Project Report

Application Service for Classifying American Sign Language

Table of Contents

1. Roles and responsibilities

Project participants are:

1. Gaurav Goyal
2. Sneha Kumari
3. Teja Indrakanti
4. Thomas Wilkinson

2. Team goals and Application objectives

This project is an application service for classifying American Sign Language. The whole project is divided into two parts, Part 1 and part 2.

Objectives of part 1 of the project are:

To develop a mobile application with the listed functionalities

- The user is shown a video of a gesture.
- The user can replay the video at least 3 times.
- Upon clicking the “PRACTICE” button, the user can capture his or her own video through the smartphone’s front camera for a period of at most 5 seconds.
- The videos are uploaded to a server.

The mobile application should have three screens:

- **Screen 1:** A drop-down menu of 20 different gestures will be shown on this screen. Once a single gesture is selected, the user will be taken to Screen 2. Gesture list: {buy, house, fun, hope, arrive, really, read, lip, mouth, some, communicate, write, create, pretend, sister, man, one, drive, perfect, mother}
- **Screen 2:** The video of an expert performing the gesture will be shown on this screen. The video will have to be downloaded on the phone from the Signing Savvy (https://www.signingsavvy.com/) ASL gesture repository. