A
Minor Project Report on
“AMERICAN SIGN LANGUAGE INTERPRETER”

In partial fulfillment of requirements for the degree of

Bachelor of Technology (B. Tech.)

in

Computer Science and Engineering

Submitted by

Ms. Aparajita Halder (170396)

Under the Guidance of

Ms. Sonal Shukla

B.Tech CSE (CC)

SCHOOL OF ENGINEERING AND TECHNOLOGY

Mody University and Science and Technology
Lakshmangarh, Distt. Sikar-332311

December 2020

A C K N O W L E D G E M E N T

I would like to express my special thanks of gratitude to Miss Sonal Shukla Mam, who gave me the golden opportunity to do this wonderful project on the topic (American Sign Language Interpreter), which also enabled me in doing a lot of Research and helped me to explore so many new things.

I would like to express my gratitude towards Dean SET, Dr. Senthil (HOD) and all the faculty members of School of Engineering and Technology, who have always been a great help and inspiration for me. They motivated me in my journey and helped me achieve what I desire for.

Lastly, I would also like to thank my parents and friends who supported me a lot in finishing this project within the limited time frame.

Aparajita Halder (170396)

CERTIFICATE

This is to certify that the minor project report entitled " American Sign Language Interpreter " submitted by Ms. Aparajita Halder, as a partial fulfillment for the requirement of B. Tech. VII Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of Ms. Sonal Shukla has undergone the requisite duration as prescribed by the institution for the project work.

PROJECT GUIDE:

Approval Code: AUT_20_CSE_F24_03

Name:Miss Sonal Shukla

ABSTRACT

The American Sign Language (ASL) is a standard communication methodology, which is propagated widely by schools of the deaf & dumb community for an ease of communication. There is a need to help people communicate with each other in a more accessible and intuitive way. Hence, with the help of technology like neural network (CNN) and computer vision (opencv), this project bridges that gap. It is a camera- based sign language recognition system which is highly useful for the deaf & dumb people, by converting sign language gestures to text data.

This software is acting as a mediator between two people who want to communicate with each other in American Sign Language. It interprets the signs made by different hand gestures and also forms accurate sentences in live video feed using the trained model. The system is fast enough to catch up with communication in real time and provides a decent accuracy at the same time. It is also be flexible enough to provide sentence formation feature as well features like edit, remove etc. to minimize human error.

Table of Contents

Chapter1: Introduction

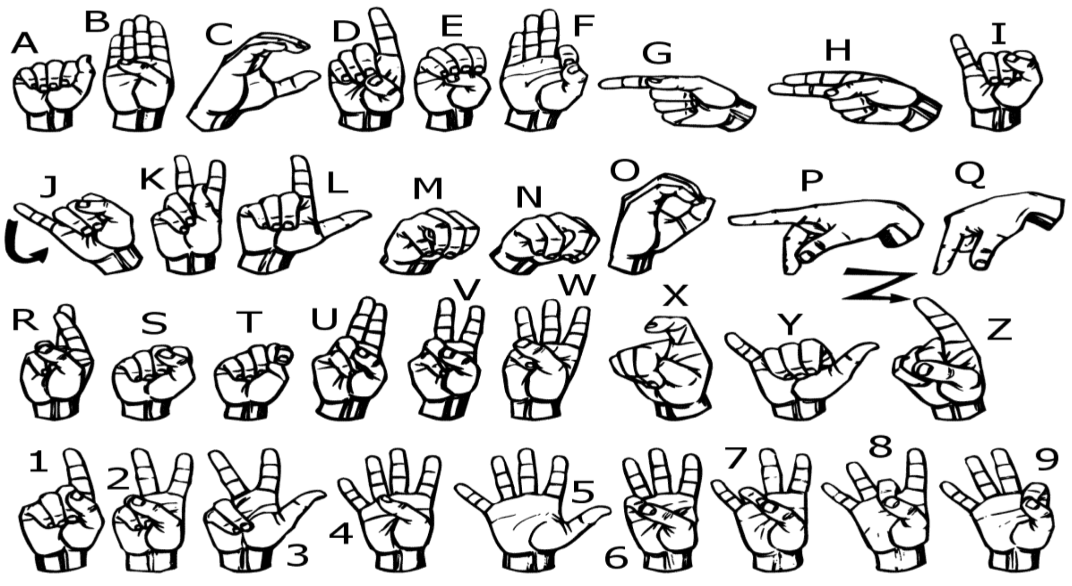
INTRODUCTION

American Sign Language

ASL also known as American Sign Language originated in the early 19th century in the American School for the Deaf (ASD) in West Hartford. Since then, ASL use has propagated widely by schools for the deaf & dumb community for an ease of communication.

There was need of a technology which can make the sign language interpreting process easier. One of the earliest attempts in creating the system was by Mr. Pugeault and Mr. Bowden in 2011. They developed a classifier for 24 letters of American Sign Language.

Since, it is a standard collection of hand gestures which represents several characters and Alphabets. Hence it has gained popularity and systems are being devised in this area to improve the communication process with the help of technology.



1.1 PRESENT SYSTEM

This project aims to bridge the gap between the deaf and dumb people. The basic idea of this project is to make a system using which these community people can use to significantly communicate with all other people using their normal hand gestures. The system requires the background to be perfectly white. The project uses trained CNN model to identify, standard English alphabetic sign language used by the deaf people to communicate and converts them into text so that normal people can understand.

Using technologies like opencv and convolutional neural network (deep learning), which is a subset of Artificial Intelligence a system is devised such that when a hand gesture is made in live camera, it predicts the accurate sign language or hand gesture according to standard American sign language format. It also ensure that the system ignores the minimal hand gestures error as every person has different hand size and may slightly make different hand gestures. Hence, angle, shape, size are some of the features that are taken care of. For that the model is trained in a diversified dataset, which is self created and contain appropriate number of images per gesture for a proper functioning of the system. Data of more than one person is taken to maintain diversity and each gesture is having 750 train images and 250 set of test images. The American sign language interpreter predicts 26 different sign languages with a good accuracy and also help in sentence formation, edition and deletion. All this happens on real-time basis. For features like removal and deletion I have added 4 extra gestures in the dataset. The extra gestures are respectively for blank space, one letter removal, entire sentence removal and an additional no gesture dataset. This additional data is for situation when no gesture is made. It is like default sign so that our model will not make random false prediction. This a software which can act as a mediator between two people who want to communicate with each other in American sign language. It interprets the signs made by different hand gestures and also form accurate sentences in live video feed using the software. The system is fast enough to catch up with communication in real time and provides a decent accuracy at the same time. It is also be flexible enough to provide sentence formation feature as well features like edit, remove etc. to minimize human error.

1.2 PROPOSED SYSTEM

Sign Language is a primary means of communication for many deaf and dumb people. They make different hand gestures which represents the signs. But this sign language is only understood by very few people.

There are situations when one person who is unable to speak and his or her only means of communication is the sign language, wants to communicate something to another person who does not recognizes the gestures. To communicate, the other person will either need to learn the hand gesture or make a guess.

Develop a software which can act as a mediator between two people. It should interpret the signs made by different hand gestures and also form accurate sentences in live video feed using the software. The system should be fast enough to catch up with communication in real time and provide a decent accuracy at the same time. It should also be flexible enough to provide sentence formation feature as well features like edit, remove etc. to minimize human error.

Using technologies like deep learning, which is a subset of Artificial Intelligence a system needs to be devised such that when a hand gesture is made in live camera, it should predict the accurate sign language or hand gesture according to standard American sign language format. Also, ensure that the system should ignore the minimal hand gestures error as every person has different hand size and may slightly make different hand gestures. Hence, angle, shape, size are some of the features that needs to be taken care of. So, a diverse dataset is appreciated.

