International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 108-117
Published Online January 2023 in IJEAST (http://www.ijeast.com)

SIGN LANGUAGE RECOGNITION USING LSTM DEEP LEARNING MODEL

Rutkumar Nitinchandra Patel
B.E. Information Technology Graduate,
SVIT, Vasad, Anand, Gujarat, India

Abstract- **Speech impaired people use hand signs and gestures to communicate. The problem is translator needs every time to translate sign language. Artificial Intelligence capture hand expression with the help of camera show us what a person says and result shown in the text format. In the starting stage we provide the data for sign language for better result with good accuracy and testing this model. Various deepmachine learning algorithms are applied on the datasets, including Long Short-Term Memory (LSTM) RNN.**

Index Terms- **Action Detection, Deep neural network, LSTM Model, and Sign Language.**

I. INTRODUCTION

Communicationis all around us, from the moment we wake and read the newspaper, turn on the television, pass the advertising boards on the way to work and listen to the train announcements. In our daily life, communication helps us to build relationships by allowing us to share our experiences, and the needs, and helps us connect to others. It's the essence of the life, allowing us to express feelings, pass on information and share thoughts. However, unfortunately, for the people who severe speaking or hearing impairments, there is a communication gap. In the deaf and dumb community, sign language plays a vital role to communicate with people. Sign language is the mode of communication which uses visual ways like expressions, hand gesture, and body movements to convey meaning. Sign language recognition refers to the conversion of the gestures into the words or alphabets of existing formally spoken languages. Thus, conversion of sign language into words by an algorithm or a model can help bridge the gap between people with hearing or speaking impairment and the rest of the world.

There are more than 120 distinct sign languages, such as American sign language, Indian Sign Language, Italian sign language, etc. for example [Fig.1]. In this proposed model, American sign language (ASL) is used to provide an example. The signs that were included are commonly used words, such as hello, goodbye, good morning, thanks, etc.

Figure.1 American Sign Language

The system was designed so that it uses the natural gesture input to create a sign language and then passes this to the system for further pre-processing and processing tasks to predict the exact word that the gesture expressed. There are basically two types of approaches: vision-based and gloves-based for hand gesture recognition. This work main focus is on creating a vision-based system to do real-time sign language recognition.

II. OBJECTIVE

The Sign Language Recognition Prototype is a real-time vision-based system whose purpose is to identify the American Sign Language given in the alphabets of Fig. 1. The aim of the prototype was to test the validity of a vision-based system for sign language recognition and at the same time, test and select hand features that could be used with machine learning algorithms allowing their application in any real-time sign language recognition systems.

The implemented solution uses only one camera, and is based on a set of assumptions, hereby defined:

1. The user must be within a defined distance range, due to camera limitations.
2. Hand pose is defined with a bare hand and not occluded by other objects.
3. The system must be used indoor, since the selected camera does not work well under sun light conditions.

III. METHODOLOGY

Sign language recognition is a real-time sign language detection flow using python to build a system which detects a bunch of different poses and specifically sign language signs using a key model. For this, Media pipe holistic is used to

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

extract key points from user hands, and body, and face. Once key points are extracted then the LSTM model is built to predict the actions which are captured using webcam in this particular case actions are going to be a sign language. Furthermore, let's see our approach to build a model step by step:

A. Key points Using Media pipe Holistic
Key points detection consists of locating key object parts. For example, the key parts of our faces include nose tips, eyebrows, eye corners, and so on. We used the MediaPipe Holistic, it is one of the pipelines which contains optimized face, hands, and pose components which allows for holistic tracking, thus enabling the model to simultaneously detect hand and body poses along with the face landmarks. One of main reason for using media pipe holistic is to detect face and hands and extract key points to pass on to a computer vision model.
In addition, Media pipe holistic utilize the pose, face and hand landmark models, respectively generate a total of 543 landmarks (33 pose landmarks, 468 face landmarks, and 21 hand landmarks per hand.)

