SIGN LANGUAGE TO SPEECH TRANSLATOR

A MINOR PROJECT REPORT

Submitted in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted

by

Praveen Pushkar - 00620902819

Vikas - 03620902819

Ruchita Singh, 05420902819

Under the Supervision of

Dr. Krishna Singh (Associate Professor, ECE)

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
GB PANT GOVERNMENT ENGINEERING COLLEGE
Okhla Phase-III, New Delhi-110020
January, 2023

GB PANT GOVERNMENT ENGINEERING COLLEGE
Okhla Phase-III, New Delhi-110020
ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the work embodies in this dissertation entitled '**SIGN LANGUAGE TO SPEECH TRANSLATOR'** being submitted by **'Praveen Pushkar(00620902819), Vikas(03620902819) & Ruchita Singh(05420902819)'** for partial fulfillment of the requirement for the award of '**Bachelor of Technology in Electronics and Communication Engineering** discipline to **GB Pant Government EngineeringCollege**, Okhla Phase-III, New Delhi during the academic year 2022 - 2023 is a record of bonafide piece of work, undertaken by him/her the supervision of the undersigned.

Approved and Supervised by

Signature
Dr. Krishna Singh
Associate Professor, ECE

Signature
Ms. Aman Saini
HOD ECE

EXTERNAL EXAMINER

GB PANT GOVERNMENT ENGINEERING COLLEGE

Okhla Phase-III, New Delhi-110020

ELECTRONICS AND COMMUNICATION ENGINEERING

DECLARATION

We **'Praveen Pushkar, Vikas & Ruchita Singh'**, are students of **'Bachelor of Technology in Electronics and Communication Engineering', session: 2022 - 2023, GB Pant Government Engineering College**, Okhla Phase-III, New Delhi, hereby declare that the work presented in this Project Work entitled **'Sign Language To Speech Translator'** is the outcome ofour own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: -

Praveen Pushkar, 00620902819

Vikas, 03620902819

Ruchita Singh, 05420902819

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project supervisor **Dr. Krishna Singh (Associate Professor, ECE)** for providing invaluable guidance, comments and suggestions throughout the project and for constantly motivating us to work harder and suggesting to us different ways to tackle the problems encountered. We would like to express our gratitude for the Department of Electronics and Communication Engineering, G. B. Pant Government Engineering College to provide us with the opportunity to work on this project and to all the teachers and mentors guiding us along the way.

ABSTRACT

Hand gestures are one of the non-verbal methods used in sign language. There is a lacks of efficient gesture recognition system designed specifically for the differently abled. This project is an effort towards developing a system that aims at convert such sign gestures into speech that can be understood by normal people. The entire model pipeline is developed by CNN architecture for the classification of 26 alphabets and one extra alphabet for null character. Sign language is an emerging field with a lot of scope of research in computer vision. This proposed system comprises of 3 stages as preprocessing, Feature Extraction and then Sign to text translation. It deals with bare hands, thus allowing the user to interact with the system in natural way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S. N0.	**Abbreviation**	**Full Form**
1.	ASL	American Sign Language
2.	CNN	Convolutional Neural Network
3.	HoG	Histogram of Gradients
4.	SIFT	Scale Invariant Feature Transform
5.	FV	Fisher Vector
6.	ANN	Artificial Neural Network
7.	D&M	Deaf and Mute
8.	IDE	Integrated Development Environment
9.	RAM	Random Access Memory
10.	GPU	Graphics Processing Unit
11.	RGB	Red Green Blue
12.	Relu	Rectified Linear Unit

CHAPTER 1 - INTRODUCTION

1.1 OVERVIEW

The only disability deaf and mute people have is communication related and they cannot use spoken languages hence the only way for them to communicate is through sign language.

Communication is a very vital process of exchange of thoughts and messages in various ways such as speech, signals, behavior and visuals. Deaf and dumb people make use of their hands to express different gestures to express their ideas with other people.

Gestures are the nonverbally exchanged message and these gestures are understood with vision. This nonverbal communication of deaf and dumb people is called sign language.

The main aim of this project is identifying alphabets in American Sign language from the corresponding gestures. There is well structured and elaborated work done in the field of American sign language but there are a few steps taken in the direction for Sign language. Instead of using high-end technology like gloves or kinect, the proposed project aims to solve this problem using the state of computer vision and deep learning algorithms.

Sign language has 3 major components:

1. Fingerspelling: In this system words are spelled letter by letter.
2. Word level: In this system there are some actions related to particular words.
3. Non manual features: Facial expression and tongue, mouth and body position are used for this purpose.

This project focuses on producing a model which can recognize Finger spelling-based gestures toform a complete word.

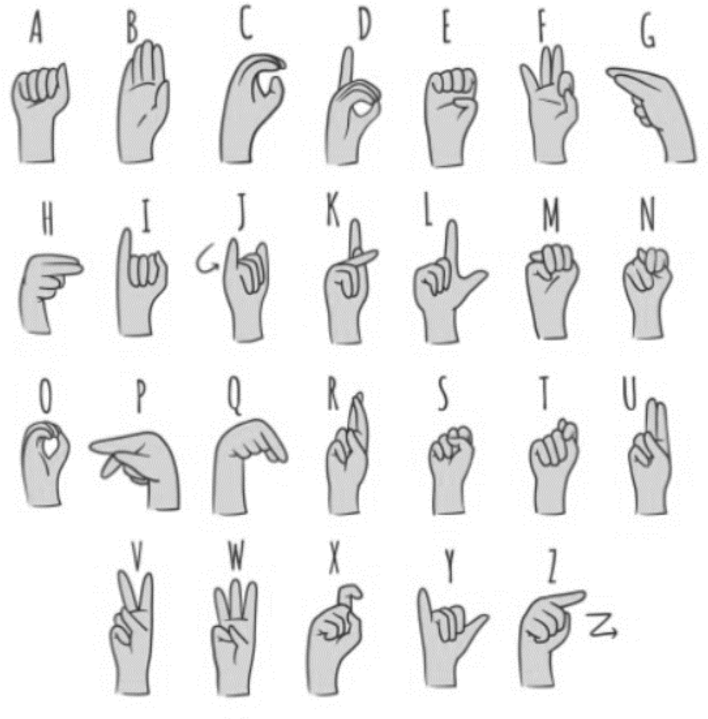


Figure 1: Sign conventions of alphabets in ASL

1.2 MOTIVATION

The World Health Organization estimates that approximately 466 million people have disabling hearing loss globally, all of whom require a means of communication. For deaf and hard-of-hearing people, sign language is the most natural way to express themselves as they rely on sign language for communication. Still, other people commonly do not learn sign language, thus leading to the isolation of disabled people. The proposed system will provide an opportunity for a deaf-dumb people to communicate with non-signing people fastly and efficiently without an interpreter and offer them a boost in developing autonomous systems for understanding and aiding them.

1.3 PROBLEM STATEMENT

The objective of the project is to bridge the communication gap between a normal person and a deaf or mute person. This can be done by simply providing the gestures through hands through the web camera after which the CNN is used to recognize and match the input with the character from the dataset on which model is trained. The simple interface will enable users to interact easily.

1.4 SCOPE OF PROJECT

- To find an algorithm which is best for American Sign language recognition.
- To Make an application which utilizes the recognition algorithm and form a sign to speech translation.
- To form characters, words, and sentences with interpretation of symbols in American Sign language in a portable and real time application

1.5 DOMAIN OF THE PROJECT

In this project, Machine learning and deep learning is used. Machine learning is an application ofartificial intelligence that includes algorithms that parse data, learn from that data, and then applywhat they've learned to make informed decisions. Deep learning is a subfield of machinelearning that structures algorithms in layers to create an artificial neural network that can learn and make intelligent decisions on its own. In this project Convolutional Neural Networks are used for the purpose of image analysis and then predicting the output.

1.6 ACCESS TO USERS

The project model is hosted with an interface. The user can show signs towards the webcam and the system is trained on the ASL Dataset and it analyses the image and produces the output. It is easy to use as the reference chart is provided alongside the window for ease of use. This is for the deaf and dumb community for the communication purpose.

CHAPTER 2 - LITERATURE REVIEW

2.1 PREFACE

In this section, some of the earlier works are presented in the direction of this project. There has been some amount of work that has already been done in the field of easing the process of hand gesture recognition. Also, various research papers, articles are studied to get a clear picture. The project proposes a system that focuses on producing a model which can recognize Finger spelling based gestures to form a complete word.

2.2 LITERATURE SURVEY

Sanil Jain and KV Sameer Raja [1] worked on Sign language Recognition, using coloured images. They used feature extraction methods like bags of visual words, Gaussian random and the Histogram of Gradients (HoG). Three subjects were used to train SVM, and they achieved an accuracy of 54.63% when tested on a totally different user.

One such approach included key point detection of Image using SIFT and matching the keypoint of a new image with keypoint of standard symbols (alphabets and numericals) to classify the new image with the closest resembled image [2].

In Another research eigenvectors of covariance matrix were calculated from the vector representation of image by the use of euclidean distance of the new image eigenvector with the images of training data to classify the new image [3].

Some of the previous researches have used Neural Networks for training their dataset. But their dataset consisted of only single handed images. Also, their feature vectors were based on the angle between fingers, number of fingers etc [4].

A hand recognition system based on local linear embedding is proposed by Xiaolong Teng, Bian Wu, Weiwei Yu, Chongqing Liu [5]. They proposed a real-time vision system within the visual interaction environment. They used general purpose hardware and low cost sensors so any user could make use of it. A novel hand shape descriptor is proposed, which is based on a set of geometric features (SoGF) and Fisher Vector (FV). They used multi-class SVM with FVs as input to construct a classifier.

In another research paper Sign language Recognition based on Histograms of Oriented Gradient [6] in this paper they used Gradient Computation, Orientation Binning. Their purpose is to implement the algorithm of HOG features, for the gesture recognition purpose they used these features to pass in neural network training. They include glove based and vision based techniques. They build a system that recognises gestures using natural hand gestures.

Artificial neural network (ANN) based method is proposed which automatically recognizes the fingerspelling in Sign language. Their method uses artificial neural networks and digital image processing techniques for recognizing different signs [7]. They used four major steps: Image Acquisition and Pre-processing, Hand Segmentation, Feature Extraction and Classification. For extracting the hand region from the image they used skin colour based segmentation. Their method had low computational complexity.

Hand Gesture Technology used for Sign Language Recognition [8]. In this paper they have implemented a vision based method which includes approach based on appearance and approach based on a 3-D model. An approach based on an instrumented glove is used in which a sensor attached with the glove captures the hand movement and position. An approach based on colored marker is applied. For tracking the movement, position and motion of the hand colour of the glove is used.

Vision based human hand tracking is proposed by Rehag, James M., and Takeo Kanade [9]. In this to decompose global image patterns into incremental, local motions of simple shapes they used kinematic and geometric hand models, along with a high temporal sampling rate. Two hand tracking systems based on reduced and full-state hand models, using one and two cameras are developed.

Khan and Ibraheem discussed various methods including neural network, hidden markov model, colour space extraction method, fuzzy c means clustering etc [10]. In gesture recognition the image is segmented then gesture recognition and features are extracted from these gestures. They use Artificial neural networks as a classifier in the gesture recognition process. Applications where hand gesture recognition are used are recognition of sign language, control of the robot and control of the graphic editor. They are also used in virtual environments, recognition of numbers, television control and 3D modelling.

P. V. V. Kishore and P. Rajesh Kumar [11] proposed a video based Sign language recognition system that has four processing stages i.e video preprocessing, image segmentation, feature extraction and pattern classification. For detecting the shape of various hand signs a wavelet based video segmentation technique is used.

A static hand gesture recognition algorithm using k-mean [12] is proposed by Dipak Kumar Ghosh, Samit Ari to obtain a rotation invariant gesture image which coincides the 1st principal component of the segmented hand gestures with vertical axes. For distinguishing the static hand gestures from one class to another the shape of contour property is used. K-mean based radial basis function neural network is used for doing classification.

2.3 EXISTING SYSTEM

Majority of research work has been done on American Sign Language but very limited has been on Sign language. For persons who are deaf or dumb, sign language is crucial as it is the only means of communication for such people, therefore it is important that others grasp their language. This study suggests a method or algorithm for an application that will aid in the recognition of Sign language's various signs. There are currently two types of gesture recognition techniques used in ASL: vision-based and non vision-based. For sign identification, vision-based approaches use a web camera and markers. The signers' or users' gestures must be identified and localised in image frames for gesture identification. Non-vision technique includes sensors mainly and other electronic hardware i.e glove based for gesture recognition.

2.4 PROPOSED SYSTEM

This project focuses on bridging the gap between Normal people and the deaf, blind and mute community. This system will eliminate the need of an interpreter. This model is developed with the help of ANN technique which used CNN algorithm.

CHAPTER 3 - PROBLEM DESCRIPTION

3.1 PROBLEM DESCRIPTION

Sign language has been used since ages. Even though sign language is the most natural and oldest form of language used for communication, the majority of the people are not aware of the same and interpreters are very difficult to come by. A neural network based model will be developed for the easy conversion of Sign language to Text.

3.2 SPECIFICATION

Machine learning is a booming industry. The prime objective of machine learning is easing human effort and bringing comfort to human life. Machine learning helps in studying the huge data and implementing it in some day to day affairs.one such implementation of machine learning is image recognition. In this project image recognition technique is used to predict sign language symbols which are passed as input.

Sign language to speech translator is a task that involves computer vision, machine learning and deep learning concepts to recognise the context of a hand gesture from the camera and describe them in a natural language like English.

The aim of sign language to text translator is to automatically recognise and describe the hand gesture with one or more natural language sentences. This is a problem that integrates computer vision, machine learning and deep learning concepts. CNN algorithm is used for the training purpose as it is most reliable for the case of image recognition[11] [12].

3.3 PROCESS FLOW

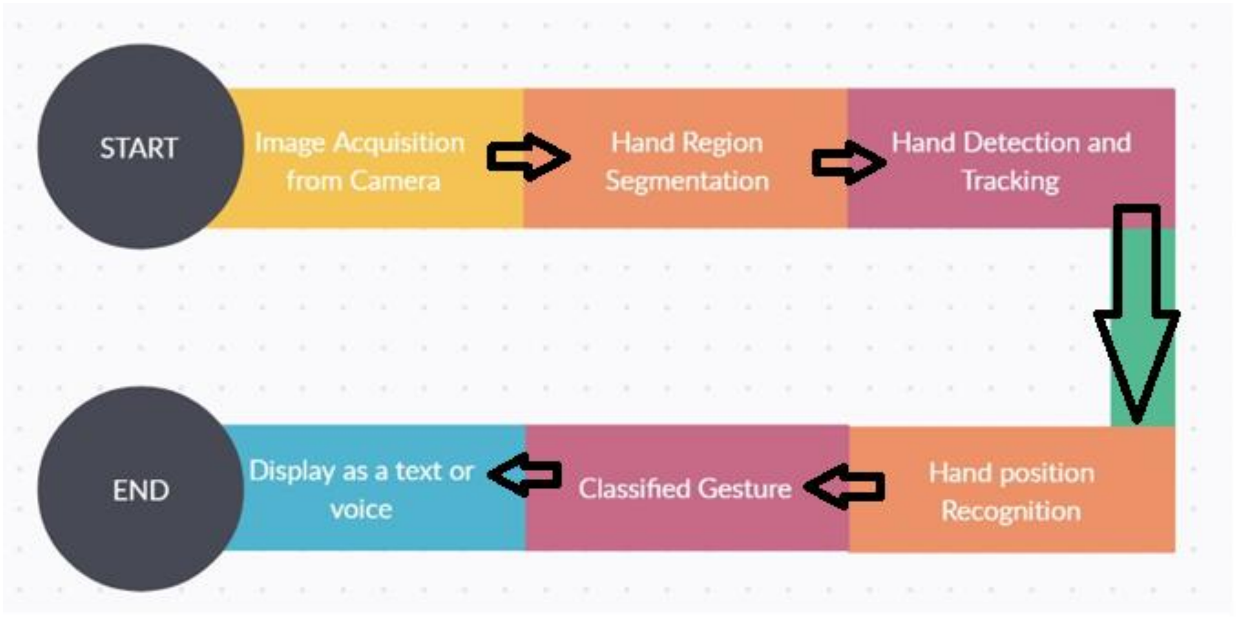

START
Image Acquisition from Camera
Hand Region Segmentation
Hand Detection and Tracking
Hand position Recognition
Classified Gesture
Display as a text or voice
END

Figure 2: Flow Diagram of sign language translation process

3.4 FUNCTIONAL REQUIREMENTS

3.4.1 Software Requirements

- **Python**
- **IDE :**
  - JupyterLab or Visual Studio code
- **Python Libraries:**
  - Tensorflow
  - Keras:
  - Numpy
  - OpenCV
  - gTTS
  - sklearn
  - Matplotlib
  - ModelCheckpoint

3.4.2 Hardware Requirements

- **Processor:**
  - Intel i3+ or AMD A8+
- **RAM:**
  - 4GB+
- **Hard Disk Space:**
  - 20GB+
- **GPU:**
  - 2GB+

3.5 NON-FUNCTIONAL REQUIREMENTS

- Dataset containing 26 classes with each having around 200 images.
- For training the model - Jupyter Notebook, Kaggle Notebook or Google Colab should be preferred

CHAPTER-4 SYSTEM DESIGN

4.1 OVERVIEW

This project aims to provide ease in the communication for the deaf, blind and mute community. It translates the actions into the character and with the help of characters it develops words simultaneously and further adds them to the sentence. A convolutional neural network is used for training the model after analyzing many other algorithms. Gaussian filter is used for image input and transformation.

4.2 INTRODUCTION

Sign language is one of the oldest and most natural forms of language for communication, but since most people do not know sign language and interpreters are very difficult to come by. Thus, a real time method is proposed using neural networks for fingerspelling based on American Sign language.

In this method, the image of hand is first passed through a gaussian filter and after the filtering the image then it passes through a classifier created through CNN which predicts the class of the hand gestures.

4.3 CONVOLUTIONAL NEURAL NETWORK

In convolution neural networks the neurons are arranged in 3 dimensions: width, height and depth. The neurons in the layer will only be connected to a small region of the layer i.e the window size before it, instead of all of the neurons in a fully connected manner. Final output layer would have dimensions, because by the end of the CNN architecture the full image will be reduced into a single vector of class scores[7].