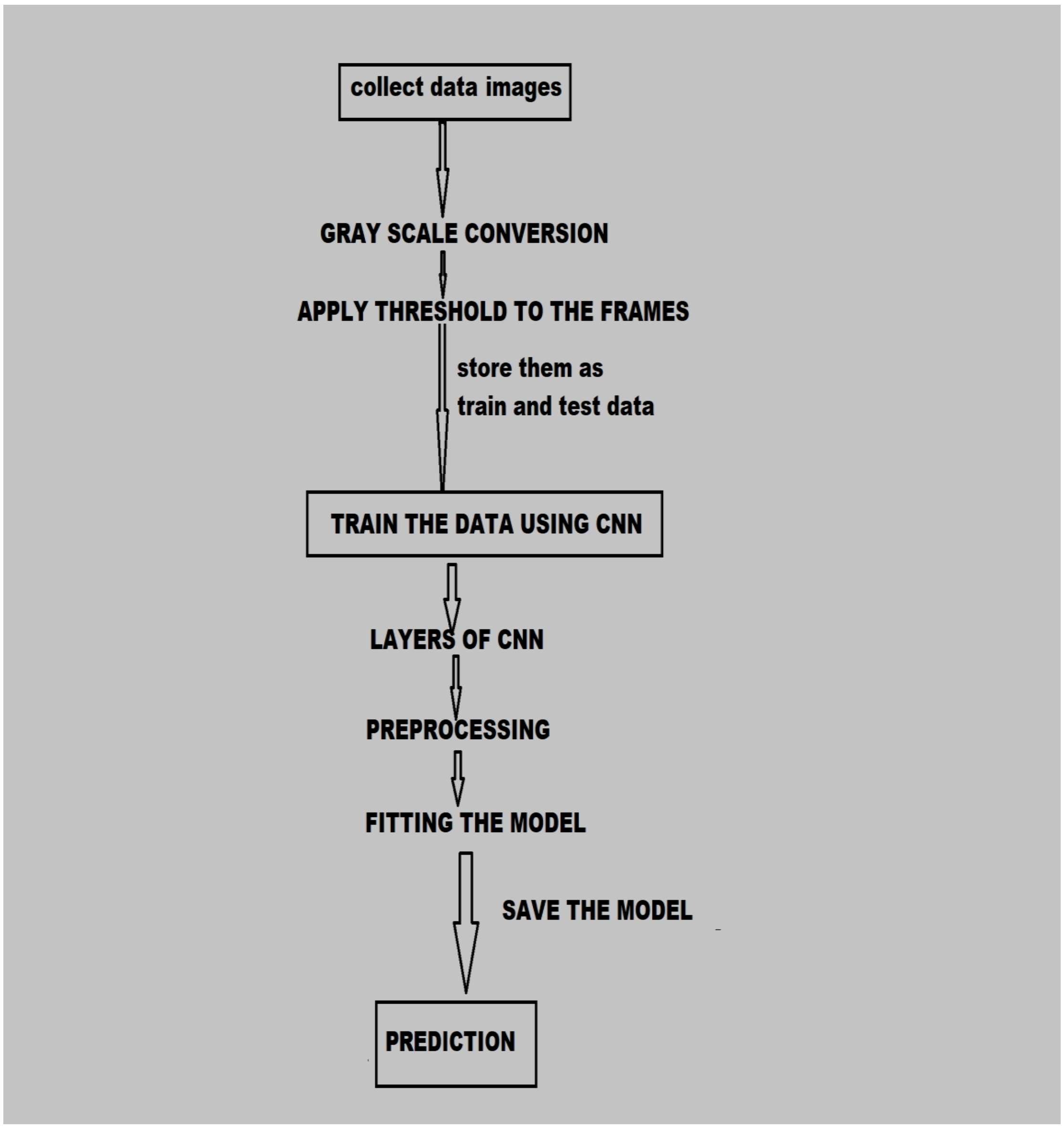
The model should be trained in a diversified dataset, which should be self created and contain appropriate number of images per gesture for a proper functioning of the system. The American sign language interpreter will predict 26 different sign languages with a good accuracy and also help in sentence formation, edition and deletion. All this will happen on real-time basis. For features like removal and deletion add 4 extra gestures in your dataset. The extra gestures will be respectively for blank space, one letter removal, entire sentence removal and an additional no gesture dataset. This additional data is for situation when no gesture is made. It is like default sign so that our model will not make random false prediction.

The project should be a camera- based sign language recognition system which would be in use for the deaf for converting sign language gestures to text data. The objective is to design a solution that is intuitive and simple. Communication for the majority of people is not difficult and an integral part of life. It should be the same way for the deaf and dumb community.

Chapter 2 : System Design

2.1 SYSTEM FLOWCHART

The data is first collected thorough applying threshold and converted as black and white image. Then we train the data using convolutional neural networks. Finally we predict the gestures using the trained data. Steps involved in design and planning of system are –



2.2 SYSTEM DESIGN AND PLANNING

- **Research on algorithm**
  CNN was the most suitable algorithm for this project as it is used for object classification.

- **Set environment**
  Anaconda jupyter notebook was set up and and a separate environment other than base environment was created.

- **Install and import required libraries**
  Installation of required libraries like keras, CNN and OpenCV. Then they were imported in the project.

- **Collect Data**
  User defined dataset was collected for each gesture and stored in the dataset folder.

- **Splitting into train and test data**
  Dataset was split into test and train folder . 750 images was stored in the train folder and 250 in the test folder for each gesture.

- **Pre-Processing of data**
  Pre-processing is required for better accuracy. Resizing of data, gray-scale conversion and threshold was applied to assign pixel values.

- **Train the model**
  Then we created a sequential model and trained our data using CNN. Convolution and max-pooling layer were applied twice for better feature extraction.

- **Prediction**

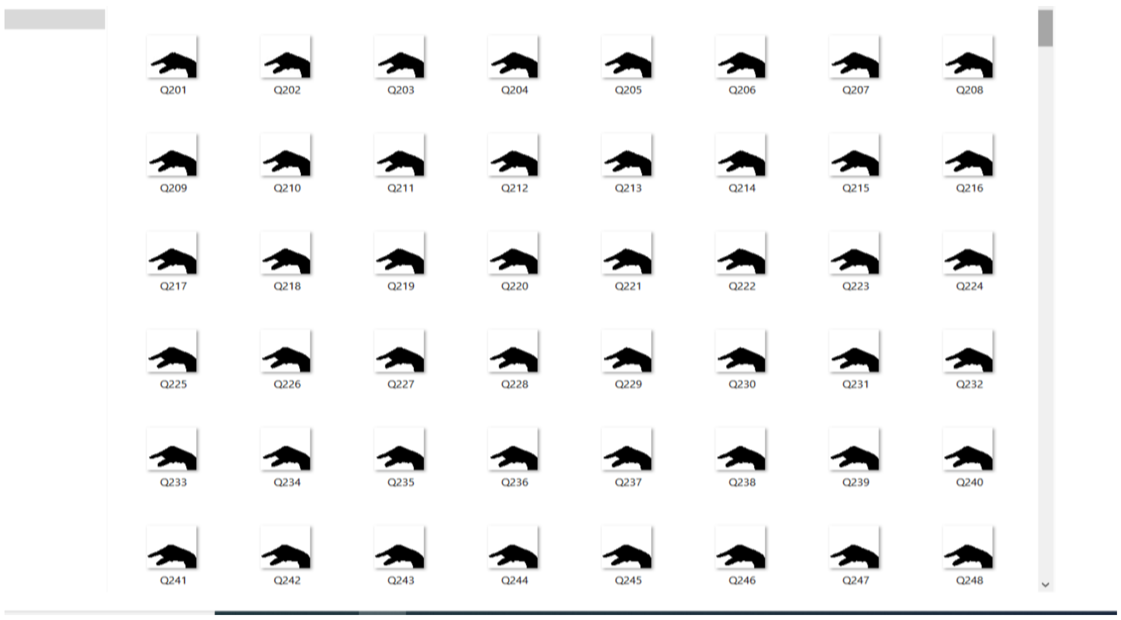
Hand gestures will be captured in live video feed and predict the sign languages in real time.

Dataset

American Sign Language (ASL) is standard language that serves as the communication means of Deaf communities all across the globe. It is a set of 26 hand gestures known as the American manual alphabet, which can be used to spell out words from English.

This is widely popular among deaf and dumb community to communicate using a standard communication language.

We store self created dataset through live video feed and split the data into two parts train and test. The training data for each gesture consist of 750 images and test data contains 250 images.



Convolutional Neural Network

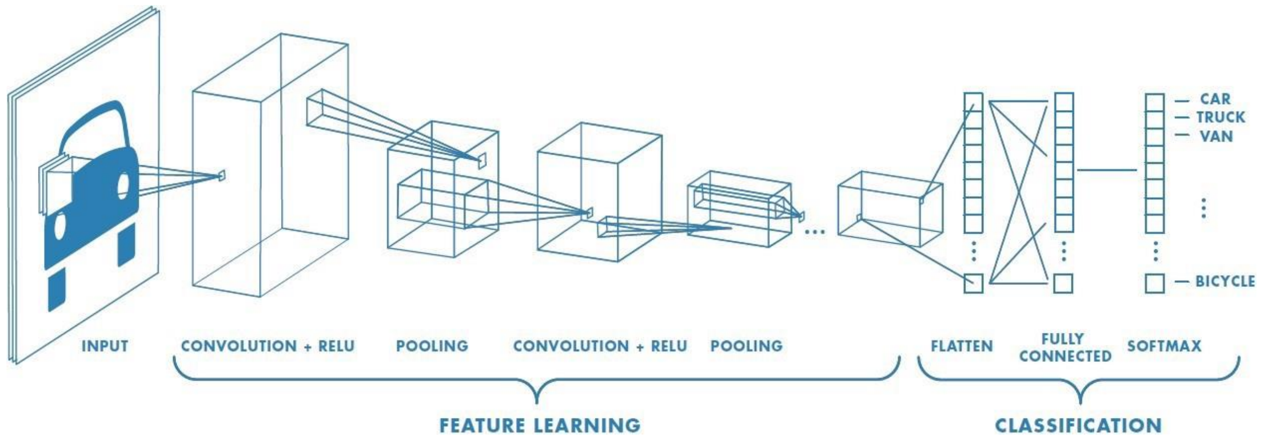
We use CNNs over other NNs as we do not need to flatten the input images to 1D as they are capable of working with image data in 2D. This method helps in retaining the “spatial” properties of images. CNN works best with image like data. If we use ANN then processing of such huge no. of pixels will be computationally expensive.