For example, the following code snippet is a function to access image input from system web camera using OpenCV framework, and detect and extract keypoints for face and hands (Full Code is in section IV).

```
# Draw face connections
mp_drawing.draw_landmarks (image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
mp_drawing.DrawingSpec(color=(80,110,10),thickness=1,circle_radius=1),
 mp_drawing.DrawingSpec(color=(80,256,12), thickness=1, circle_radius=1))
```

Results:

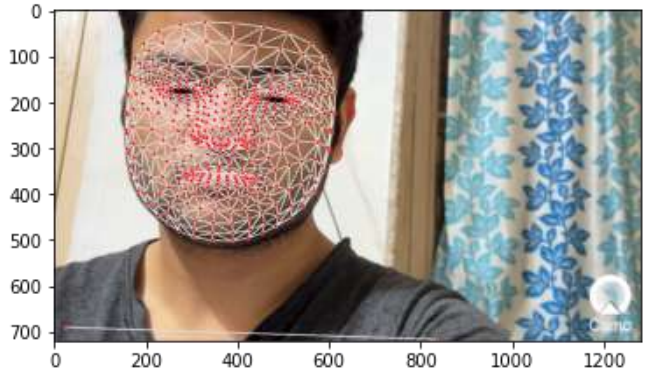


Figure 2. Keypoints for face landmark

B. Collecting Key points values for training and Testing.
Here we are collecting data for three different actions: "Hello", "Thank you", and "I Love You". For each action we collect 30 sequence or videos data and sequences or videos are in 30 frames in length, So, that means 30 different key points and all datas are stored in numpy array. Following figure shows data collection for "Hello" action:


Figure 3. Collecting keypoints for gesture "Hello".

C. Build and Train LSTM neural network
In this prototype, Keras and Tensor Flow are used to create a simple LSTM model, and train and test. Using Keras and Tensor Flow makes building neural network much easier to build than from scratch. First we need to import following import modules:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
```

for an LSTM neural network is that it has to have LSTM cells or at least one LSTM layer. If we add different types of layers and cells, we can still call our neural network an LSTM, but it would be more accurate to give it a mixed name. To build an LSTM model, first we initialize a sequential model then we add LSTM layers, which makes LSTM neural network and afterwards a dense (fully connected) output layer is added. Code as follow:

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

```
model = Sequential()
model.add(LSTM(64,return_sequences=
True,activation='relu',input_shape=(
30,1662)))
model.add(LSTM(128,return_sequences=
True, activation='relu'))
model.add(LSTM(64,return_sequences=
False, activation='relu'))
model.add(Dense(64,activation='relu'
))
model.add(Dense(32,activation='relu'
))
model.add(Dense(actions.shape[0],
activation='softmax'))
```

LSTM layer has a way more parameters than a simple RNN layer. LSTM layers has four times the number of parameters as Now, it's time to load data, here for our prototype data are in sequences and labels. Previously, these data are stored in numpy array.

```
from sklearn.model_selection import
train_test_split
from tensorflow.keras.utils import
to_categorical
label_map = {label:num for num,
label in enumerate(actions)}
sequences, labels = [], []
for action in actions:
    for sequence in
range(no_sequences):
        window = []
        for frame_num in
range(sequence_length):
            res =
np.load(os.path.join(DATA_PATH,
action, str(sequence),
"{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
labels.append(label_map[action])

X = np.array(sequences)
y=to_categorical(labels).astype(int)
```

X_train, X_test, y_train, y_test=train_test_split (X,y,test_size=0.05)

a simple RNN layer. Below model. summary() shows the parameters for our prototype:

```
Model. summary()
```

Result:

Layer (type)	Output Shape	Param #
Lstm_6 (LSTM)	(None, 30, 64)	442112
lstm_7 (LSTM)	(None, 30, 64)	98816
lstm_8 (LSTM)	(None, 64)	49408
dense_6 (Dense)	(None, 64)	4160
dense_7 (Dense)	(None, 32)	2080
dense_8 (Dense)	(None, 2)	66
Total params: 596,642 Trainable params: 596,642 Non-trainable params: 0		

Table 1. Shows parameters for our model

For model fitting x values are in sequences means our hand gestures movements and y values labels ('Hello': 0, 'Thanks': 1, 'I Love You': 2). Before we test or train our model we have to compile. In our model compilation we will specify the loss function, in this case Categorical Cross Entropy, our optimizer, Adam and our metrics, Categorical Accuracy.