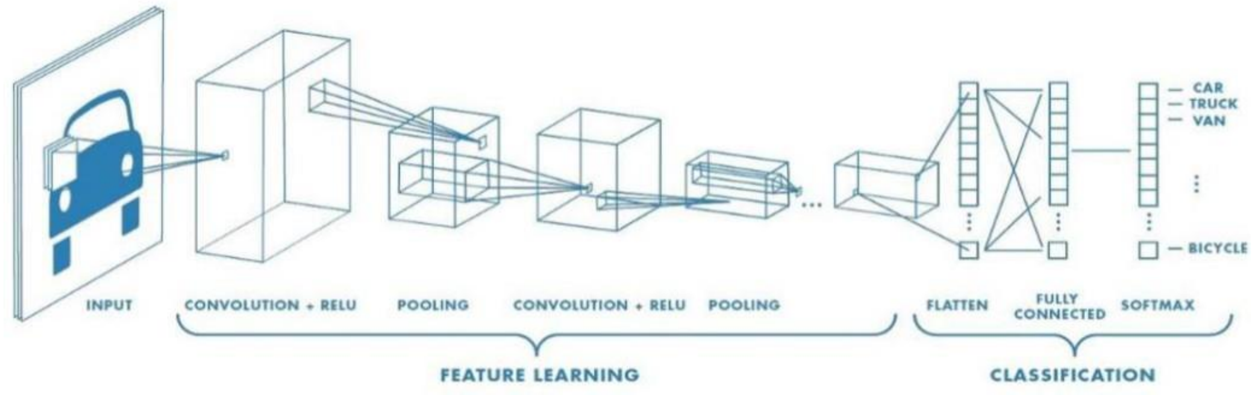


Figure 3: CNN working

4.3.1 Convolution layer

In the convolution layer a small window size is taken that will extend to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration the window is slided by the stride size and computes the dot product of the filter entries and input values at a given position. Networks 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 colour[4] [7] .

4.3.2 Pooling layer

Pooling layer is used to decrease the size of the activation matrix and reduce the learnable parameters. There are two types of pooling:

1. Max Pooling
2. Average Pooling

4.3.3 Fully connected Layer

In convolution layers neurons are connected only to a local region while in a fully connected region well connect all the inputs of neurons.

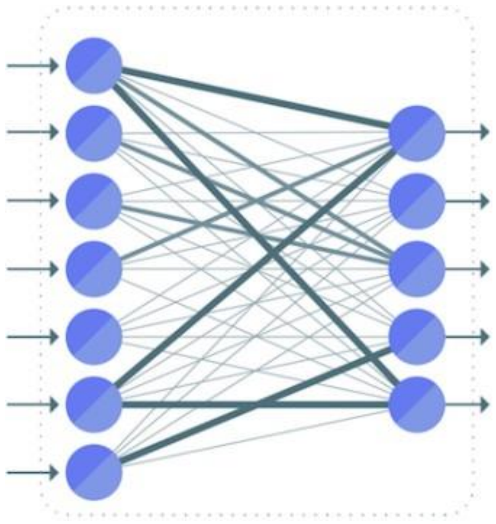


Figure 4: Fully connected layer diagram

4.3.4 Final Output Layer

After getting values from a fully connected layer, then will connect them to the final layer of neurons (having count equal to total number of classes), that will predict the probability of each image in different classes.

4.3 PROBLEMS FACED

1. At the initial period of this project SURF detection was used but recently it got patented and was not available for open-source usage. Therefore, its alternatives were searched so that accuracy is not compromised.
2. Another issue was to select a filter which could be applied on the images so that proper features of the images could be obtained and hence that image as input is provided for CNN model. Various filters including binary threshold, canny edge detection, gaussian blur etc. were studied, but finally implemented a project with a gaussian blur filter.
3. Minor issues related to accuracy occurred which improved eventually after training

the model and making slight changes to the dataset.

4.5 FLOWCHART

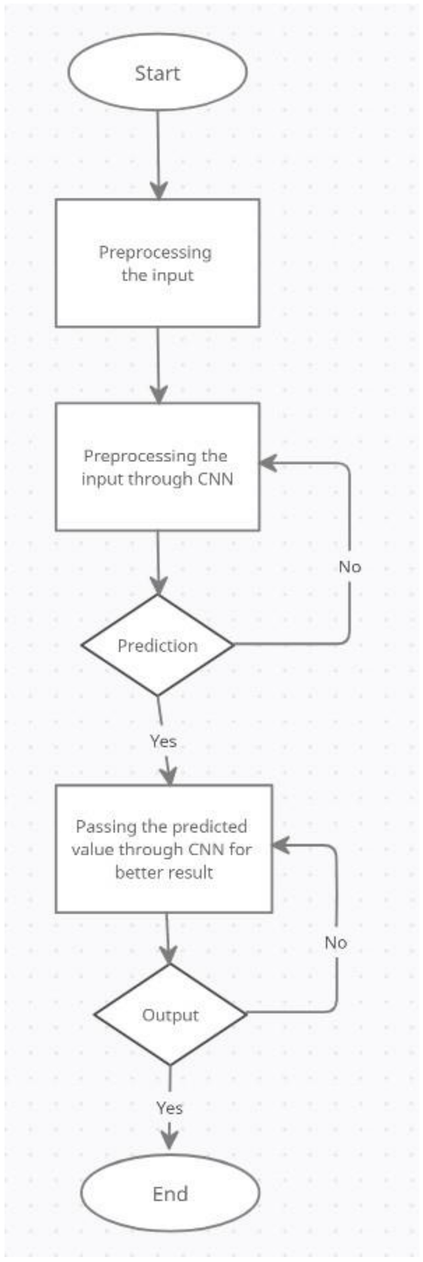

Start
Preprocessing the input
Preprocessing the input through CNN
No
Prediction
Yes
Passing the predicted value through CNN for better result
No
Output
Yes
End

Figure 5: Flowchart

CHAPTER-5 RESULT AND IMPLEMENTATION

5.1 OVERVIEW

In this chapter, implementation of the project is presented. All the steps taken to implement the project are also covered. Moving further user interface of the Application and various details of the project is shown. Lastly, testing of this project is discussed.

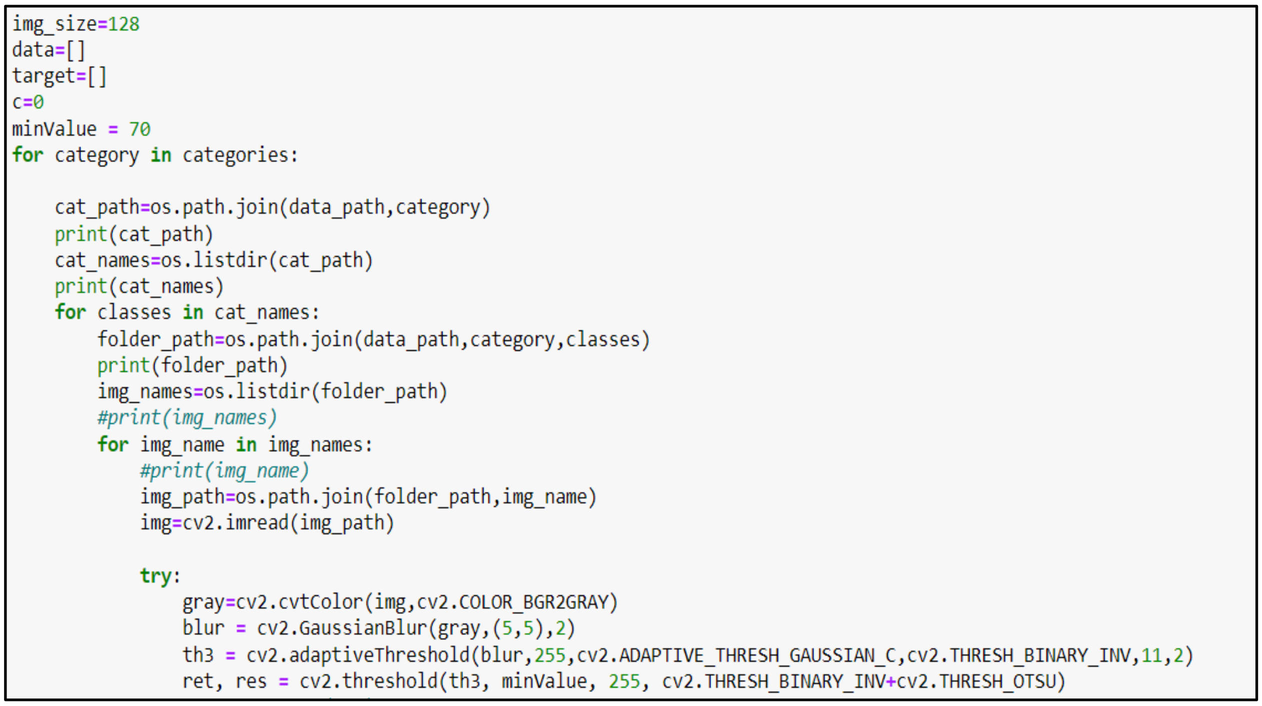
5.2 PREPROCESSING

5.2.1 Dividing Dataset into Train and Test

The dataset consists of 17,113 images divided into 27 classes. There are 26 classes from alphabets A-Z and class 0 for representing null. The division ration of the dataset is 75:25 where 75 is training data and 25 is testing data. At the same time, the code is converting the normal images (RGB) captured from the webcam. These images are then converted into black and white and then further converted into processed images developed through gaussian filter.

5.2.2 Image Preprocessing

The input images are passed through the function *"func"* with the image path as its argument. First the image is converted into grayscale from the RGB input image using cv2.cvtColorfunction. Then the grayscale image is passed through a gaussian blur filter to obtain a processed image for training the CNN model. Adaptive threshold function also used to separate the hand from the background.