A Convolutional Neural Network is a Deep Learning algorithm which takes in an input image, assign weights and biases to pixels in the image and extract features from them. This enables it to differentiate one image from the other.

It sequentially builds model layer by layer.

It consists of convolutional layer, max-pooling layer, flattening layer and dense layer.

Activation functions are applied in the layers according to the need of the features.



Layers of CNN:

- **Convolutional Lyer**

  A filter is placed and updated by and weights are initialized randomly. SO we place the filter in top left of the image and it moves throughout the image. We take the average value of pixels and central value is updated. The filter may contain the positive and negative value as the weights are randomly initialized.

- **Normalization Layer**

In this layer we pass an activation function, normally relu so that the negative values becomes zero. As, negative pixel intensities are no use to our system.

- **Pooling Layer**

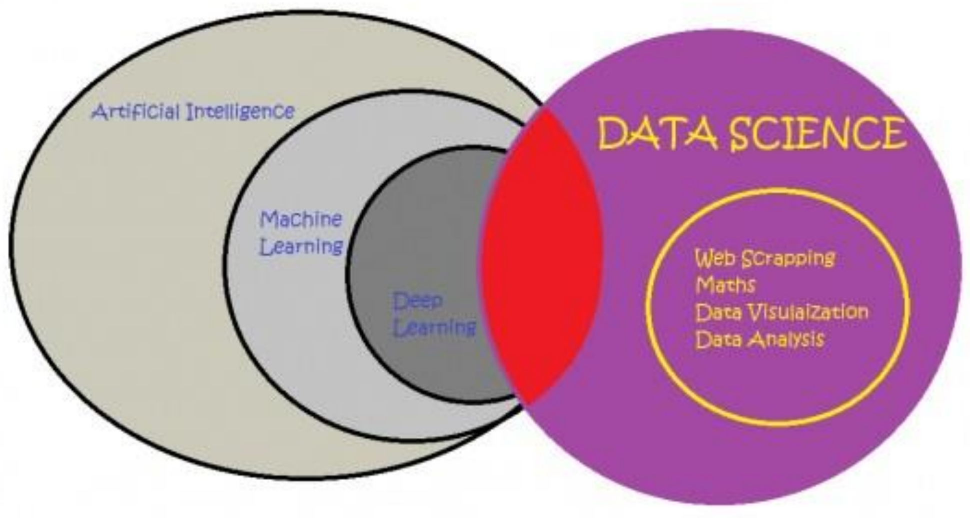
  The pooling layer reduces the no. of pixels as it pools out the maximum value over which the window is placed. It does not repeat pixels unlike convolutional layer.

- **Fully connected Layer**

  Then we flatten the result and feed it to a fully connected layer.

Python

Python is a widely used programming language for the domain of Machine learning and AI. Because of it's easy to use, rich libraries, readability and fast features it is the most used programming language.



Artificial Intellligence and Deep Learning

Artificial Intelligence is a vast field. Deep learning is the subset od AI. Unlike traditional machine learning, we do not have to provide the feature to the algorithm. It automatically extracts features from it. It is best for high computational data. CNN is a type of neural network on which deep learning is based.

Computer Vision

Computer Vision is inspired by human eyes. It makes the process easier and faster as compared to traditional methods. Generally it is preferred for image like data. OpenCV is a computer vision library being used here, which helps in image extracting and processing.

Chapter 3 : Hardware and Software Details

SPECIFICATIONS

Hardware Specification

CPU - Dual Core 2.2 GHz

Processor – Intel Core i3 7th Generation

Ram - 4 GB minimum

OS - 64-bit Windows 10

Software Specification

Language - Python

Tool – Anaconda

IDE - jupyter notebook

Technogies – Artificial Intelligence

Libraries :

- cv2
- numpy
- os
- Sequential from keras.models
- Convolution2D from keras.layers
- MaxPooling2D from keras.layers
- Flatten from keras.layers
- Dense from keras.layers

Software and Hardware Requirements

- The CNN and keras code needs to be written in a programming language, Since, we are using AI for our task. So, programming language such as R and Python is best suitable. In this project, I have implemented in Python as it is having rich library and many more benefits over R.

- Anaconda software was used and jupyter notebook for IDE. We installed the required libraries in Anaconda seperate environment other than base environment.

- A CPU such as i7–7500U is capable of training an average of 115 examples/second. So for smaller tasks CPU is sufficient. However for complex tasks a GPU may be needed otherwise it will slow down the training process.

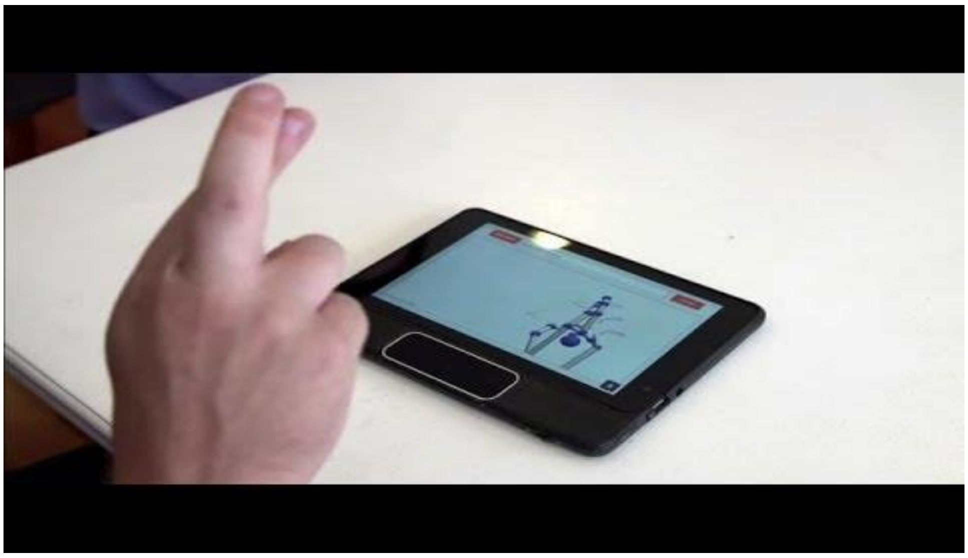
- A GPU can perform convolutional based operations on a batch of images of 128 or 256 images at once in just a few milliseconds. However, the power consumption is huge which leads to a total of 400W.

- In this system one epoch took approximately 7-8 minutes to run. Hence, an average of 25 epochs would need 3-4 hours in CPU.

Chapter 4 : Implementation Work details

4.1 Real Life Applications

- This system is widely used among many American deaf and dumb communities and their organizations.
- Also it is practiced and taught in deaf and dumb European schools.
- A google translater app Gynosys uses deep learning neural network and computer vision for translating sign language.
- Many deaf people and dumb people use mobile applications which helps them translate the sign language and have a communication with ease with other people.
- The Hand Talk is an app which is lead by Hugo translates the ASL texts and also the brazillian sign language through the help of technology like Artificial Intelligence.
- Mimx3D sign language app is another popular android application available for free in google playstore.



4.2 Data Implementation and Program Execution

The code is executed in three phases.

