Sign Language recognition and Motion recognition

PROJECT SUPERVISOR
DR Farrukh Hassan

PROJECT CO-SUPERVISOR
Engr-Shahbaz Siddidui

PROJECT TEAM

Syed Azmat Ali Abedi	K180248
Aftab Ahmed	K180223
Vikash kumar	K180136

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

June 2022

Project Supervisor	DR Farrukh Hassan
Project Team	Syed Azmat Ali Abedi K180248 Aftab Ahmed K180223 Vikash kumar K180136
Submission Date	

Mr. Supervisor Name DR Farrukh hassan

Supervisor ______________________

Mr. Co-Supervisor Name Engr-Shahbaz SIDDIQUI

Co- Supervisor ________________________

Dr. Zulfiqar Ali Memon _____________________________

Head of Department

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

Acknowledgement

This dissertation is my unique work, and it has not yet been submitted for this or any other award. Any parts borrowed from my previous work, or the work of others are quoted and approved with a clear reference to the actual source. Author, page, and source (s). Non-original images are also quoted. I know the situation. Failure to do so will be construed as cheating and will result in failure in this course.

Abstract

Deaf and hard of hearing people use sign language to exchange information within their own community but also with other people even outside their group. Computer recognition of sign language begins with the acquisition of sign gestures and continues through the creation of text and voice. Sign motions can be classified into two parts: static and dynamic. Although static gesture recognition is easier to use than dynamic gesture recognition, both recognition methods are critical to the human community`s ability to communicate. So we tried different strategy to solve this problem. When it comes to recognizing the static sign, our proposed method makes advantage of the MediaPipe Holistic. The MediaPipe is useful for obtaining the most important features from the hand. In the previous research studies, the CNN (Convolution Neural Network) was used to recognize the sign; nevertheless, CNN was unable to function. As a result, we did away with the CNN`s convolution portion and instead used the MediaPipe and the ANN to recognize the Static Sign. The collection contains 63,300 photographs of the hand, arranged alphabetically from letter A to letter Z, then we tried 2nd strategy Our proposed design makes use of the MediaPipe Holistic for receiving input to recognize the motion of a deaf individual. The MediaPipe layer captures a series of frames of the poses and passes them along to the LSTM layers. The LSTM contains three gates, one of which is a "forget" gate, which aids in the storage of critical information and the forgetting of unimportant information. "Update Gate" is a tool that assists in updating information that has been forgotten as well as updating information that has been stored. The information will be passed to the next LSTM cell through the "output gate." We feed the LSTM output into a fully connected neural network (ANN) and output the action that was predicted to be the most effective 3rd strategy Our second proposed

design is more sophisticated than the LSTM architecture. We utilize the same MediaPipe Holistic that we used for identifying the motion of the deaf person to collect the input for this project. The MediaPipe captures a sequence of frames of the poses and passes them on to the GRU layers for further processing. GRU contains two gates: the "Update" gate, which assists in updating past information together with the current input, and the "Reset Gate," which assists in resetting information that has been forgotten and adding new information. After that, the information is passed on to the next GRU cell. We feed the GRU output into a fully connected neural network (ANN) and output the action that was predicted to be the most effective and the solution we found Our second proposed design is more sophisticated than the LSTM architecture. We utilize the same MediaPipe Holistic that we used for identifying the motion of the deaf person to collect the input for this project. The MediaPipe captures a sequence of frames of the poses and passes them on to the GRU layers for further processing. GRU contains two gates: the "Update" gate, which assists in updating past information together with the current input, and the "Reset Gate," which assists in resetting information that has been forgotten and adding new information. After that, the information is passed on to the next GRU cell. We feed the GRU output into a fully connected neural network (ANN) and output the action that was predicted to be the most effective

content

List of Figure

Sign Language recognition and Motion recognition

DR Farrukh Hassan
farrukh.hassan@nu.edu.pk
Assistant professor Fast national university

Engr-Shahbaz Siddidui
Shahbaz.siddiui@nu.edu.pk
Assistant professor Fast national university

Syed Azmat Ali Abedi k180248@nu.edu.pk
AftabAhmedk180223@nu.edu.pk
Vikash kumar k180136@nu.edu.pk