```
model.compile(optimizer='Adam',
loss='categorical_crossentropy',
metrics=['categorical_accuracy'])

model.fit(X_train, y_train,
epochs=2000,callbacks=[tb_callback])
```

Now, the last thing is to test the model, for that we predict the sample data which are collected previously. This code for that:

```
res=model.predict(np.expand_dims(
      sequence, axis=0))[0]

print(actions[np.argmax(res)])
```

Now, Sign Language recognition model is ready to convert sign languages or fingerspelling into the text format and full code for this model is in section IV. for example (Figure. 3) shows the final output of the model:

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

Figure. 4: Final output

IV. CODE FOR IMPLEMENTATION

Here is the full code for the model:

```
pip install tensorflow==2.10.1
tensorflow-gpu==2.10.1 opencv-
python mediapipe sklearn
matplotlib
import cv2
import numpy as np
import os
from matplotlib import pyplot as
plt
import time
import mediapipe as mp
mp_holistic =
mp.solutions.holistic # Holistic
model
mp_drawing =
mp.solutions.drawing_utils #
Drawing utilities
def mediapipe_detection(image,
model):
    image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB) # COLOR
CONVERSION BGR 2 RGB
    image.flags.writeable = False
# Image is no longer writeable
    results = model.process(image)
# Make prediction
    image.flags.writeable = True
# Image is now writeable
    image = cv2.cvtColor(image,
cv2.COLOR_RGB2BGR) # COLOR
COVERSION RGB 2 BGR
    return image, results
def draw_landmarks(image,
results):
mp_drawing.draw_landmarks(image,
results.face_landmarks,
mp_holistic.FACEMESH_TESSELATION)
# Draw face connections
mp_drawing.draw_landmarks(image,
results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS) #
Draw pose connections
mp_drawing.draw_landmarks(image,
results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS) #
Draw left hand connections
mp_drawing.draw_landmarks(image,
results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS) #
Draw right hand connections
def draw_styled_landmarks(image,
results):
    # Draw face connections
mp_drawing.draw_landmarks(image,
results.face_landmarks,
mp_holistic.FACEMESH_TESSELATION,
mp_drawing.DrawingSpec(color=(80,1
10,10), thickness=1,
circle_radius=1),
mp_drawing.DrawingSpec(color=(80,2
56,121), thickness=1,
circle_radius=1)
                                )
    # Draw pose connections
mp_drawing.draw_landmarks(image,
results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
mp_drawing.DrawingSpec(color=(80,2
2,10), thickness=2,
circle_radius=4),
mp_drawing.DrawingSpec(color=(80,4
4,121), thickness=2,
circle_radius=2)
                                )
```

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

```
                                 )
    # Draw left hand connections
mp_drawing.draw_landmarks(image,
results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
mp_drawing.DrawingSpec(color=(121,
22,76), thickness=2,
circle_radius=4),
mp_drawing.DrawingSpec(color=(121,
44,250), thickness=2,
circle_radius=2)
                                 )
    # Draw right hand connections
mp_drawing.draw_landmarks(image,
results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
mp_drawing.DrawingSpec(color=(245,
117,66), thickness=2,
circle_radius=4),
mp_drawing.DrawingSpec(color=(245,
66,230), thickness=2,
circle_radius=2)
                                 )
cap = cv2.VideoCapture(1)
```

```
# Set mediapipe model
with
mp_holistic.Holistic(min_detection
_confidence=0.5,
min_tracking_confidence=0.5) as
holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results =
mediapipe_detection(frame,
holistic)
        print(results)