1. Collecting Data

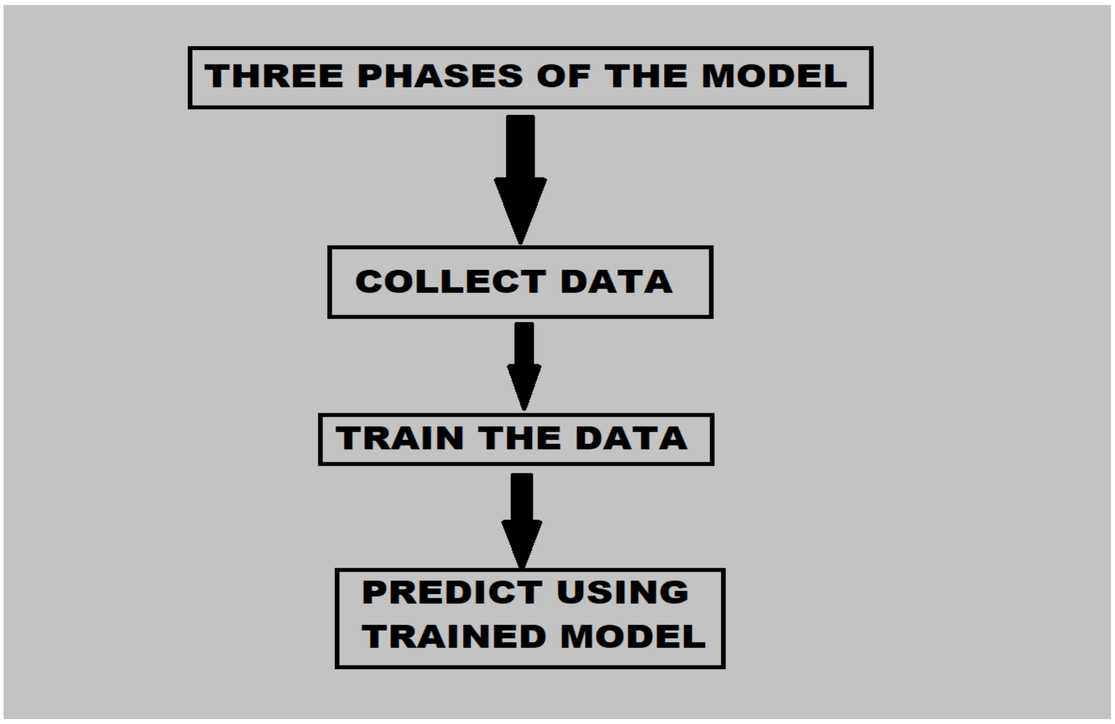
Firstly, a dataset will be created of different hand gestures. They will be split into test and train part.

2. Training

Then we will train our data using CNN. A Convolutional Neural Network is a feed forward NN just like ANN. It worls best with image like data. We train the collected data through CNN layers, compiling it and fitting it into the model.

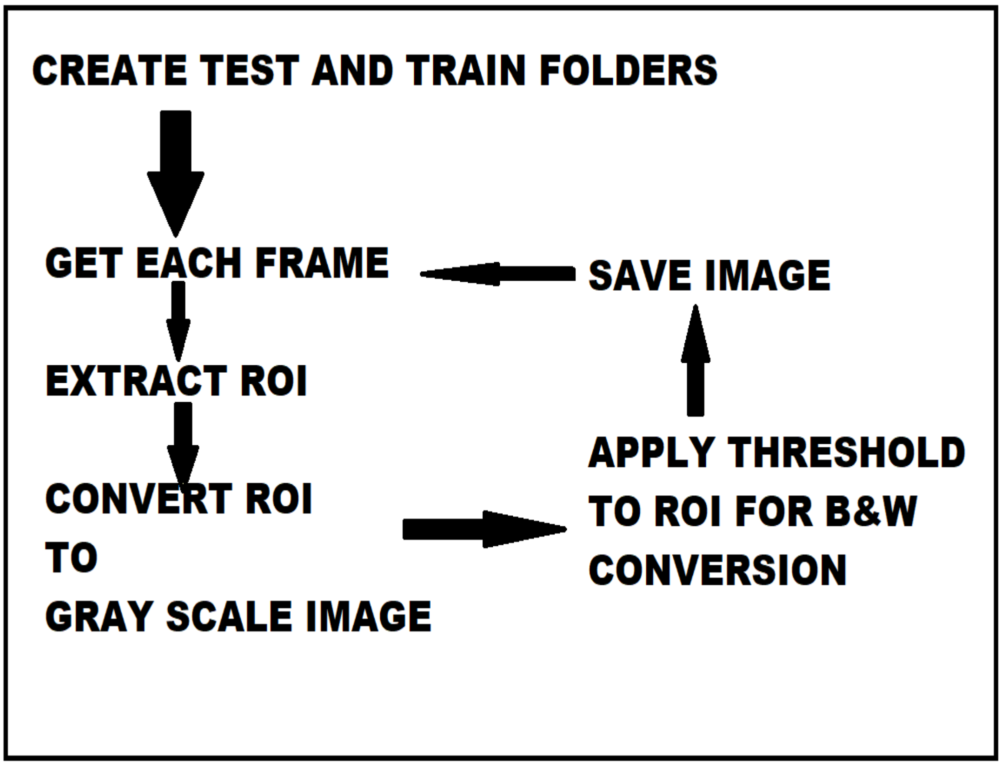
3. Prediction

Hand gestures will be captured in live video feed and predict the sign languages in real time. The system will also form sentences.



Data Collection

- create folders using os library
- draw a rectangle for extracting region of interest (roi) from each frame
- convert roi to grayscale image
- Feed the new roi to threshold function
- The threshold function gives entire black and white image using thresh_binary method
- Write the image to the train and test folders


CREATE TEST AND TRAIN FOLDERS
GET EACH FRAME
SAVE IMAGE
EXTRACT ROI
APPLY THRESHOLD TO ROI FOR B&W CONVERSION
CONVERT ROI TO GRAY SCALE IMAGE

Train the data using CNN

- CREATE CNN MODEL

  Create Neural network sequential architecture

  Add Layers (convolution, normalization, maxpooling) twice

  Flatten the model and feed it to NN

  Add Output layer

- COMPILE THE MODEL

  optimizer

  loss function

  metrics

- PREPROCESS AND DATA AUGMENTAION

  #ImageDataGenerator

  rescale 0-255 to 0-1

  shear range

  zoom

  horizontal flip

  #train_datagen.flow_from_directory

  Generator generating augmenting images

  resize image

  color_mode (gray scale)

  class_mode (categorical)