Introduction

In recent years, the creation of an application for the deaf and hard of hearing population has risen to the top of the list of priorities for the organization [1]. Technology developments and educational opportunities that open new vistas of potential are changing the globe at an increasingly rapid pace. As a result, it is becoming increasingly important for everyone to keep one step ahead of their competitors in this race. This programmed is being developed in order to aid the deaf and dumb people in garnering attention, as well as to provide them with a common platform on which to communicate and express their ideas with the rest of the world [12]. Furthermore, the sign language is classified into two categories: static sign language and motion recognition [3], with the first of these categories can be define as the sign of the alphabets. Like letter 'A' has own symbol, in this way there are 26 letters, and each letter has own symbol. So we have 62375 images. Previous research paper tried to classify these symbols using the Machine Learning's models and computer vision [6]. But their approaches work well when the sign is giving in static format when the static signs are giving in the continuous video format the model could not perform well. For solving this issue we come up with new and latest method that help us to extract the features of the hand and reduce the complexity that is mediapipe [4] moreover mediapipe helps to remove the convolution part of the CNN [7] after extraction the features and save that features into the csv file We employ a simple neural network (ANN) for predicting the static signs [5]. Motion detection is very hard process because, there are large number of parameters that are needed to detect the motion, moreover, detecting a motion is the sequential process means we need the sequence of input or the sequence of actions like if someone is walking so just one step is not enough to detect the actions. To tackle this problem, we use the sequential model GRU [8]. previous research paper solve the motion detection problem using the CNN+LSTM [8]. The result was good but lot of computation is required to the convolution part so we need advanced technique that helps use to remove the convolution part and direct give the results to the LSTM. So we use the MediaPipe for detecting the hands, face and body [4]. For motion detection we have created our own dataset by taking the survey that what are the most fundamental action that are used all around the world. So we got the 10 actions that are international and most fundamental. We train our model on this dataset, but our results were not good enough. So, we use the more advanced model of LSTM that is GRU. After applying GRU we got the good result but from the previous one. But when we are giving the motions to the model on the run time. It is still not performing well. To improve the performance of GRU we add the attention part of the transformer [9], that helps us a lot. Our final model mean GRU+ attention preformed excellent. So we have built our application on the Flutter and deploy our model on the Flutter.

Literature Review

There are two categories of people who exist in our society: those who are considered to be "normal," and those who are thought to be "special," such as persons who are deaf or dumb. Normal people are those who are able to hear and speak normally. They are a part of our society, but they do not play a very significant role in the way our society. There is not a single smartphone application that can help dump people use in their day-to-day lives. Having stated that, we have found a variety of options that will help to develop the mobile application, including one of the research articles that tries to tackle this problem with a convolution neural network (CNN) [7]. These solutions will help us build the mobile application. CNN does an excellent job when we display the image of the sign one at a time; nevertheless, the convolution did a poor job when it was tested in real time and given a series of demanding tasks. This occurred because the convolutional technique did not quickly extract the features from the image. Furthermore after applying the convolution we are losing the important information, means individual feature of the hand and fingure. This was the root cause of the problem. You Only Look Once (YOLO), an acronym that stands for "you only look once," is the second solution that we found to address this difficulty [13]. [YOLO] stands for "you only look once." It is an improved version of CNN, and this strategy works even when applied to real-time coverage of events. It is advised that we use YOLO when we are addressing the problem of picture classification in a single image or the problem like image caption [9], and both YOLO and CNN functioned very well on the 2D image. YOLO first creates the bounding boxes and then assigns a label to the image it generates. We came up with the most recent technique, which is now known as the Media pipe [4]. With the assistance of Media Pipe, we will be able to get knowledge regarding the features or landmarks. This includes landmarks on the left hand, landmarks on the right hand, and landmarks on the face. In addition to this, the Media Pipe is also hard at work on the 3D image; for instance, if we are talking about the landmarks on the left side of the screen, the Media Pipe is working on those landmarks. The two-dimensional and three-dimensional images will each lose 63 characteristics as a result of this operation. If we give the artificial neural network (ANN) these features, it will be able to recognise the sign in a way that is both straightforward and efficient [1]. Incredible outcomes were achieved by the combination of using Media pipe and ANN. Nevertheless, this is not the end of the challenge; we still need to determine the action, which may be something like "thank you" or "drink," amongst other things. When it comes to action recognition, we need to make use of a sequential model that is capable of accepting input in a sequential fashion. After that, we came across the Recurrent neural network, also known as an RNN [11], which is capable of first anticipating the output by using a sequence of input.