        # Draw landmarks
draw_styled_landmarks(image,
results)

        # Show to screen
        cv2.imshow('OpenCV Feed',
image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF
== ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

```
draw_landmarks(frame, results)
plt.imshow(cv2.cvtColor(frame,
cv2.COLOR_BGR2RGB))
```

```
pose = []
for res in
results.pose_landmarks.landmark:
    test = np.array([res.x, res.y,
res.z, res.visibility])
    pose.append(test)

pose = np.array([[res.x, res.y,
res.z, res.visibility] for res in
results.pose_landmarks.landmark]).
flatten() if
results.pose_landmarks else
```

```
np.zeros(132)
face = np.array([[res.x, res.y,
res.z] for res in
results.face_landmarks.landmark]).
flatten() if
results.face_landmarks else
np.zeros(1404)
lh = np.array([[res.x, res.y,
res.z] for res in
results.left_hand_landmarks.landma
rk]).flatten() if
results.left_hand_landmarks else
np.zeros(21*3)
rh = np.array([[res.x, res.y,
res.z] for res in
results.right_hand_landmarks.landm
ark]).flatten() if
results.right_hand_landmarks else
np.zeros(21*3)

def extract_keypoints(results):
    pose = np.array([[res.x,
res.y, res.z, res.visibility] for
res in
```

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

```
results.pose_landmarks.landmark]).
flatten() if
results.pose_landmarks else
np.zeros(33*4)
    face = np.array([[res.x,
res.y, res.z] for res in
results.face_landmarks.landmark]).
flatten() if
results.face_landmarks else
np.zeros(468*3)
    lh = np.array([[res.x, res.y,
res.z] for res in
results.left_hand_landmarks.landma
rk]).flatten() if
results.left_hand_landmarks else
np.zeros(21*3)
    rh = np.array([[res.x, res.y,
res.z] for res in
results.right_hand_landmarks.landm
ark]).flatten() if
results.right_hand_landmarks else
np.zeros(21*3)
    return np.concatenate([pose,
face, lh, rh])
```

```
result_test =
extract_keypoints(results)
```

```
# Path for exported data, numpy
arrays
DATA_PATH =
os.path.join("/Users/Rut/MP_Data")

# Actions that we try to detect
actions = np.array(['Hello',
'Thanks'])

# Thirty videos worth of data
no_sequences = 30

# Videos are going to be 30 frames
in length
sequence_length = 30

for action in actions:
    for sequence in
range(no_sequences):
        try:
os.makedirs(os.path.join(DATA_PATH
, action, str(sequence)))
        except:
            pass
```

```
cap = cv2.VideoCapture(1)
# Set mediapipe model
with
mp_holistic.Holistic(min_detection
_confidence=0.5,
min_tracking_confidence=0.5) as
holistic:

    # NEW LOOP
    # Loop through actions
    for action in actions:
        # Loop through sequences
aka videos
        for sequence in
range(no_sequences):
            # Loop through video
length aka sequence length
            for frame_num in
range(sequence_length):
```

```
                # Read feed
                ret, frame =
cap.read()

                # Make detections
                image, results =
mediapipe_detection(frame,
holistic)
#                 print(results)

                # Draw landmarks
draw_styled_landmarks(image,
results)

                # NEW Apply wait
logic
                if frame_num == 0:
cv2.putText(image, 'STARTING
COLLECTION', (120,200),
cv2.FONT_HERSHEY_SIMPLEX, 1,
(0,255, 0), 4, cv2.LINE_AA)
cv2.putText(image, 'Collecting
frames for {} Video Number
{}'.format(action, sequence),
(15,12),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 255), 1, cv2.LINE_AA)
                    # Show to
```