- TRAIN AND SAVETHE MODEL

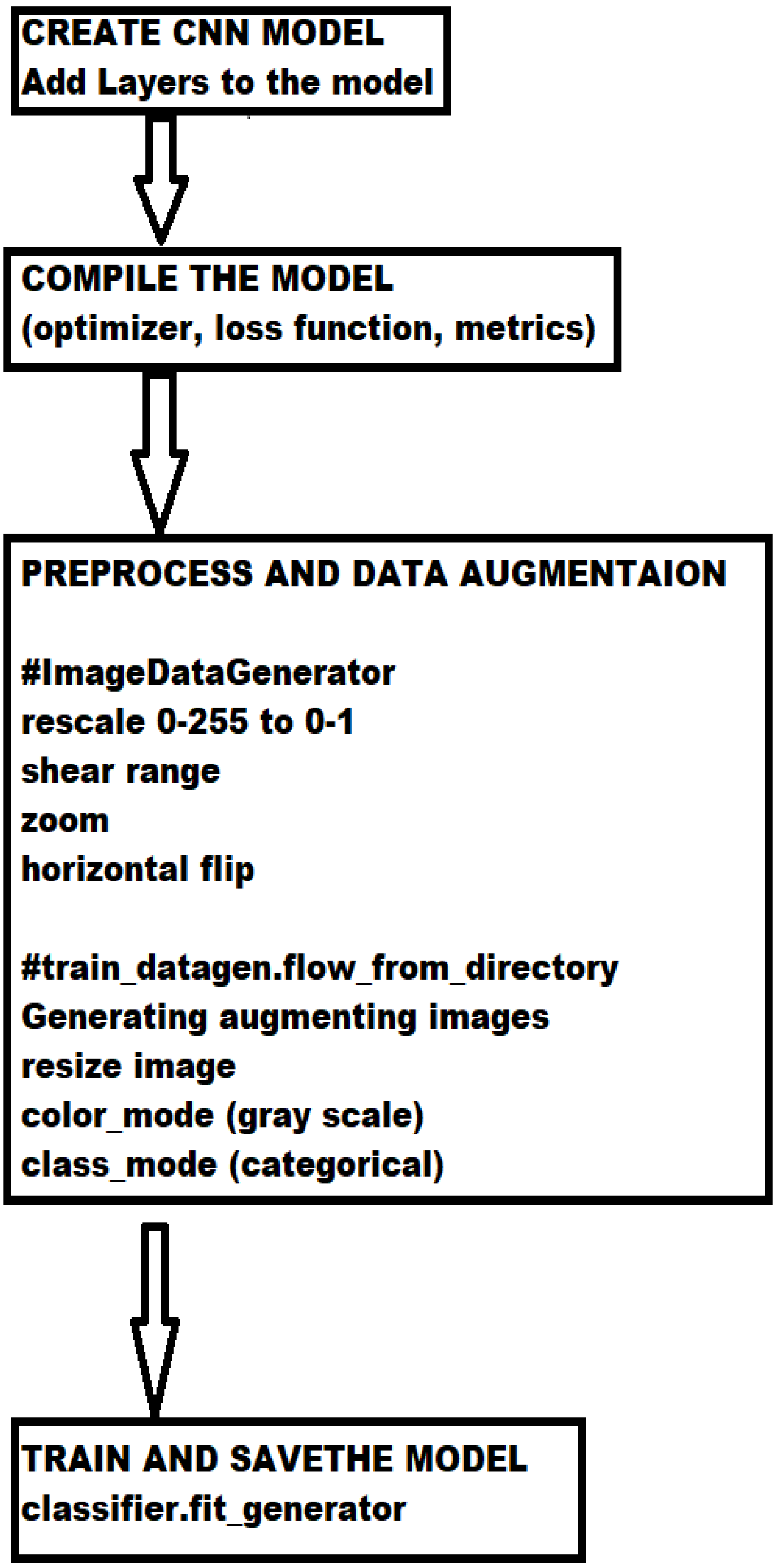
  classifier.fit_generator

  loss : during training

  val_loss : loss in validation

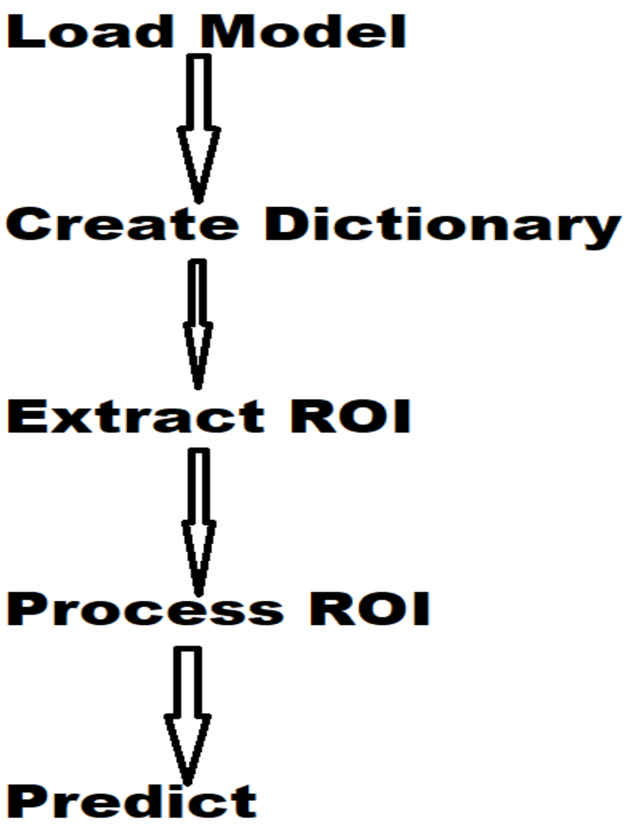
  acc : during training

  val_acc : accuracy in validation


CREATE CNN MODEL
Add Layers to the model
COMPILE THE MODEL
(optimizer, loss function, metrics)
PREPROCESS AND DATA AUGMENTAION
#ImageDataGenerator
rescale 0-255 to 0-1
shear range
zoom
horizontal flip
#train_datagen.flow_from_directory
Generating augmenting images
resize image
color_mode (gray scale)
class_mode (categorical)
TRAIN AND SAVETHE MODEL
classifier.fit_generator

Prediction

- Load Model
- Open Webcamera
- Create dictionary represting each gesture
- Extract ROI
- Gray conversion
- Apply binary threshold
- result=loaded_model.predict()
- Predict



Chapter 5 : Code Snippets

<u>DATA CREATION CODE</u>

1. to join folders

```
   os.path.join()
```

2. creates folder if path not exist

```
   if os.path.exists(path):
           os.makedir(path)
```

```
3.  cap=cv2.VideoCapture(0)

4. while True:
           _,frame=cap.read()
           frame=cv2.flip(frame,1)
           #we flip frame so that the blue box
           #does not fall in front of our face
           # 1:y-axis, 0:xaxis, -1:both
           cv2.rectangle((x1,y1),(x2,y2),(255,0,0),1)
           cv2.rectangle(src,dest,bgr,width)
           #draw a rectangle in the frame to capture the gesture


           extract roi
           resize roi
           cv2.imshow("window name",frame)
           roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

```
        _,roi=cv2.cvtColor(roi, 120,255, cv2.THRESH_BINARY)

        0-black 255-white

        B_120_W

        source image grayscale image

        threshold value

        maxVal

        thresh-binary - convert into 0 and black


        cv2.imwrite(path of train/test,roi)


        PRESS ESCAPE KEY TO EXIT WINDOW

        interrupt = cv2.waitKey(10)

        if interrupt & 0xFF == 27:

            break


5. cap.release()

   cv2.destryAllWindows()
```

TRAINING THE MODEL

```
#LIBRARIES

from keras.models import sequential

from keras.layers import convolution2D, MaxPooling2D, Flatten, Dense
```

```
classifier=sequential()

# Sequential is a stack of layers.

# Each layer has one input and output tensor.


classifier.add(Convolution2D(32, (3, 3), input_shape=(64, 64, 1), activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Convolution2D(32, (3, 3), activation='relu'))

classifier.add(MaxPooling2D(pool_size=(2, 2)))


# We added a convolution layer of 32 filters each of size 3*3.

# The input shape is 64*64  having 1 as channel (1-gray 3-rgb)

# The activation function relu is added here only so that we don't have to add
normalization layer

# Then a maxpooling layer of 2*2 size was added.

# We again perform convolution and maxpooling in our architecture

classifier.add(Flatten())

# Then we flattened our model into a neural network


classifier.add(Dense(units=128, activation='relu'))

# Dense is a neural network

# Next layer of NN has 128 nodes

# Relu converts negative value to zero.
```

```
classifier.add(Dense(units=30, activation='softmax'))

# The ouput layer has 30 nodes as it has 30 classes.

# Ouput layer grnerally uses softmax function as it is used for multiclass.

# Softmax transforms value between 0 and 1 so that they can be expressed as
probablities.


classifier.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Then we compile our model

# Optimizer ()-adam, replacement for stochastic gradient descent

# loss function (calculates error in prediction) - categorical_crossentropy

# calculates probablity difference between categories

# metrics (judges the performance of model)- accuracy

# Loss function is used while training the model but result of metrics is not


from keras.preprocessing.image import ImageDataGenerator


train_datagen = ImageDataGenerator(

    rescale=1./255,

    shear_range=0.2,

    zoom_range=0.2,

    horizontal_flip=True)
```

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


# Data Preprocessing and Augmentation

# Keras Image Data Generator helps in augmenting images even when model is
training

# rescale - original image is between 0-255. But this is too high to process.

# Therefore, we scale it between 0 to 1.

# horizontal flip flips the half images horizontally.

# shear range distorts the image along axis.


training_set = train_datagen.flow_from_directory('C:\\Users\\Hp\\ASL Minor
Project\\alphabets\\train',

                                                 target_size=(64, 64),

                                                 batch_size=5,

                                                 color_mode='grayscale',

                                                 class_mode='categorical')


test_set = test_datagen.flow_from_directory('C:\\Users\\Hp\\ASL Minor
Project\\alphabets\\test',

                                            target_size=(64, 64),

                                            batch_size=5,

                                            color_mode='grayscale',

                                            class_mode='categorical')
```

```
# This is a generator which generates batches of augment images.

# All images resize to 64*64

# color_mode - gray scale

# class_mode - categorical (sice multiple classes)


classifier.fit_generator(

        training_set,

        steps_per_epoch=22500,

        epochs=25,

        validation_data=test_set,

        validation_steps=7500)

# Using the generators to train the model

# training images - 22500

# test images – 7500


# Saving the model

model_json = classifier.to_json()

with open("model-bw.json", "w") as json_file:

    json_file.write(model_json)

classifier.save_weights('model-bw.h5')
```