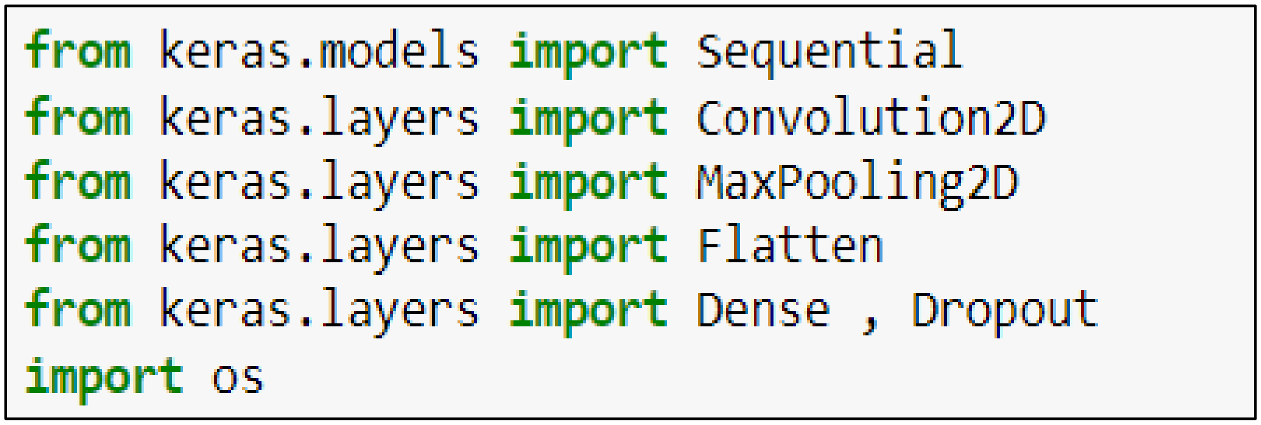
```
img_size=128
data=[]
target=[]
c=0
minValue = 70
for category in categories:

    cat_path=os.path.join(data_path,category)
    print(cat_path)
    cat_names=os.listdir(cat_path)
    print(cat_names)
    for classes in cat_names:
        folder_path=os.path.join(data_path,category,classes)
        print(folder_path)
        img_names=os.listdir(folder_path)
        #print(img_names)
        for img_name in img_names:
            #print(img_name)
            img_path=os.path.join(folder_path,img_name)
            img=cv2.imread(img_path)

            try:
                gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                blur = cv2.GaussianBlur(gray,(5,5),2)
                th3 = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
                ret, res = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

Figure 6: Image Preprocessing function

5.3 TRAINING THE DATASET

5.3.1 Importing Python Libraries For Building Model

Keras is an open-source library written in python to build deep learning models. Here the sequential model is used for the proposed application. Convolution2D, Maxpooling2D, Flatten, Dense and Dropout as the layers for building the CNN model.



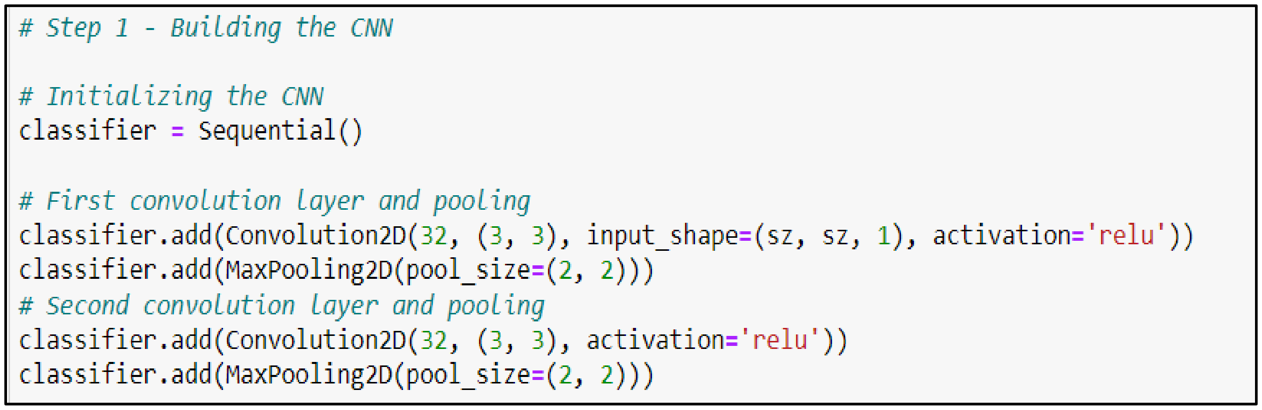
```python
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense , Dropout
import os
```

Figure 7: Keras imports

5.3.2 Building CNN Classifier

For the first convolution layer and pooling 32 filters were used for the first layer and the kernel size was 3 x 3. The activation function for the first layer was Relu. The pool size for max pool layer is 2 x 2.

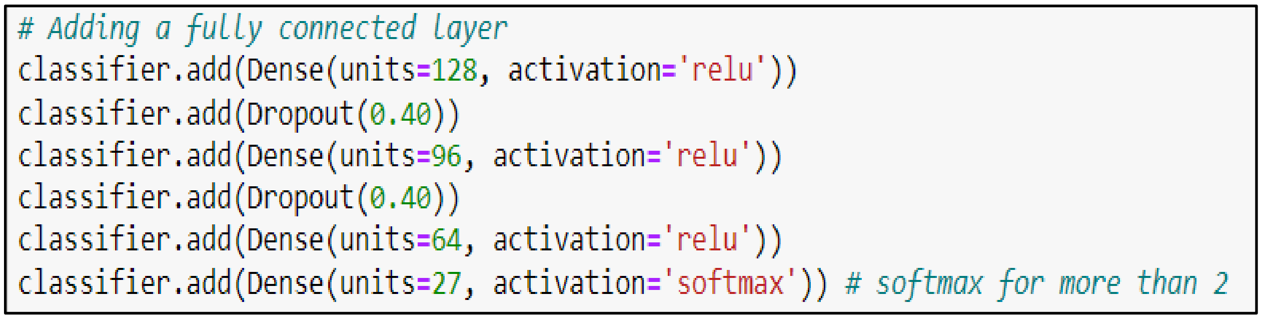
For the second convolution layer and pooling 32 filters were used for the first layer and the kernel size was 3 x 3. The activation function for the first layer was Relu. The pool size for max pool layer is 2 x 2.



```python
# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
```

Figure 8: Building the CNN model with layers 1 and 2

5.3.3 Flattening the Layers and Adding a Fully ConnectedLayer

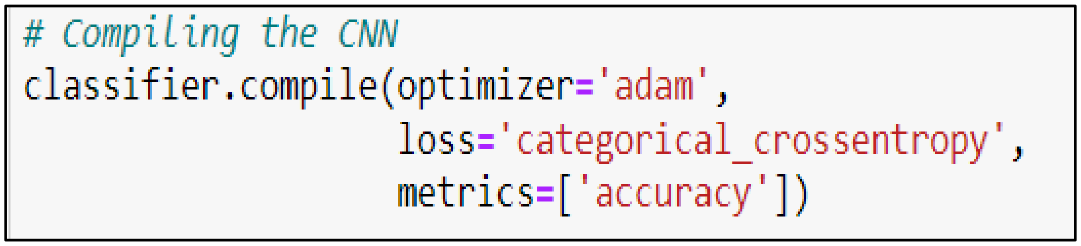