The RNN has the disadvantage of not being able to store the longest sequence and long-term dependencies for the purpose of output prediction. This is a significant limitation of the RNN. Because the RNN has only a single online gate, it is impossible to select the data that should be forgotten and the data that should be saved. This is because of the fact that the RNN only owns a single online gate. Because of this, we came up with an extra technique that we now refer to as the longest sequence short term memory (LSTM) [6]. The RNN has been refined and modernised to produce this new algorithm. The LSTM method is capable of preserving information as well as long-term dependencies, in addition to having the potential to hold the longest sequence that can possibly be stored. INPUT, FORGET, and OUTPUT are the names given to the three gates that make up the LSTM in its most fundamental version [8]. The outcome that we are obtaining is still not good, despite the fact that we are getting the feature from the media pipe and passing it on to the LSTM. However, this takes up a substantial amount of time during training, and because we have more than ten classes, we make use of the GRU, which is an upgraded version of the LSTM and only has two gates. This allows us to more efficiently store and retrieve information. RESET and UPDATING the gate is required. Since this is the case, GRU is rapidly training the model. The GRU is not capable of acquiring all of the relevant information. despite the fact that the required amount of processing power and training time have both been steadily lowering over the years. We were successful in locating a research publication that provided an explanation of the attention mechanism [9], which is a mechanism that is also utilized in transformers. Because of this, we were able to resolve the issue with the machine translation. The attention mechanism considers all aspects of the competition and provides information that is comprehensive in its significance. Therefore, after carefully reviewing a large number of research papers, I arrived at the single solution that GRU plus attention is doing an excellent job for recognizing the activity of a dump person.

Technical Reviews

IDE : Jupyter Notebook

The entire system is written in Python 3 utilizing the Jupyter Notebook integrated development environment. IDEs such as this one are used because they are highly interactive, data exploration and visualization are the best available among all IDEs, the computational output can be documented for later use, code can be written in separate "cells," which means iteration time is reduced, and computational output can be documented for later use are all advantages of using an IDE such as this one.

Libraries/ Modules

- **Numpy**: numpy helps to perform the data manipulation and computation in series
- **Panda:** This library is built in on NumPy which is used in my implementation to perform data structure and data analysis.
- **Matplotlib :** All of the visualisation of my data set is carried out using this library. It plays a crucial role in giving me insight into my data using graph , bar chart etc.
- **Scikit-learn:** This library provides easy to implement It is also used in feature selection, train test split, confusion matrix and train test and split.
- **Tensorflow:** This library help us to use the deep learning model.

Design

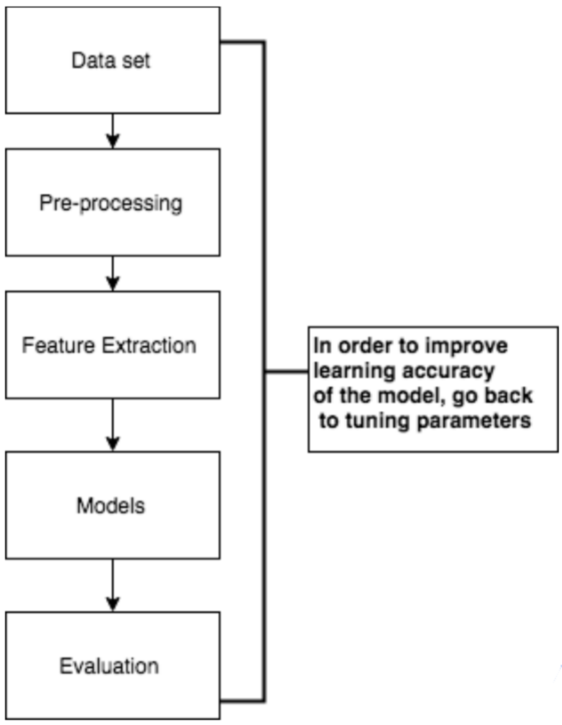

Fig.1

This was the first concept that sprang to mind while I was thinking about my sign language recogniton system. It is the first version, which is the most fundamental, and it is modelled after the standard method used in detection systems. This design served as a starting point for me to build upon during the subsequent design phase. This architecture addresses the vast majority of critical procedures in a sign language recognition system; however, it does not include the feature. selection phase, which I discovered after it was completed. During the design phase, I gained a great deal of understanding about sign language detection, which is reflected in this graphic. As a result, no pre-processing strategy was chosen, nor were the types of classifiers to be used, nor the methods of feature extraction and selection. Aside from that, this system does not provide specific processes for implementing sign language detection in Python. However, this phase was extremely important in the development of the structure of my final sign language detection system, and I built on what I learnt from trail and test throughout the process. The existence of this framework aided me in the development of my final design for my sign language detection system.