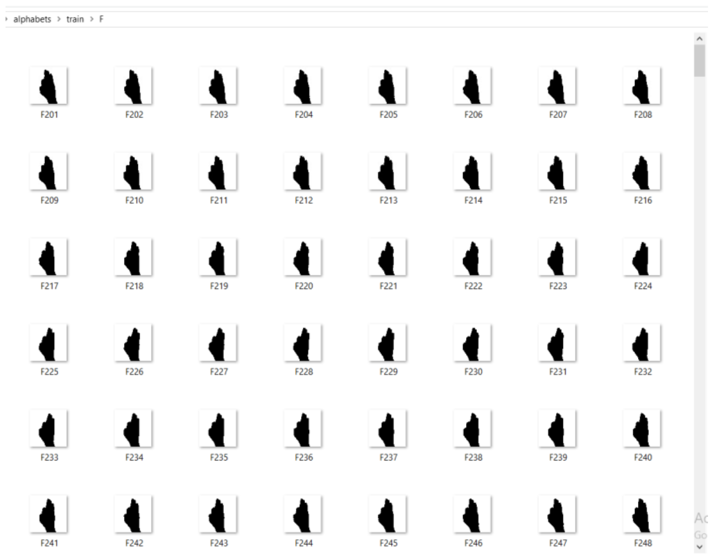
PREDICT

```
import numpy as np

from keras.models import model_from_json

#import operator

import cv2

import sys, os

#import webbrowser

#import subprocess



# Loading the model

json_file = open("model-bw.json", "r")

model_json = json_file.read()

json_file.close()

loaded_model = model_from_json(model_json)

# load weights into new model

loaded_model.load_weights("model-bw.h5")

print("Loaded model from disk")



cap = cv2.VideoCapture(0)




categories = {0: 'A', 1: 'B', 2: 'C',3: 'D', 4: 'E', 5: 'F',6: 'G', 7: 'H', 8: 'I',

            9: 'J', 10: 'K', 11: 'L',12: 'M', 13: 'N', 14: 'O',15: 'P', 16: 'Q',
        17: 'R', 18: 'S',19: 'T', 20: 'U', 21: 'V',22: 'W', 23: 'X', 24: 'Y',25: 'Z',

        26: 'SPACE', 27: 'REMOVE 1', 28: 'REMOVE ALL', 29: 'BLANK'}

s=""

d={}

p=""

count=0




image = cv2.imread("1.png")

asl=cv2.imread("asl.png")

asl=cv2.resize(asl,(400,500))

while True:



    count+=1





    _, frame = cap.read()

    frame = cv2.flip(frame, 1)



    x1 = int(0.5*frame.shape[1])

    y1 = 10

    x2 = frame.shape[1]-10

    y2 = int(0.5*frame.shape[1])
```

```
cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
roi = frame[y1:y2, x1:x2]


roi = cv2.resize(roi, (64, 64))
roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
_, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
cv2.imshow("test", test_image)
result = loaded_model.predict(test_image.reshape(1, 64, 64, 1))
prediction = {'A': result[0][0],
              'B': result[0][1],
              'C': result[0][2],
              'D': result[0][3],
              'E': result[0][4],
              'F': result[0][5],
              'G': result[0][6],
              'H': result[0][7],
              'I': result[0][8],
              'J': result[0][9],
              'K': result[0][10],
              'L': result[0][11],
              'M': result[0][12],
              'N': result[0][13],
              'O': result[0][14],
                'P': result[0][15],
                'Q': result[0][16],
                'R': result[0][17],
                'S': result[0][18],
                'T': result[0][19],
                'U': result[0][20],
                'V': result[0][21],
                'W': result[0][22],
                'X': result[0][23],
                'Y': result[0][24],
                'Z': result[0][25],
                'SPACE': result[0][26],
                'REMOVE 1': result[0][27],
                'REMOVE ALL': result[0][28],
                'BLANK': result[0][29]}


    prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
    cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 2,
(255,0,0), 1)


    if count<=500:
        if prediction[0][0] in d:
            d[prediction[0][0]]+=1
        else:
            d[prediction[0][0]]=1
        count+=1

    if count>500:
        inverse = [(value, key) for key, value in d.items()]
        a=max(inverse)[1]
        if a=='SPACE':
            s+=' '
            cv2.putText(frame, 'SPACE PRINTED' , (10, 100),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
            cv2.waitKey(5000)
        elif a=='REMOVE 1':
            s=s[:-1]
        elif a=='REMOVE ALL':
            s=''
        elif a!='BLANK':
            s+=a
        d={}
        count=0

    cv2.putText(frame, s, (10, 400), cv2.FONT_HERSHEY_PLAIN, 3, (0,255,255), 1)
    cv2.imshow("Frame", frame)

    cv2.imshow("SIGNS",asl)


    interrupt = cv2.waitKey(10)

    if interrupt & 0xFF == 27:

        break


cap.release()

cv2.destroyAllWindows()
```

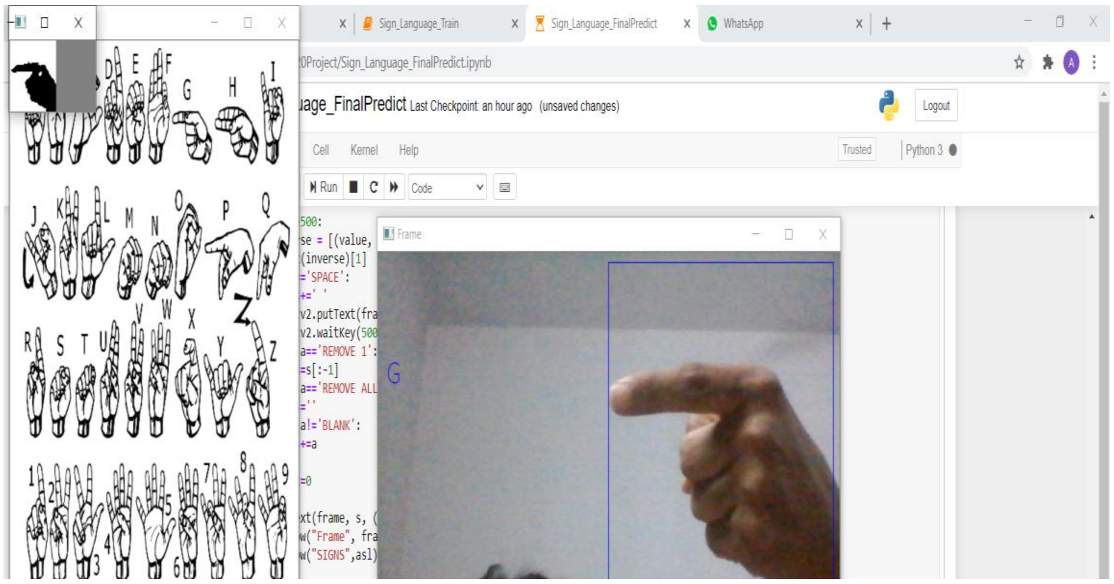
Chapter 6 :Output

Dataset



alphabets > train > F
F201 F202 F203 F204 F205 F206 F207 F208
F209 F210 F211 F212 F213 F214 F215 F216
F217 F218 F219 F220 F221 F222 F223 F224
F225 F226 F227 F228 F229 F230 F231 F232
F233 F234 F235 F236 F237 F238 F239 F240
F241 F242 F243 F244 F245 F246 F247 F248

Prediction



Sign_Language_Train
Sign_Language_FinalPredict
WhatsApp
0Project/Sign_Language_FinalPredict.ipynb
uage_FinalPredict Last Checkpoint: an hour ago (unsaved changes)
Logout
Cell
Kernel
Help
Trusted
Python 3
Run
Code
Frame
G

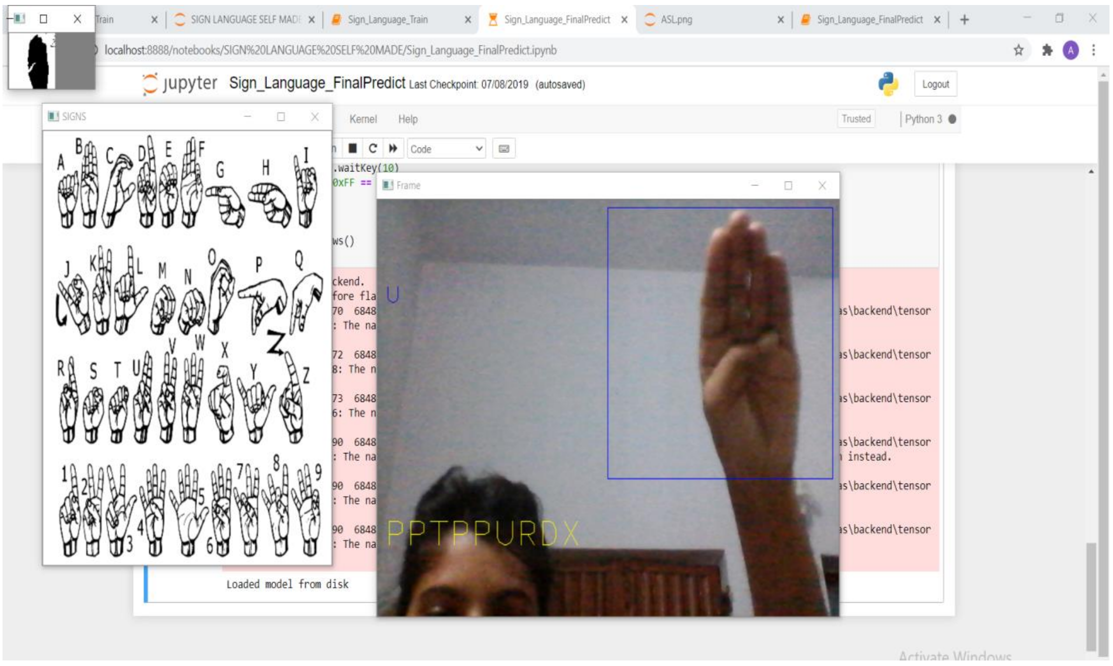
```
500:
se = [(value,
(inverse)[1]
='SPACE':
+=' '
v2.putText(fra
v2.waitKey(500
a=='REMOVE 1':
=s[:-1]
a=='REMOVE ALL
=''
a!='BLANK':
+=a

=0

xt(frame, s, (
w("Frame", fra
w("SIGNS",asl)
```