```
# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2
```

Figure 9: Fully connected layer code

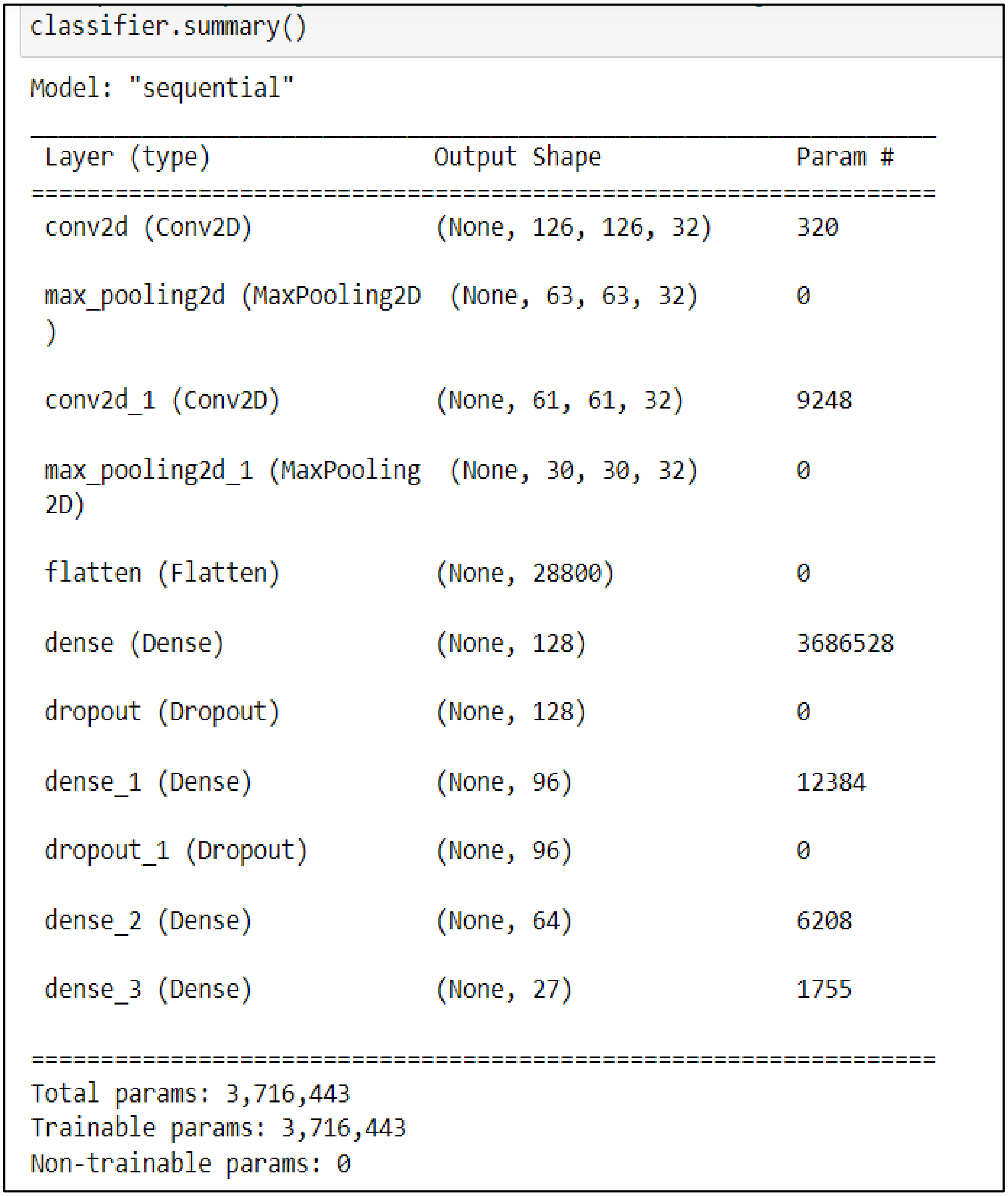
5.3.4 Compiling the CNN and Building a Classifier

The optimiser used for this classifier is adam. The loss function used for this model is categorical crossentropy and accuracy matrix is used.



```
# Compiling the CNN
classifier.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=['accuracy'])
```

Figure 10: Compiling the CNN model

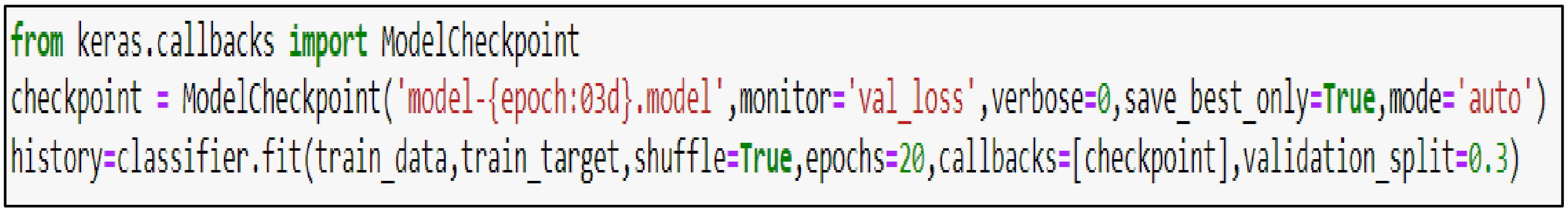


```
classifier.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 61, 61, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 30, 30, 32)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 28800)             0         
                                                                 
 dense (Dense)               (None, 128)               3686528   
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 96)                12384     
                                                                 
 dropout_1 (Dropout)         (None, 96)                0         
                                                                 
 dense_2 (Dense)             (None, 64)                6208      
                                                                 
 dense_3 (Dense)             (None, 27)                1755      
                                                                 
=================================================================
Total params: 3,716,443
Trainable params: 3,716,443
Non-trainable params: 0
```

Figure 11: Classifier summary

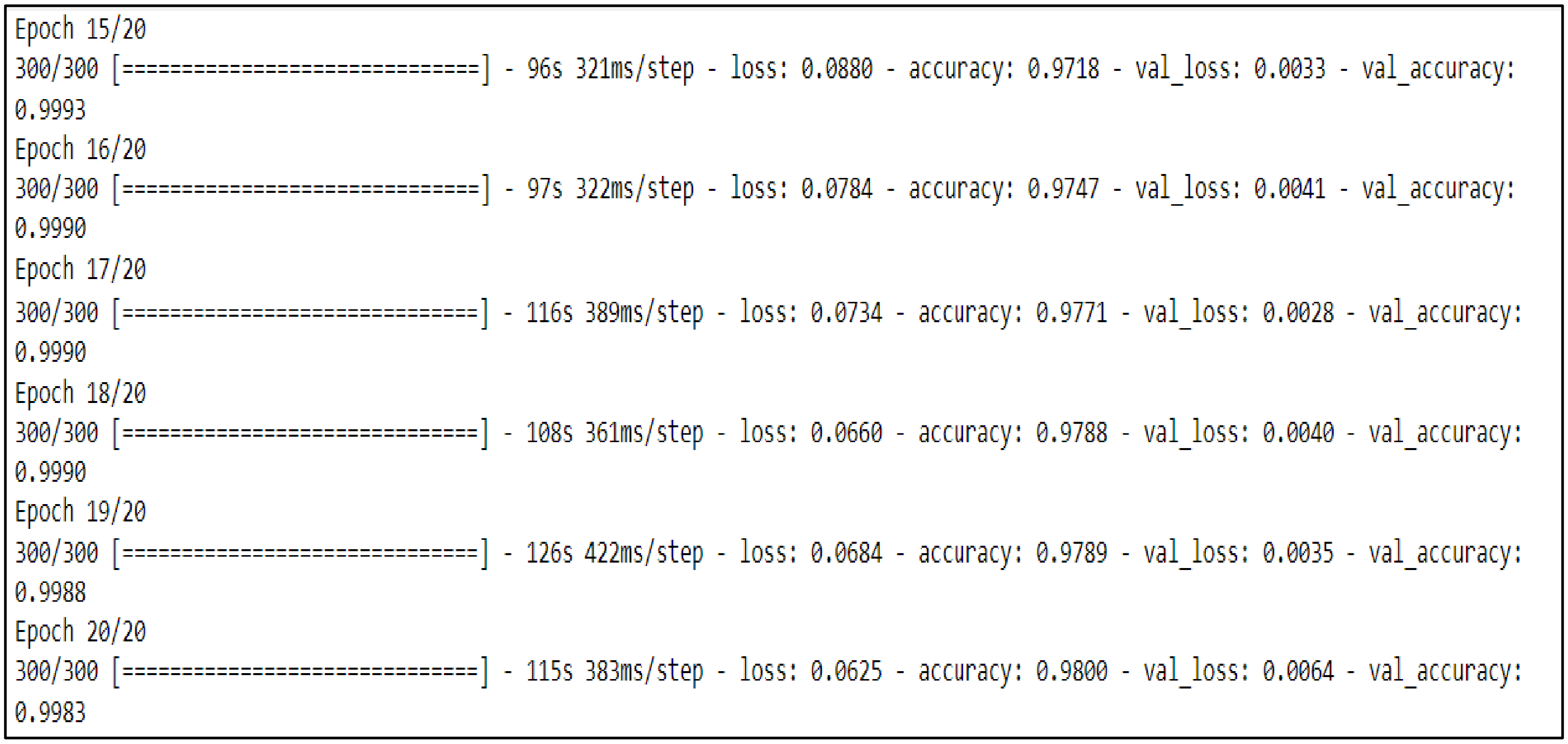
5.3.5 Training The Model

The calculation for the steps per epoch is done and the steps per epoch is calculated by dividing the number of training images by the batch size. 20 epochs are run to train the model.



```
from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
history=classifier.fit(train_data,train_target,shuffle=True,epochs=20,callbacks=[checkpoint],validation_split=0.3)
```

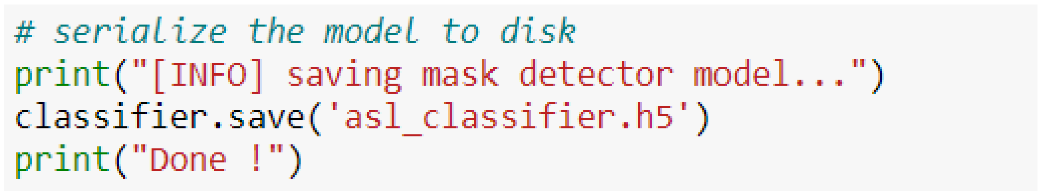
Figure 12: Fitting the model with epoch runs