Dataset

We are working with two separate datasets. The first one is for recognizing the static sign, and the second one is for recognizing the action. We analyses all 62375 images that were captured for the static sign in order to determine the characteristics of the hand. And then save the document as a.csv file. In order to construct the dataset used for action recognition. We carried out the survey at the school for exceptional children. They provide us with the most common action that are utilized to the greatest extent in this world, such as hello, tghank you, stand up, and seat down. On this basis, we were able to collect all ten actions necessary actions . After that, we use media pipe to assist in the construction of the dataset for the action recognition. For example, for the thank you action, we carried it out thirty times before storing its sequence in the array. therefore, in this regard

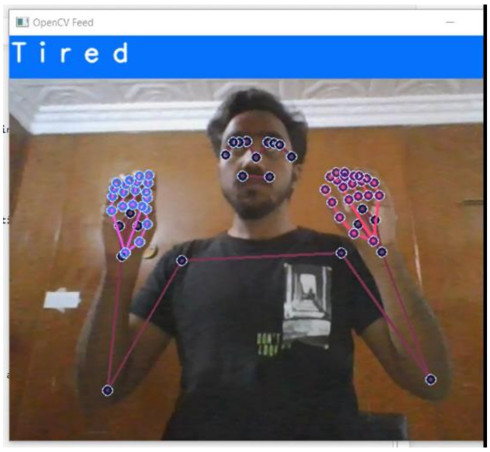

Fig.2

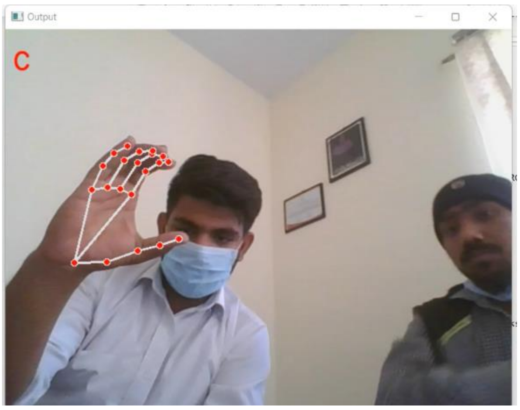

Fig.3

```
In [4]:  print(x_train.shape)
         print(y_train.shape)

(62375, 63)
(62375,)
```

Fig.4

```
In [11]:  #action =np.array(['stop'])
          actions = np.array(['call', 'cook','drink','dustbin','help','hungry','no_action','pain','sitdown','standup','thrusty','tired

In [12]:  # Path for exported data, numpy arrays
          DATA_PATH = os.path.join('MP_Data_data')

          # Actions that we try to detect
          actions = np.array(['call', 'cook','drink','dustbin','help','hungry','no_action','pain','sitdown','standup','thrusty','tired
          #action=np.array(['stop'])
          # Thirty videos worth of data
          no_sequences = 30
          print(len(actions))
          # Videos are going to be 30 frames in length
          sequence_length = 30
```

Fig.5

So we have these 13 total classes for the action recognition.

Methods

Sign Language Static Sign

When it comes to recognizing the static sign, our proposed method makes advantage of the MediaPipe Holistic. The MediaPipe is useful for obtaining the most important features from the hand. In the previous research studies, the CNN (Convolution Neural Network) was used to recognize the sign; nevertheless, CNN was unable to function. As a result, we did away with the CNN`s convolution portion and instead used the MediaPipe and the ANN to recognize the Static Sign. The collection contains 63,300 photographs of the hand, arranged alphabetically from letter A to letter Z.

Sign Language Static Sign When it comes to recognizing the static sign, our proposed method makes advantage of the MediaPipe Holistic. The MediaPipe is useful for obtaining the most important features from the hand. In the previous research studies, the CNN (Convolution Neural Network) was used to recognize the sign; nevertheless, CNN was unable to function. As a result, we did away with the CNN`s convolution portion and instead used the MediaPipe and the ANN to recognize the Static Sign. The collection contains 63,300 photographs of the hand, arranged alphabetically from letter A to letter Z.