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

```
screen
cv2.imshow('OpenCV Feed', image)
cv2.waitKey(2000)
                else:
cv2.putText(image, 'Collecting
frames for {} Video Number
{}'.format(action, sequence),
(15,12),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 255), 1, cv2.LINE_AA)
                    # Show to
screen
cv2.imshow('OpenCV Feed', image)

                # NEW Export
keypoints
                keypoints =
extract_keypoints(results)
                npy_path =
os.path.join(DATA_PATH, action,
str(sequence), str(frame_num))
```

```
                np.save(npy_path,
keypoints)

                # Break gracefully
                if cv2.waitKey(10)
& 0xFF == ord('q'):
                    break

    cap.release()
    cv2.destroyAllWindows()

from sklearn.model_selection
import train_test_split
from tensorflow.keras.utils import
to_categorical

label_map = {label:num for num,
label in enumerate(actions)}
```

```
sequences, labels = [], []
for action in actions:
    for sequence in
range(no_sequences):
        window = []
        for frame_num in
range(sequence_length):
            res =
np.load(os.path.join(DATA_PATH,
action, str(sequence),
"{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
labels.append(label_map[action])

X = np.array(sequences)

y =
to_categorical(labels).astype(int)

X_train, X_test, y_train, y_test =
train_test_split(X, y,
test_size=0.05)

from tensorflow.keras.models
import Sequential
from tensorflow.keras.models
import Model
from tensorflow.keras.layers
```

```
import LSTM, Dense
from tensorflow.keras.callbacks
import TensorBoard

log_dir = os.path.join('Logs')
tb_callback =
TensorBoard(log_dir=log_dir)

model = Sequential()
model.add(LSTM(64,
return_sequences=True,
activation='relu',
input_shape=(30,1662)))
model.add(LSTM(128,
return_sequences=True,
activation='relu'))
model.add(LSTM(64,
return_sequences=False,
activation='relu'))
model.add(Dense(64,
activation='relu'))
model.add(Dense(32,
activation='relu'))
model.add(Dense(actions.shape[0],
activation='softmax'))
```

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

```
res = [.7, 0.2, 0.1]

actions[np.argmax(res)]

model.compile(optimizer='Adam',
loss='categorical_crossentropy',
metrics=['categorical_accuracy'])

model.fit(X_train, y_train,
epochs=2000,
callbacks=[tb_callback])

model.summary()

res = model.predict(X_test)
```

```
from sklearn.metrics import
multilabel_confusion_matrix,
accuracy_score

yhat = model.predict(X_test)

ytrue = np.argmax(y_test,
axis=1).tolist()
yhat = np.argmax(yhat,
```

```
axis=1).tolist()

multilabel_confusion_matrix(ytrue,
yhat)

accuracy_score(ytrue, yhat)
```

```
colors = [(245,117,16),
(117,245,16), (16,117,245)]
def prob_viz(res, actions,
input_frame, colors):
    output_frame =
input_frame.copy()
    for num, prob in
enumerate(res):
cv2.rectangle(output_frame,
(0,60+num*40), (int(prob*100),
90+num*40), colors[num], -1)
        cv2.putText(output_frame,
actions[num], (0, 85+num*40),
cv2.FONT_HERSHEY_SIMPLEX, 1,
(255,255,255), 2, cv2.LINE_AA)

    return output_frame

plt.figure(figsize=(17,17))
plt.imshow(prob_viz(res, actions,
image, colors))
```

```
# 1. New detection variables
sequence = []
sentence = []
threshold = 0.8

cap = cv2.VideoCapture(1)
# Set mediapipe model
with
mp_holistic.Holistic(min_detection
_confidence=0.5,
min_tracking_confidence=0.5) as
holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results =
```