```
Epoch 15/20
300/300 [==============================] - 96s 321ms/step - loss: 0.0880 - accuracy: 0.9718 - val_loss: 0.0033 - val_accuracy:
0.9993
Epoch 16/20
300/300 [==============================] - 97s 322ms/step - loss: 0.0784 - accuracy: 0.9747 - val_loss: 0.0041 - val_accuracy:
0.9990
Epoch 17/20
300/300 [==============================] - 116s 389ms/step - loss: 0.0734 - accuracy: 0.9771 - val_loss: 0.0028 - val_accuracy:
0.9990
Epoch 18/20
300/300 [==============================] - 108s 361ms/step - loss: 0.0660 - accuracy: 0.9788 - val_loss: 0.0040 - val_accuracy:
0.9990
Epoch 19/20
300/300 [==============================] - 126s 422ms/step - loss: 0.0684 - accuracy: 0.9789 - val_loss: 0.0035 - val_accuracy:
0.9988
Epoch 20/20
300/300 [==============================] - 115s 383ms/step - loss: 0.0625 - accuracy: 0.9800 - val_loss: 0.0064 - val_accuracy:
0.9983
```

Figure 13: Few epoch training outputs

5.3.6 Saving The Model

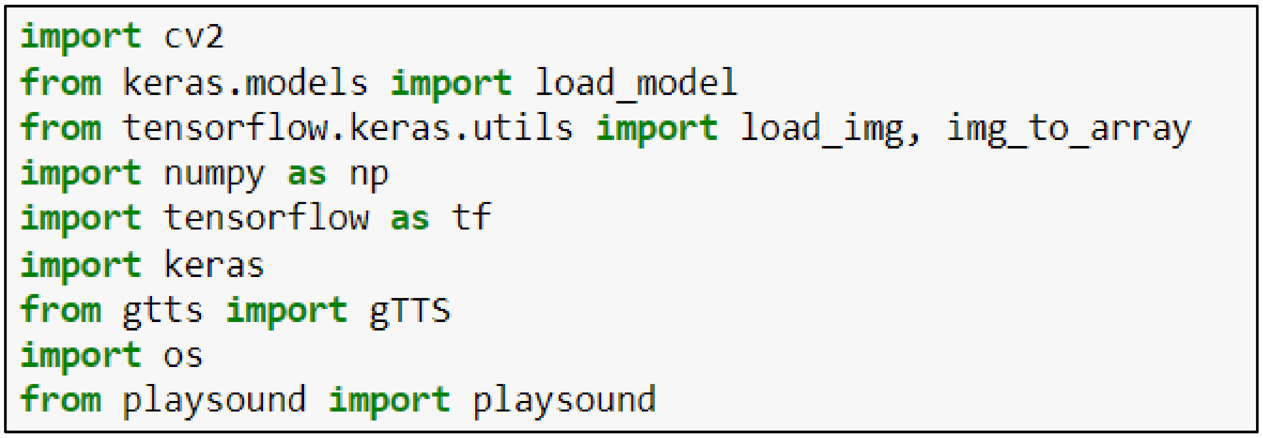


```
# serialize the model to disk
print("[INFO] saving mask detector model...")
classifier.save('asl_classifier.h5')
print("Done !")
```

Figure 14: Saving the trained model using CNN

5.4 CREATING ASL_Real-time.py

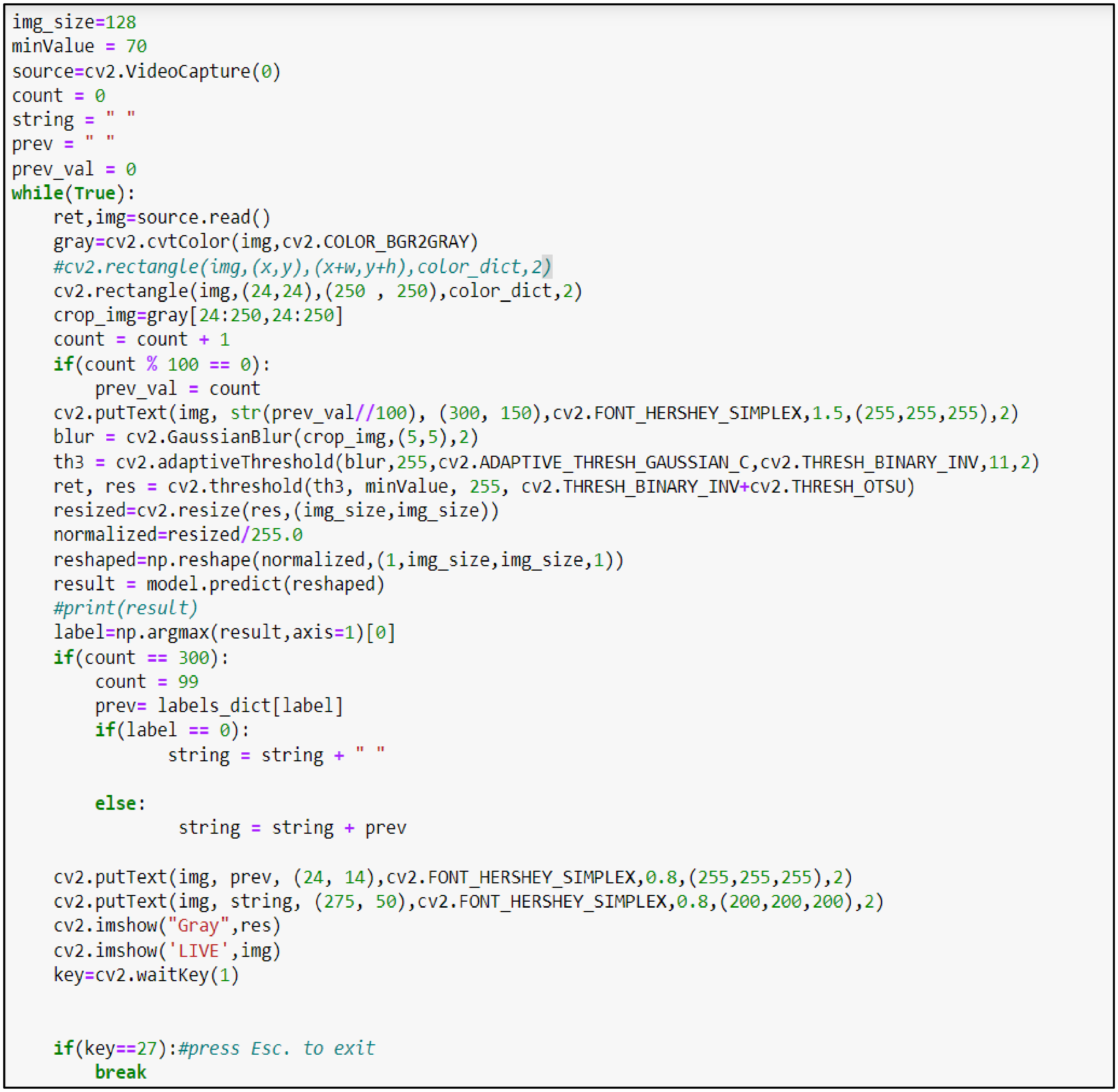
5.4.1 Libraries Used To Create ASL_Real-time.py



```
import cv2
from keras.models import load_model
from tensorflow.keras.utils import load_img, img_to_array
import numpy as np
import tensorflow as tf
import keras
from gtts import gTTS
import os
from playsound import playsound
```

Figure 15: Different Libraries imported for the ASL_Real-time.py

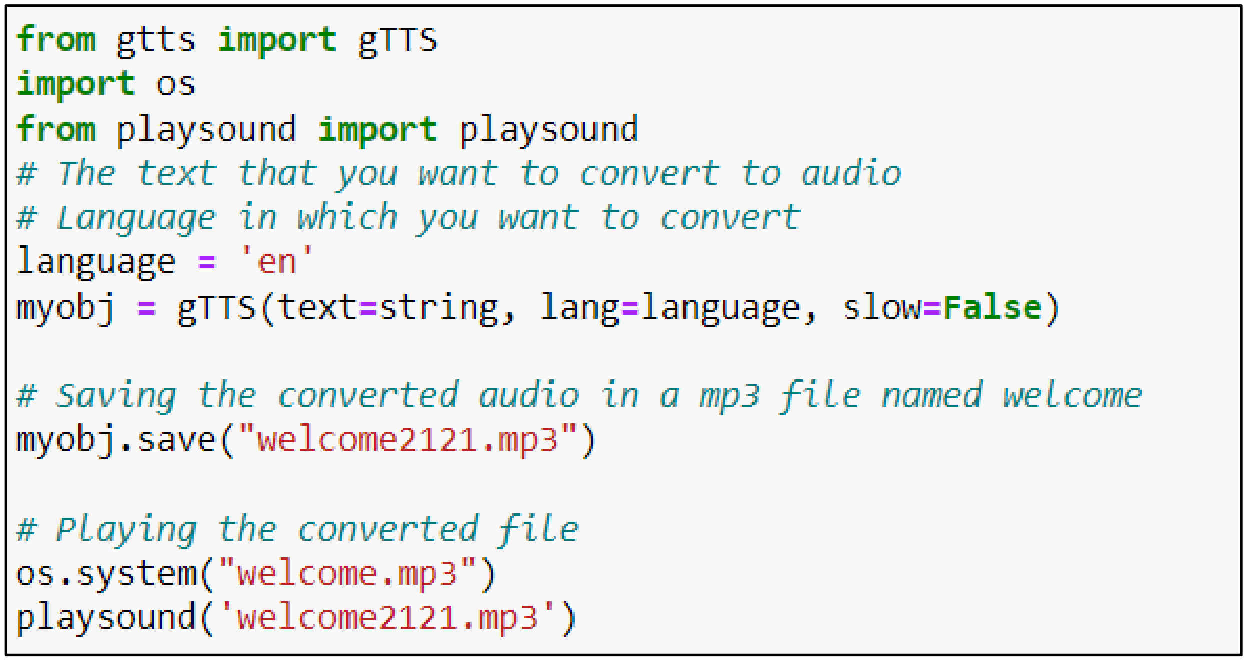
5.4.2 Predicting Characters