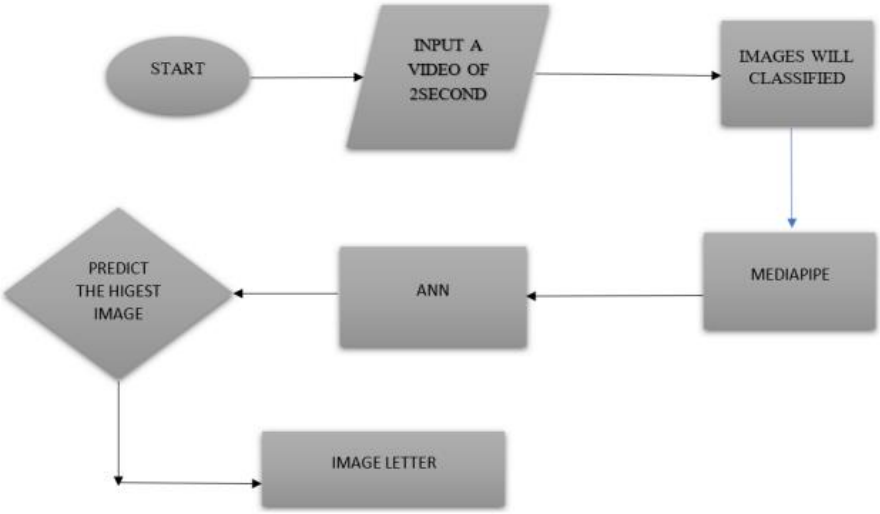

Fig.6

Action Recognition Gesture Recognition Using LSTM

Our proposed design makes use of the MediaPipe Holistic for receiving input to recognize the motion of a deaf individual. The MediaPipe layer captures a series of frames of the poses and passes them along to the LSTM layers. The LSTM contains three gates, one of which is a "forget" gate, which aids in the storage of critical information and the forgetting of unimportant information. "Update Gate" is a tool that assists in updating information that has been forgotten as well as updating information that has been stored. The information will be passed to the next LSTM cell through the "output gate." We feed the LSTM output into a fully connected neural network (ANN) and output the action that was predicted to be the most effective

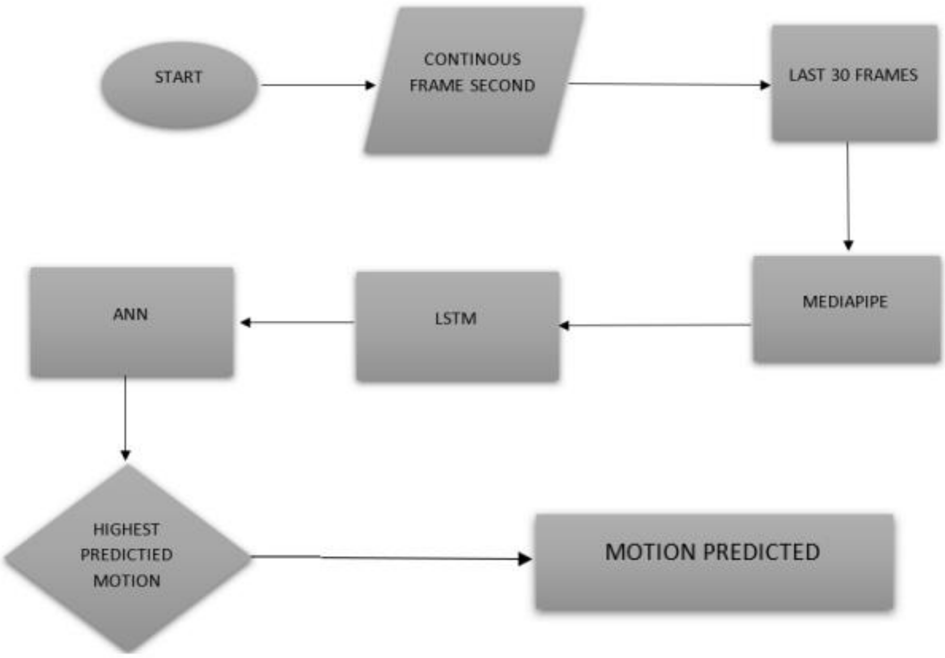

Fig.7

Gesture Recognition using GRU

Our second proposed design is more sophisticated than the LSTM architecture. We utilize the same MediaPipe Holistic that we used for identifying the motion of the deaf person to collect the input for this project. The MediaPipe captures a sequence of frames of the poses and passes them on to the GRU layers for further processing. GRU contains two gates: the "Update" gate, which assists in updating past information together with the current input, and the "Reset Gate," which assists in resetting information that has been forgotten and adding new information. After that, the information is passed on to the next GRU cell. We feed the GRU output into a fully connected neural network (ANN) and output the action that was predicted to be the most effective.