Training Epoch

```
Epoch 1/25
22500/22500 [==============================] - 895s
40ms/step - loss: 0.0646 - acc: 0.9824 - val_loss: 0.7378
- val_acc: 0.9207
Epoch 2/25
22500/22500 [==============================] - 865s
38ms/step - loss: 0.0077 - acc: 0.9981 - val_loss: 0.6174
- val_acc: 0.9411
Epoch 3/25
22500/22500 [==============================] - 826s
37ms/step - loss: 0.0067 - acc: 0.9985 - val_loss: 0.7771
- val_acc: 0.9401
Epoch 4/25
22500/22500 [==============================] - 889s
40ms/step - loss: 0.0049 - acc: 0.9990 - val_loss: 1.0078
- val_acc: 0.9256
Epoch 5/25
22500/22500 [==============================] - 938s
42ms/step - loss: 0.0050 - acc: 0.9992 -
Epoch 21/25
22500/22500 [==============================] - 818s
36ms/step - loss: 0.0159 - acc: 0.9989 - val_loss: 1.0940
- val_acc: 0.9293
Epoch 22/25
22500/22500 [==============================] - 835s
37ms/step - loss: 0.0152 - acc: 0.9989 - val_loss: 0.8378
- val_acc: 0.9464
Epoch 23/25
22500/22500 [==============================] - 826s
37ms/step - loss: 0.0139 - acc: 0.9990 - val_loss: 0.9330
- val_acc: 0.9413
Epoch 24/25
22500/22500 [==============================] - 817s
36ms/step - loss: 0.0149 - acc: 0.9990 - val_loss: 0.8196
- val_acc: 0.9475
Epoch 25/25
22500/22500 [==============================] - 857s
38ms/step - loss: 0.0106 - acc: 0.9992 - val_loss: 1.0233
- val_acc: 0.9348
```

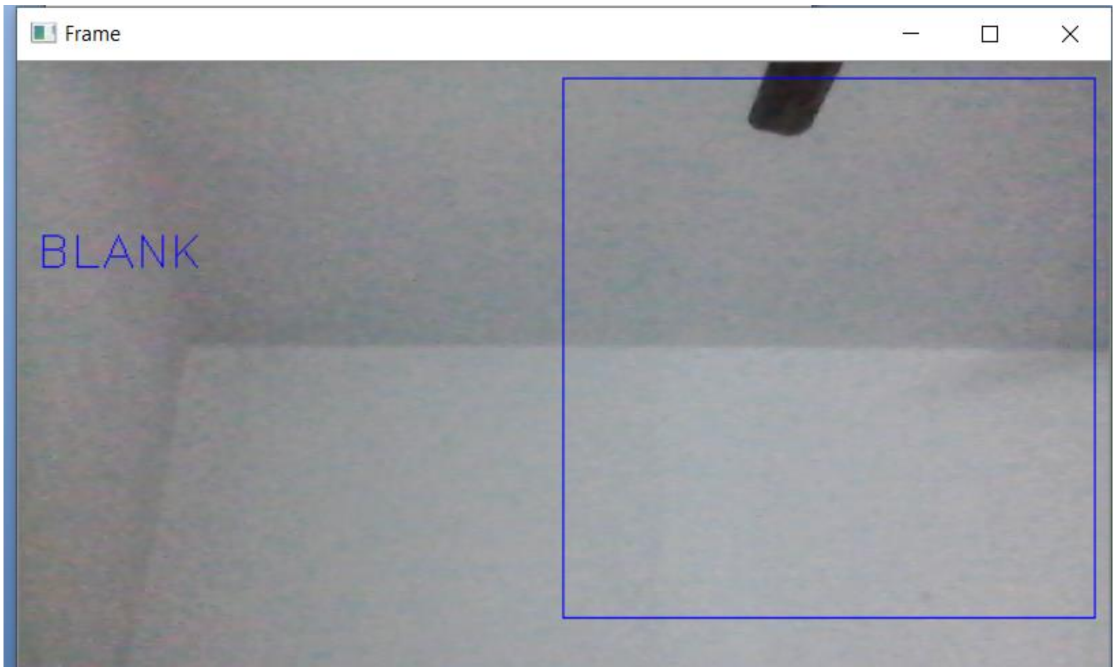
Chapter 7 : System Testing

Senetence Formation

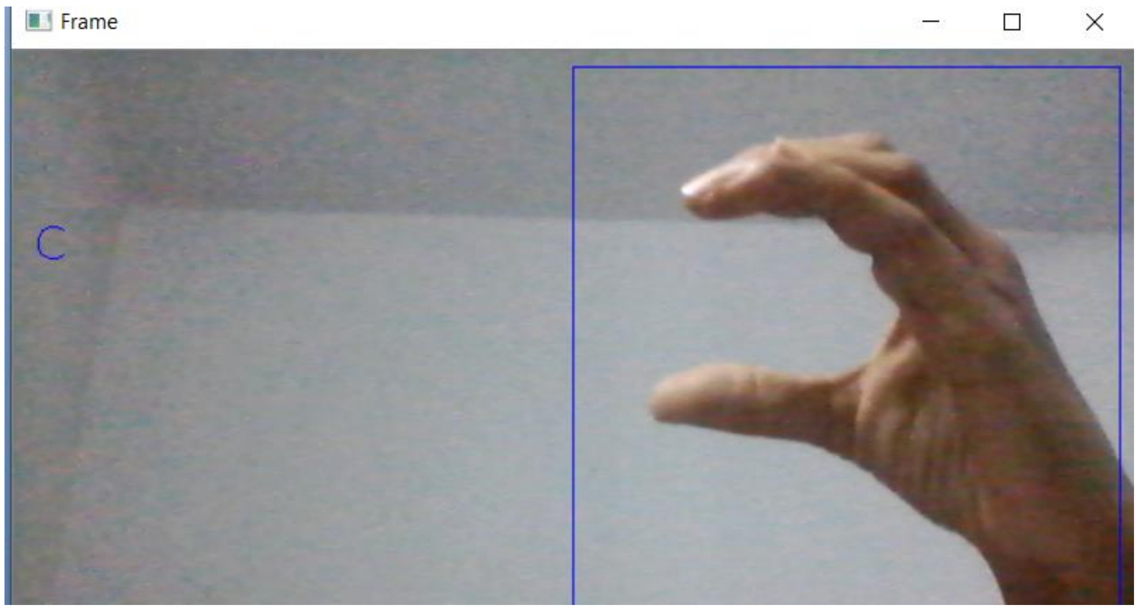

SIGN LANGUAGE SELF MADE
Sign_Language_Train
Sign_Language_FinalPredict
ASL.png
localhost:8888/notebooks/SIGN%20LANGUAGE%20SELF%20MADE/Sign_Language_FinalPredict.ipynb
jupyter Sign_Language_FinalPredict Last Checkpoint: 07/08/2019 (autosaved)
Logout
SIGNS
Kernel
Help
Trusted
Python 3
Code
Frame
U
PPTPPURDX

```
Loaded model from disk
```

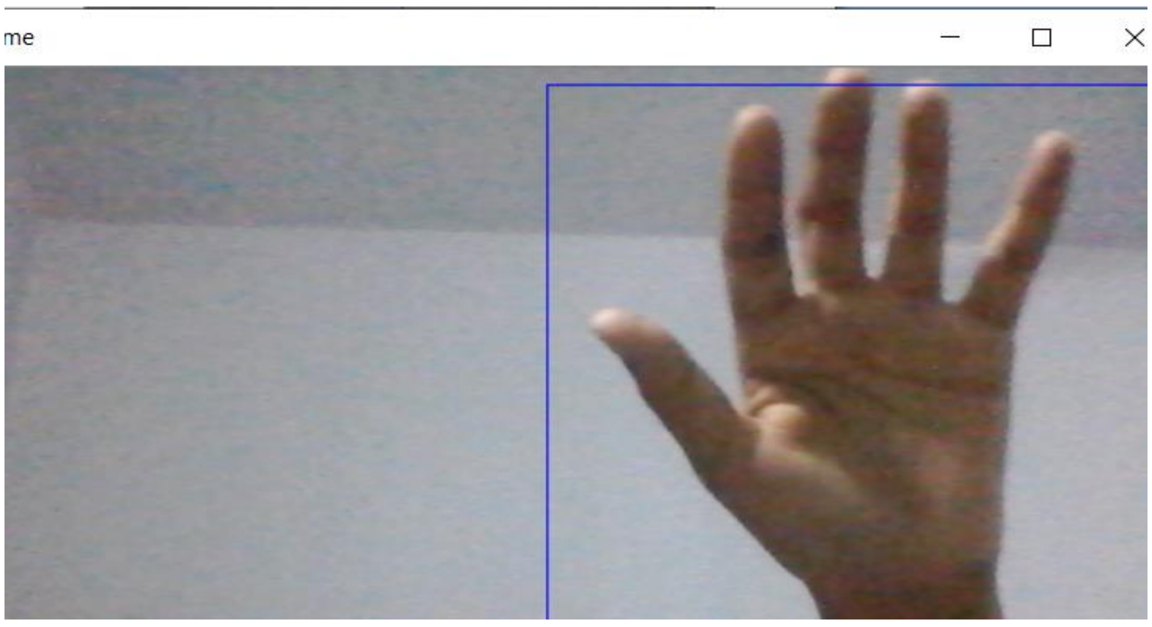
Blank



Character



Space



Chapter 8 : Individual Contribution

Aparajita Halder

(170396)

I am the sole contributor of this project. This is a self made project ceated within a span of 2-3 months under the guidance of Ms. Sonal Shukla. I collected the dataset on my own. The dataset was contributed by my friends, family and myself. It was collected through live webcam of my laptop.