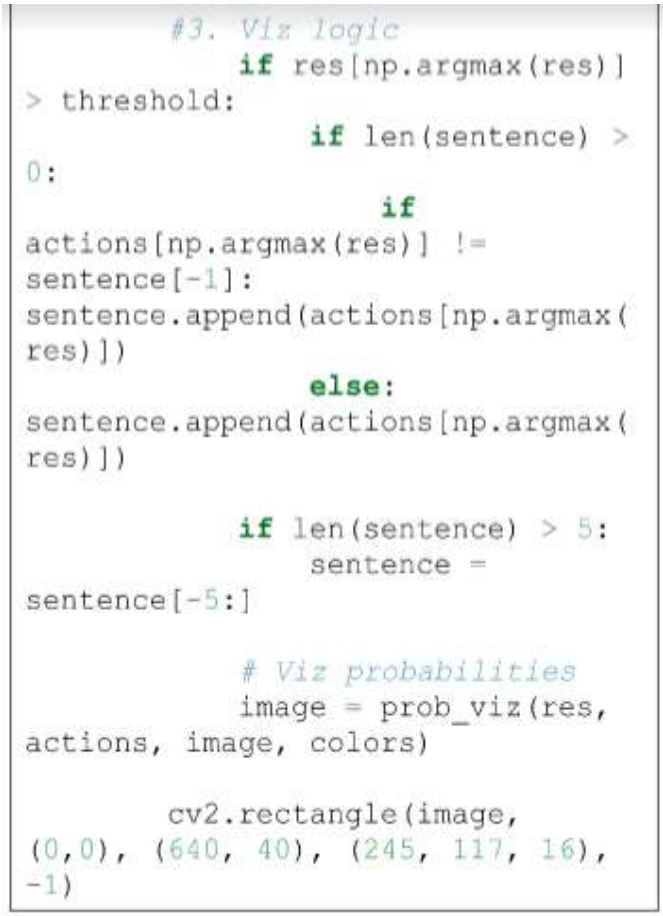
```
mediapipe_detection(frame,
holistic)
        print(results)

        # Draw landmarks
draw_styled_landmarks(image,
results)

        # 2. Prediction logic
        keypoints =
extract_keypoints(results)
#
sequence.insert(0,keypoints)
#         sequence = sequence[:30]
        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res =
model.predict(np.expand_dims(seque
nce, axis=0))[0]
print(actions[np.argmax(res)])
```

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 9, ISSN No. 2455-2143, Pages 118-127
Published Online January 2023 in IJEAST (http://www.ijeast.com)