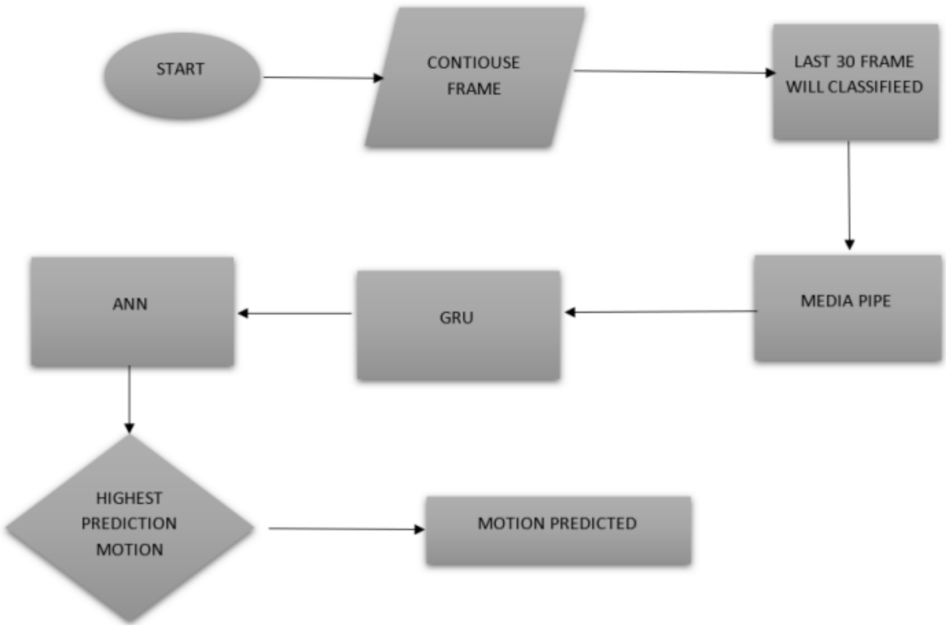

Fig.8

Gesture Recognition using GRU+ Attention part

In comparison to a simple GRU, our third proposed architecture is more advanced. We utilize the same MediaPipe Holistic that we used for recognizing the motion of the deaf individual to collect the input for this task. The MediaPipe captures a sequence of frames of the poses and passes them on to the GRU layers for further processing. After going through the succession of GRU-provided material, we pass it along to the Attention Part. Attention model: Focus specifically on specific components. For example. The human brain first focuses on a particular aspect image at high resolution and then see the surrounding area at low resolution. However, as the brain begins to understand the image, it adjusts its focus to fully understand all aspects. The purpose of using the attention model is to reduce larger and more complex tasks to smaller and more manageable tasks. Initially, attention models were used to improve computer vision and encoder decoder-based neural machine translation systems. In the end following the attention, we feed the output into a fully connected neural network (ANN) and output the action that was anticipated to be the most effective

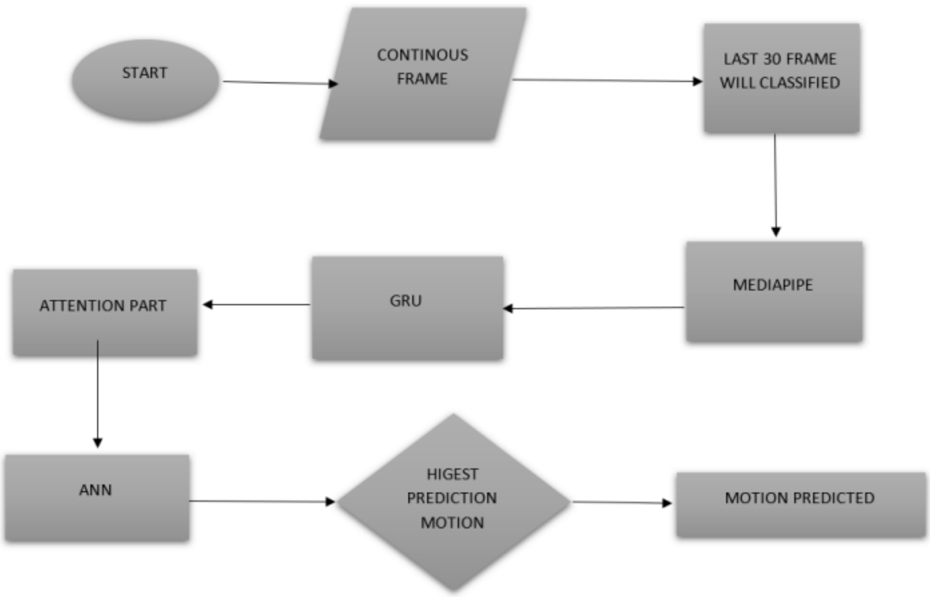

Fig.9

Implementation

Static sign recognition

```
import pandas as pd
import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import LeakyReLU
import matplotlib.pyplot as plt
import mediapipe as mp
import cv2
```

Fig.10

We load the libraries that are required for the rest of the implementation

Data understanding

```
In [4]: df = pd.read_csv("asl_alphabet_train.csv",sep=',',header=None)

        y_train = df.values[:,63]
        x_train = df.values[:,0:63]
```

```
In [5]: df
Out[5]:
```

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58
0	0.469191	0.575121	0.0	0.570774	0.501089	-0.009906	0.634131	0.375068	-0.019063	0.648172	...	0.376318	0.311486	-0.099011	0.394354	0.383876
1	0.461542	0.597693	0.0	0.573268	0.487783	0.027516	0.631623	0.368269	0.026500	0.646930	...	0.376780	0.317405	-0.086910	0.394635	0.389508
2	0.454547	0.591447	0.0	0.568208	0.499029	0.010541	0.630960	0.379245	0.002211	0.647045	...	0.377393	0.314732	-0.098095	0.389590	0.383028
3	0.471271	0.590186	0.0	0.575850	0.498384	0.002622	0.637386	0.376007	-0.005789	0.647484	...	0.368133	0.326961	-0.132763	0.391314	0.389844
4	0.471458	0.582966	0.0	0.579936	0.497440	0.011598	0.642250	0.376807	0.004018	0.655829	...	0.387560	0.325203	-0.108127	0.403230	0.395566
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
62370	0.437118	0.417013	0.0	0.568735	0.394053	0.014265	0.671439	0.378385	0.022551	0.724034	...	0.594838	0.215650	-0.041006	0.572325	0.253138
62371	0.453403	0.401617	0.0	0.573000	0.415059	-0.009460	0.679467	0.398600	-0.011516	0.730032	...	0.592851	0.204943	-0.051812	0.565064	0.246661
62372	0.442922	0.405172	0.0	0.575888	0.408325	-0.001215	0.680394	0.395461	-0.001411	0.727995	...	0.585428	0.198660	-0.043309	0.575574	0.238968
62373	0.456261	0.400682	0.0	0.591224	0.406523	-0.003146	0.694999	0.387991	-0.006214	0.737967	...	0.579323	0.191772	-0.060861	0.567600	0.239995
62374	0.437812	0.411544	0.0	0.492444	0.294803	0.033209	0.564672	0.235650	0.046498	0.636171	...	0.733303	0.261063	-0.045855	0.752099	0.230635

62375 rows × 64 columns

Fig.11

When I open the file. I see that all the features are normalized, therefor I don't need to use standard scaler or any other technique. There are total of 62375 entries. Each sets have fifty-eight dependent variable and one dependent variable;

Train

```
model = keras.Sequential([
                keras.layers.Input((63,)),
                keras.layers.Dense(100,activation=keras.layers.LeakyReLU(alpha=0.05)),
                keras.layers.Dense(75,activation=keras.layers.LeakyReLU(alpha=0.05)),
                keras.layers.Dense(50,activation=keras.layers.LeakyReLU(alpha=0.005)),
                keras.layers.Dense(26,activation="softmax")
])
```

```


model.compile(
    optimizer = "adam",
    loss="sparse_categorical_crossentropy",
    metrics = ['accuracy']
)

model.fit(
    x_train,y_train,epochs=100
)
```

Fig.12

We define the equational model for the static sign. In which we added the three dense layer and having the leakyrelu activation function. Moreover having the adam optimizer.

Modeling

LSTM

We define the LSTM model in which have three LSTM layer and three dense layer. In the hidden layer I used the relu activation function. And in the last layer I used the softmx activation function because we have more than one classes

```
LSTM_model = Sequential()
LSTM_model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,258)))
LSTM_model.add(LSTM(128, return_sequences=True, activation='relu'))
LSTM_model.add(LSTM(64, return_sequences=False, activation='relu'))
LSTM_model.add(Dense(64, activation='relu'))
LSTM_model.add(Dense(32, activation='relu'))
LSTM_model.add(Dense(actions.shape[0], activation='softmax'))
LSTM_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            82688
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 13)                429
=================================================================
Total params: 237,581
Trainable params: 237,581
Non-trainable params: 0
_________________________________________________________________
```

