.kr (J.J.H.) Received: 26August 2019; Accepted: 19 September 2019; Published: 23 September 2019
4. Sudharshan, D. P., & Raj, S. (2018). "Object recognition in images using convolutional neural networks" 2018 2nd International Conference on Inventive Systems and Control (ICISC). doi:10.1109/icisc.2018.8398912
5. Qing Chen, Georganas, N. D., & Petriu, E. M. (2008). "Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar". IEEE Transactions on Instrumentation and Measurement, 57(8), 1562–1571. doi:10.1109/tim.2008.922070
6. Gurav, R. M., & Kadbe, P. K. (2015). "Real time finger tracking and contour detection for gesture recognition using OpenCV". 2015 International Conference on Industrial Instrumentation and Control (ICIC). doi:10.1109/iic.2015.7150886
7. "Sixth Sense Technology" S.Pradeep Kumar, Asst.Prof, Rajalakshmi institute of technology, Chennai, O.Pandithurai Phd(scholar) Anna university of Technology Coimbatore.
8. Chen, Q., Georganas, N. D., & Petriu, E. M. (2007). "Real-time Vision-based Hand Gesture Recognition Using Haar-like Features". 2007 IEEE Instrumentation

& Measurement Technology Conference IMTC 2007. doi:10.1109/imtc.2007.379068

9. Panwar, M. (2012). “Hand gesture recognition based on shape parameters”. 2012 International Conference on Computing, Communication and Applications. doi:10.1109/iccca.2012.6179213
10. Sonkusare, J. S., Chopade, N. B., Sor, R., & Tade, S. L. (2015). “A Review on Hand Gesture Recognition System”. 2015 International Conference on Computing Communication Control and Automation. doi:10.1109/iccubea.2015.158