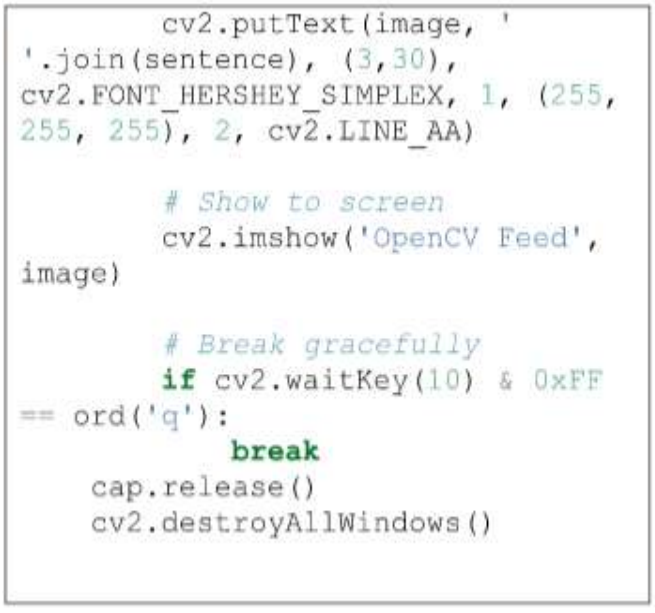


```
        #3. Viz logic
            if res[np.argmax(res)]
> threshold:
                if len(sentence) >
0:
                    if
actions[np.argmax(res)] !=
sentence[-1]:
sentence.append(actions[np.argmax(
res)])
                else:
sentence.append(actions[np.argmax(
res)])

            if len(sentence) > 5:
                sentence =
sentence[-5:]

            # Viz probabilities
            image = prob_viz(res,
actions, image, colors)

        cv2.rectangle(image,
(0,0), (640, 40), (245, 117, 16),
-1)
```



```
        cv2.putText(image, '
'.join(sentence), (3,30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2, cv2.LINE_AA)

        # Show to screen
        cv2.imshow('OpenCV Feed',
image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF
== ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

V. CONCLUSION

We conclude that LSTM neural network can be used as a deep learning for a real-time vision-based Sign Language recognition for deaf and dumb people based on ASL. However, pre-training has to be performed with a larger dataset in order to show increase in accuracy. We achieved final accuracy of 66.66% on our dataset. For user-dependent, the user will give a set of images to the model of training, so it becomes familiar with the user. This way the model will perform well for a particular user.

VI. REFERENCES

[1] Prof. Shirbhate,Radha. S., Mr. Shinde,Vedant. D., Ms. Metkari,Sanam. A., Ms. Borkar,Pooja. U., and Ms.Khandge,Mayuri. A. (2020). Sign language Recognition Using Machine Learning Algorithm. International Research Journal of Engineering and Technology.Vol**7**(3):(pp. 2122-2125).

[2] www.geeksforgeeks.org/python-facial-and-hand-recognition-using-mediapipe-holistic/

[3] www.learnopencv.com

[4] https://www.alpha-quantum.com/blog/long-short-term-memory-lstm-with-python/long-short-term-memory-lstm-with-python

[5] Varsha, M., and Nair, C. S. (2021). Indian sign language gesture recognition using deep convolutional neural network. 2021 IEEE 8th International Conference on Smart Computing and Communication(ICSCC).10.1109/ICSCC51209.2021.9528246.

[6] Kodandaram,Swatik. Ram., Kumar,N. Pavan., and Sunil G L. (2021). Sign Language Recognition. Turkish Journal of Computer and Mathematics Education.Vol**12**(14):(pp. 994-1009).

[7] Dixit, K., and Jalal, A. S. (2013). Automatic sign language recognition system. 2013 3rd IEEE International Advance Computing Conference (IACC).10.1109/iadcc.2013.6514343.

[8] Kang, B., Tripathi, S., and Nguyen, T. (2015). Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. 3rd IAPR Asian Conference on Pattern Recognition; Kuala Lumpur, Malaysia. 10.1109/acpr.2015.7486481.

[9] Sahoo, Ashok. K., Mishra, Gouri. Sankar., and Ravulakollu, Kiran Kumar. (2014). Sign language recognition: State of the art. ARPN Journal of Engineering and Applied Sciences. Vol**9**(2): (pp. 116-134).

[10] Bragg, D., Koller, O., Bellard, M., Berke, L., Boudreault, P., Braffort, A., and et al. (2019). Sign language recognition, generation, and translation: an interdisciplinary perspective. The 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'19). Association for Computing Machinery, New York, NY, USA. (pp. 16-31).

[11] Rastgoo, Razieh., Kiani, Kourosh., and Escalera, Sergio. (2021). Sign language recognition: a deep survey.

Expert Systems with Applications. Vol**164**(113794). 10.1016/j.eswa.2020.113794.

[12] Zimmermann, C., and Brox, T. (2017). Learning to estimate 3D hand pose from single RGB images. 2017 IEEE International Conference on Computer Vision (ICCV). (pp. 4913-4921).

[13] Sahoo, A. (2014). Indian sign language recognition using neural networks and kNN classifiers. Journal of Engineering and Applied Sciences. Vol**9**: (pp. 1255-1259).

[14] Kawulok, M. (2008). Dynamic Skin Detection in Color Images for Sign Language Recognition. In: Elmoataz, A., Lezoray, O., Nouboud, F., Mammass, D. (eds) Image and Signal Processing. ICISP 2008. Lecture Notes in Computer Science, vol 5099. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-69905-7_13.

[15] Utaminingrum, Fitri., Somawirata, I. Komang., and Naviri, Gilbert. Dany. (2019). Alphabet sign language recognition using K-nearest neighbor optimization. JCP. Vol**14**(1): (pp. 63-70).

[16] Thakur, Amrita., Budhathoki, Pujan.,Upreti, Sarmila.,Shrestha, Shirish., and Shakya, Subarna. (2020).Real Time Sign Language Recognition and Speech Generation. Journal of Innovative Image Processing (JIIP). Vol**2**(2):(pp. 65-76).