Fig.12

GRU

The GRU model is defined as having three GRU layers and three dense layers. I utilised the relu activation function in the hidden layer. Because we have several classes, I utilised the softmax activation method in the final layer.

```
GRU_model = Sequential()
GRU_model.add(GRU(64, return_sequences=True, activation='relu', input_shape=(30,258)))
GRU_model.add(GRU(128, return_sequences=True, activation='relu'))
GRU_model.add(GRU(64, return_sequences=False, activation='relu'))
GRU_model.add(Dense(64, activation='relu'))
GRU_model.add(Dense(32, activation='relu'))
GRU_model.add(Dense(actions.shape[0], activation='softmax'))
GRU_model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 30, 64)            62208
_________________________________________________________________
gru_1 (GRU)                  (None, 30, 128)           74496
_________________________________________________________________
gru_2 (GRU)                  (None, 64)                37248
_________________________________________________________________
dense_12 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_13 (Dense)             (None, 32)                2080
```

Fig.13

GRU_Attention

The GRU model is defined as having three GRU layers and three dense layers. I utilised the relu activation function in the hidden layer. Then added the Attention Part.

```
attention_72 = Sequential()
attention_72.add(GRU(64, return_sequences=True, activation='relu', input_shape=(30,258)))
attention_72.add(GRU(128, return_sequences=True, activation='relu'))
attention_72.add(GRU(64, return_sequences=True, activation='relu'))
attention_72.add(Attention())
attention_72.add(Dense(64, activation='relu'))
attention_72.add(Dense(32, activation='relu'))
attention_72.add(Dense(actions.shape[0], activation='softmax'))
attention_72.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 30, 64)            62208
_________________________________________________________________
gru_1 (GRU)                  (None, 30, 128)           74496
_________________________________________________________________
gru_2 (GRU)                  (None, 30, 64)            37248
_________________________________________________________________
attention (Attention)        (None, 128)               20480
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_5 (Dense)              (None, 13)                429
=================================================================
```

Fig.14

Attention model: Focus specifically on specific components.

For example. The human brain first focuses on a particular aspect image at high resolution and then see the surrounding area at low resolution. However, as the brain begins to understand the image, it adjusts its focus to fully understand all aspects. The purpose of using the attention model is to reduce larger and more complex tasks to smaller and more manageable tasks. Initially, attention models were used to improve computer vision and encoder decoder-based neural machine translation systems. In the end following the attention, we feed the output into a fully connected neural network (ANN) and output the action that was anticipated to be the most effective .in the end layer I utilised the softmax activation method because of different classes.

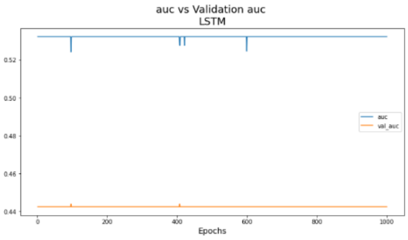

Fig.15

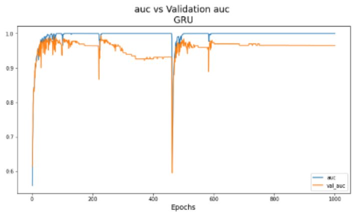

Fig.16

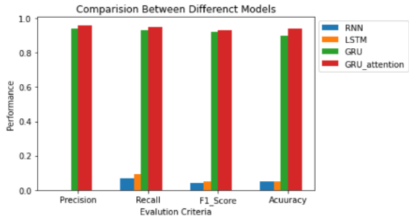

Fig.17

Testing and Evaluation

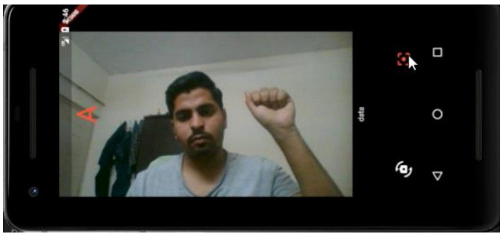

Fig.18

After training, several models including LSTM, GRU, and GRU with attention compose a portion of the transformer. LSTM gave us very poor result you can see the graph and we use the updated mode of LSTM that is GRU and that is much better than LSTM and finally we add the attention part in GRU. We have arrived at the conclusion that GRU plus attention performed the best when compared to other models. So now we are going to deploy our final model is that is based on GRU+attention on the futter. You can the static sign model has been deployed on the mobile.

Conclusion

This work addresses the fundamental problem of sign language recognition in order to bridge the communication barriers between hearing and vocally impaired people, and the rest of the society. Previous works concerned with this problem either jointly considered both spatial and temporal information or relied mainly on the temporal information. To tackle this issue, this paper proposes we come up with the single mobile application that translate the action of the deaf person and also recognition the single letter of alphabets.

References

Appendices B: Full Code