Then I pre-processed the collected data. I converted the image into gray scale conversion and stored in in binary (black and white format), which will make the prediction process easier.

Then the collected dataset was trained through CNN. Convolutional and MaxPooling layers were applied to the images. The model was compiled and preprocessed. Then, it was fit into the model and then trained using 25 epochs. After training the model was saved into the data.

In the prediction process the trained model was used for predicting the outcomes and all the alphabets alongwith four extra hand gestures were stored in a data dictionary format in python so that I can easily extract and match with the gestures and thus, system makes a good prediction and also helps in sentence formation, edition, deletion and execution.

I presented it in front of the assigned panel the outcome of my project which follows proper system design and planning as per first phase of evaluation. The report was compiled and made by me under the guidance of my mentor. All the tasks form designing, planning and executing was contributed by me. I will ensure future enhancement and maintainability of this project.

CONCLUSION

LIMITATION

1. Fluctuation in prediction

Due to huge no. of classes and similar type of hand gestures there may be fluctuations in the model. Though, we can minimize the false prediction by extracting the gesture which has maximum no. of occurrence in a given time period.

2. Only predicts pre-defined gestures

The system will only be able to guess only the pre-defined gestures. If the user shows some other gesture it will predict some similar hand gesture.

3. Speed

It may take some time to predict the hand gesture and form a sentence than the user would have been able to communicate in a real life scenario.

SCOPE

In future this project canbe deployed in cloud and it’s benefits can be leveraged by not only specially abled people but also by normal people to have an ease of communication. This system can be used among many American deaf and dumb communities and their organizations. Also it can be practiced and taught in deaf and dumb European schools. It can also act as a teacher/translator. A google translater app Gynosys uses deep learning neural network and computer vision for translating sign language. Many deaf people and dumb people use mobile applications which helps them translate the sign language and have a communication with ease with other people. The Hand Talk is an app which is lead by Hugo translates the ASL texts and also the brazillian sign language through the help of technology like Artificial Intelligence. Mimx3D sign language app is another popular android application available for free in google playstore. Existence of such apps makes our belief in the future of this system more dominant.

FUTURE ENHANCEMENT

- **Dataset Update**

  There might be a need to add new hand gestures in future. The system is providing a feature so that users will be able to collect their own user defined hand gestures as per their requirement.

- **Optimization**

  Measures will be taken to maximize efficiency and reliability of the system by updating code and updating library versions.

- **Computational Speed**

  The speed of the system can be optimized by using GPU instead of CPU. If we implement the system in GPU then, we will be able to increase the accuracy of the model by increasing no. of epochs and dataset captured.

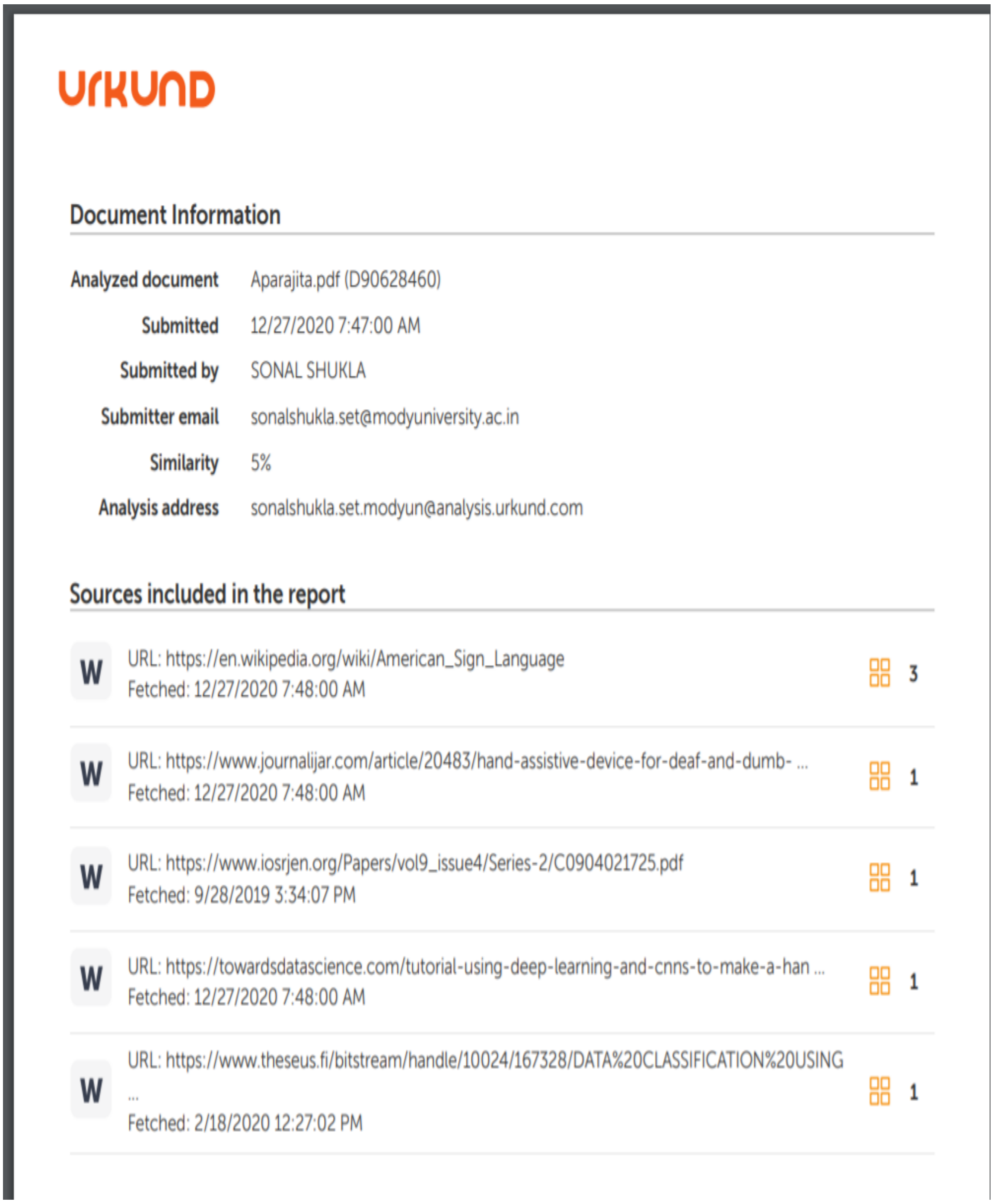
BIBILOGRAPHY

https://towardsdatascience.com/american-sign-language-recognition-using-cnn-36910b86d651

https://ieee-dataport.org/open-access/transfer-learning-cnn-american-sign-language

https://analyticsindiamag.com/hands-on-guide-to-sign-language-classification-using-cnn/

PLAGARISM



urkund

Document Information

Analyzed document	Aparajita.pdf (D90628460)
Submitted	12/27/2020 7:47:00 AM
Submitted by	SONAL SHUKLA
Submitter email	sonalshukla.set@modyuniversity.ac.in
Similarity	5%
Analysis address	sonalshukla.set.modyun@analysis.urkund.com

Sources included in the report

W	URL: https://en.wikipedia.org/wiki/American_Sign_Language Fetched: 12/27/2020 7:48:00 AM	3
W	URL: https://www.journalijar.com/article/20483/hand-assistive-device-for-deaf-and-dumb- ... Fetched: 12/27/2020 7:48:00 AM	1
W	URL: https://www.iosrjen.org/Papers/vol9_issue4/Series-2/C0904021725.pdf Fetched: 9/28/2019 3:34:07 PM	1
W	URL: https://towardsdatascience.com/tutorial-using-deep-learning-and-cnns-to-make-a-han ... Fetched: 12/27/2020 7:48:00 AM	1
W	URL: https://www.theseus.fi/bitstream/handle/10024/167328/DATA%20CLASSIFICATION%20USING ... Fetched: 2/18/2020 12:27:02 PM	1