```
img_size=128
minValue = 70
source=cv2.VideoCapture(0)
count = 0
string = " "
prev = " "
prev_val = 0
while(True):
    ret,img=source.read()
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    #cv2.rectangle(img,(x,y),(x+w,y+h),color_dict,2)
    cv2.rectangle(img,(24,24),(250 , 250),color_dict,2)
    crop_img=gray[24:250,24:250]
    count = count + 1
    if(count % 100 == 0):
        prev_val = count
    cv2.putText(img, str(prev_val//100), (300, 150),cv2.FONT_HERSHEY_SIMPLEX,1.5,(255,255,255),2)
    blur = cv2.GaussianBlur(crop_img,(5,5),2)
    th3 = cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
    ret, res = cv2.threshold(th3, minValue, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    resized=cv2.resize(res,(img_size,img_size))
    normalized=resized/255.0
    reshaped=np.reshape(normalized,(1,img_size,img_size,1))
    result = model.predict(reshaped)
    #print(result)
    label=np.argmax(result,axis=1)[0]
    if(count == 300):
        count = 99
        prev= labels_dict[label]
        if(label == 0):
               string = string + " "

        else:
                string = string + prev

    cv2.putText(img, prev, (24, 14),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    cv2.putText(img, string, (275, 50),cv2.FONT_HERSHEY_SIMPLEX,0.8,(200,200,200),2)
    cv2.imshow("Gray",res)
    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)


    if(key==27):#press Esc. to exit
        break
```

Figure 16: Prediction function from ASL_Real-time.py

5.5 Text To Speech

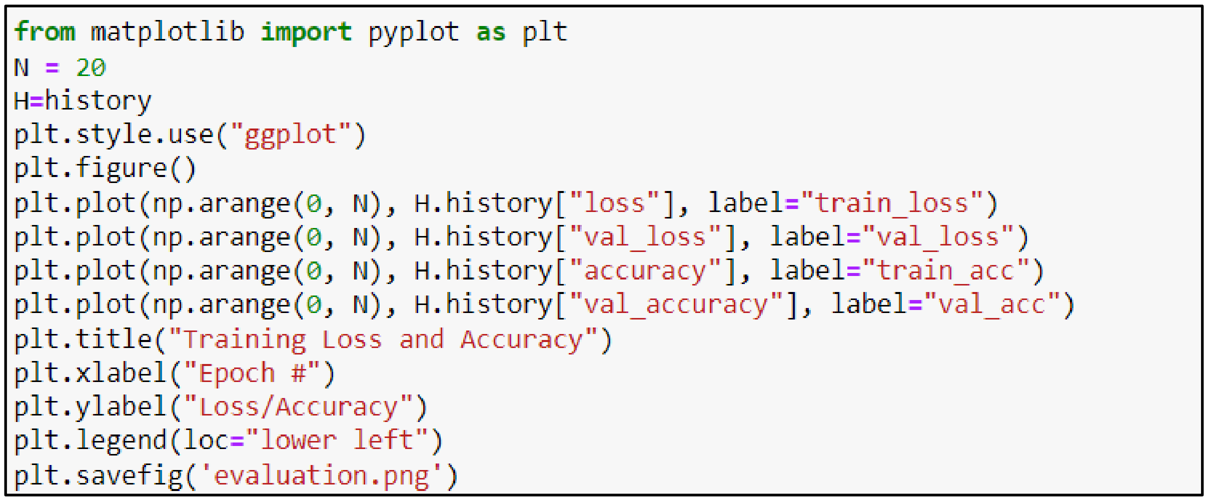


```
from gtts import gTTS
import os
from playsound import playsound
# The text that you want to convert to audio
# Language in which you want to convert
language = 'en'
myobj = gTTS(text=string, lang=language, slow=False)

# Saving the converted audio in a mp3 file named welcome
myobj.save("welcome2121.mp3")

# Playing the converted file
os.system("welcome.mp3")
playsound('welcome2121.mp3')
```

Figure 17: Code to convert text to speech and then play

5.6 RESULTS

5.6.1 Training Loss and Accuracy



```
from matplotlib import pyplot as plt
N = 20
H=history
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig('evaluation.png')
```

Figure 18: Code to plot graph using matplotlib

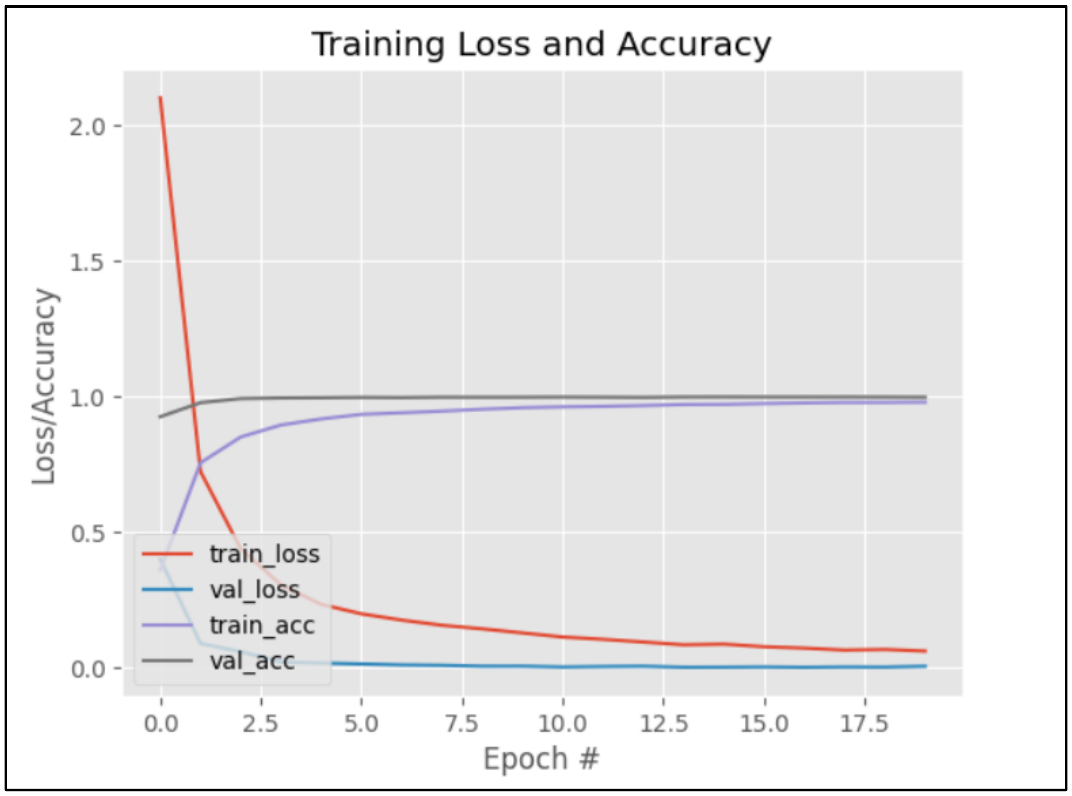


Figure 19: Graph of Training loss/accuracy v/s Epoch

5.6.2 Prediction of Various Alphabets

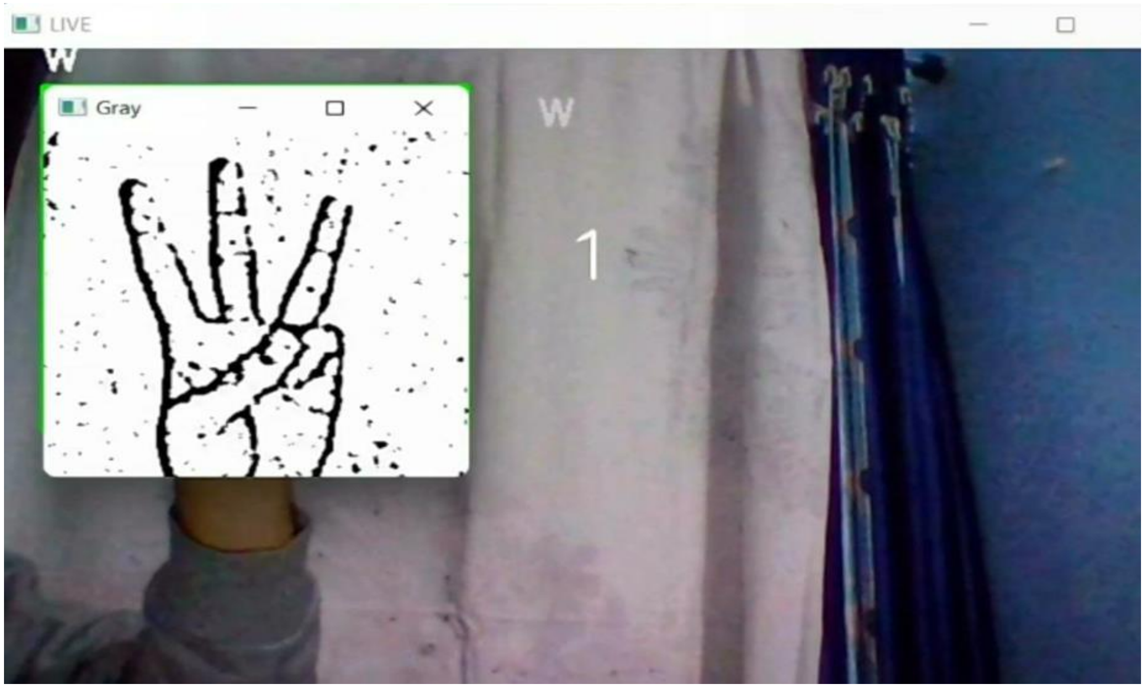


Figure 20: Figure showing W is recognized correctly and represented as English alphabet

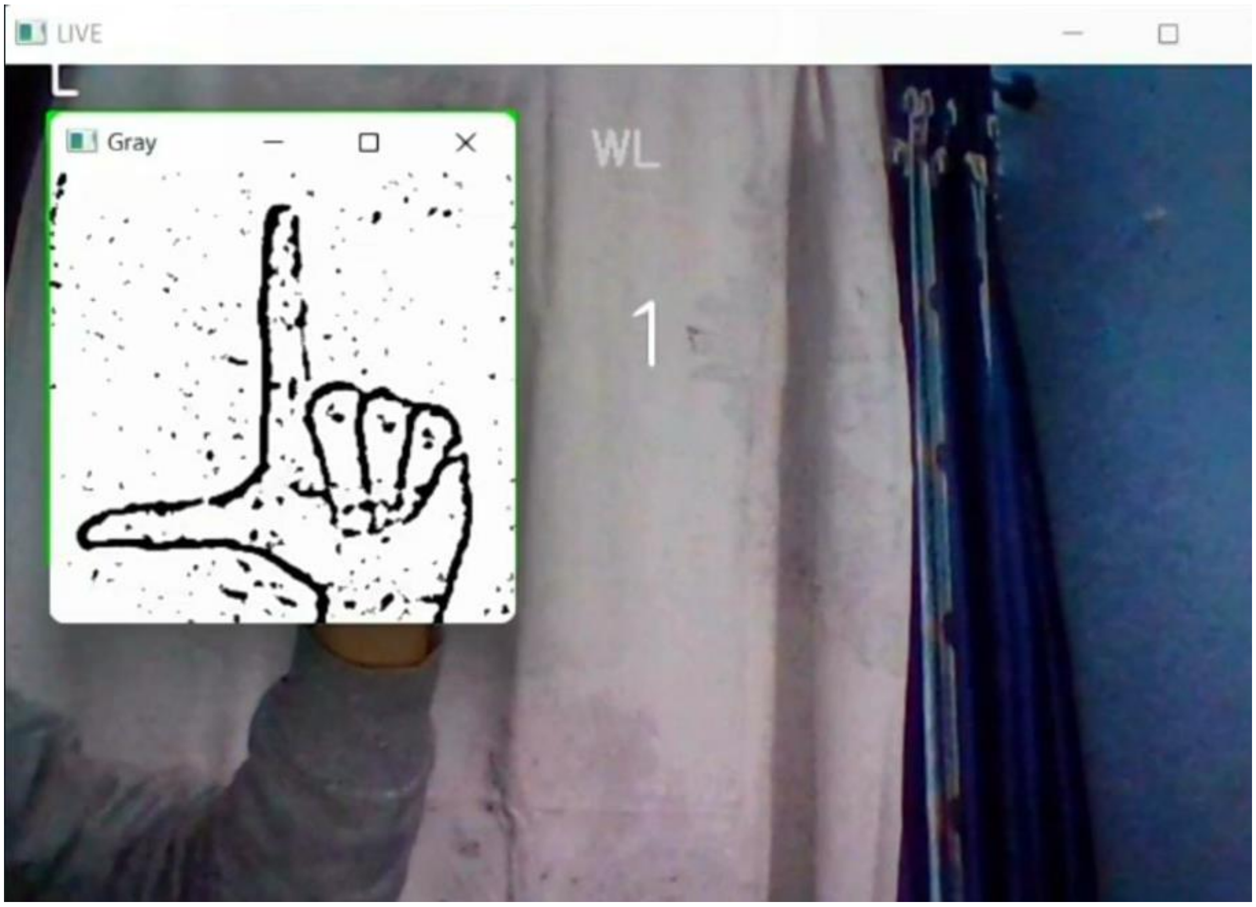


Figure 21: Figure showing L is recognized correctly and represented as English alphabet

CHAPTER-6 CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

- This application allows users to form characters, words, and sentences according to their needs. Moreover, by providing multiple suggestions for the corresponding word being formed further helps in communication.
- The main objective has been achieved, that is, the need for an interpreter has been eliminated.

6.2 FUTURE SCOPE

- In this report, a functional real time vision based American Sign language recognition for Deaf, Blind & Mute people have been developed for ASL alphabets. The prediction is improved after implementing two layers of algorithms in which symbols are predicted and verified which are more similar to each other.

- This way it is able to detect almost all the symbols provided that they are shown properly, there is no noise in the background and lighting is adequate. This project aims to create a computer application and train a model which when shown a real time video of hand gestures of American Sign language shows the output for that particular sign in text format on the screen.
- This project did not focus on Facial Expressions although it is well known that facial expressions convey an important part of sign languages. This system can be implemented in many application areas examples include accessing government websites whereby no video clip for deaf and mute is available or filling out forms online whereby no interpreter may be present to help.
- This technology could mean a new way for about 70 million people with hearing and speech impairment to verbally communicate and connect to people around them. This project aims to lower the communication gap between the deaf, blind or mute community and the normal world.
- In future work, a proposed system can be developed and implemented using Raspberry Pi. The Image Processing part should be improved so that System would be able to communicate in both directions i.e., it should be capable of converting normal language to sign language and vice versa.
- It will try to recognize signs which include motion. Moreover, will focus on converting the sequence of gestures into text i.e., words and sentences and then converting it into speech which can be heard.
- Also, the time taken for recognizing each alphabet is long enough which should be minimized using optimization techniques.

CHAPTER-7 REFERENCES

[1] Jain S., Raja K.V.S (2016). Sign language Character Recognition *Indian Institute of Technology, Kanpur*.

[2] Goyal S., Sharma I. (2013). Sign Language Recognition System for Deaf and Dumb People. *International Journal of Engineering Research Technology*.

[3] Singh J., D.K. (2013). Sign language Using eigen value weighted euclidean distance based classification technique. *International Journal of Advanced Computer Science and Applications.*

[4] Padmavathi . S, Saipreethy.M.S, V. (2013).. Sign language character recognition using neural networks. *IJCA Special Issue on Recent Trends in Pattern Recognition and Image Analysis, RTPRIA.*

[5] Xiaolong Teng, Biani Wu, W. Y., And Liu, C. (2005). A hand gesture recognition system based on local linear embedding. *Academic Press, Inc. United States.*

[6] Neha V. Tavari, P. A. V. D (2014). Sign language recognition based on histograms of oriented gradient. *International Journal of Computer Science and Information Technologies, 3657–3660.*

[7] Adithya V., Vinod P. R., Usha Gopalakrishnan (2013). Artificial Neural Network Based Method for Sign language Recognition. *IEEE Conference on Information and Communication Technologies (ICT), pp. 1080-1085.*

[8] Swapnil Chandel, Mrs. Akanksha Awasthi (April 2016). Hand Gesture Recognition for Sign Language Recognition: A Review. *IJSTE - International Journal of Science Technology & Engineering.*

[9] Rehag, James M., and Takeo Kanade (1993). DigitEyes: Vision-Based Human Hand Tracking. *Technical Report CMU-CS-93-220, School of Computer Science, Carnegie Mellon University.*

[10] Rafiqul Zaman Khan and Noor Adnan Ibraheem (July 2012). Hand Gesture Recognition: A Literature Review. *International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, PP. 161-174.*

[11] Dipak Kumar Ghosh , Samit Ari (2011). A Static Hand Gesture Recognition Algorithm Using K-Mean Based Radial Basis Function Neural Network. *IEEE, 8th International Conference on Information, Communications & Signal Processing.*

[12] P. V. V. Kishore and P. Rajesh Kumar (October 2012). A Video Based Sign language Recognition System (INSLR) Using Wavelet Transform and Fuzzy Logic. *IACSIT International Journal of Engineering and Technology, Vol. 4, No. 5.*

[13] Vasiliki E. Kosmidou, and Leontios J. Hadjileontiadis (Dec 2009). Sign Language Recognition Using Intrinsic-Mode Sample Entropy on sEMG and Accelerometer Data. *IEEE transactions on biomedical engineering, Vol. 56, No. 12.*

[14] Pragati Garg, Naveen Aggarwal and Sanjeev Sofat (2009). Vision Based Hand Gesture Recognition. *World Academy of Science, Engineering and Technology 49, pp.972-977.*

[15] Mokhtar M. Hasan, Pramoud K. Misra, (2011). Brightness Factor Matching For Gesture Recognition System Using Scaled Normalization. *International Journal of Computer Science & Information Technology (IJCSIT), Vol. 3(2).*

APPENDIX-1 KEY TERMS WITH DEFINITIONS

1.) OpenCV:

Figure 22: OpenCV Logo

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 Licence.

2.) Keras:

Figure 23: Keras logo

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. Keras is used for creating deep models which can be productized on smartphones. Keras is also used for distributed training of deep learning models. Keras is used by companies such as Netflix, Yelp, Uber, etc.

3.) Tensor Flow:

TensorFlow

Figure 24: Tensorflow logo

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

5.) Gaussian Filter:

A Gaussian Filter is a low pass filter used for reducing noise (high frequency components) and blurring regions of an image. The filter is implemented as an Odd sized Symmetric Kernel (DIP version of a Matrix) which is passed through each pixel of the Region of Interest to get the desired effect. A Gaussian Filter could be considered as an approximation of the Gaussian Function (mathematics).

6.) Adaptive Threshold:

In Simple Thresholding, a global value of threshold was used which remained constant throughout. So, a constant threshold value won’t help in the case of variable lighting conditions in different areas. Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting.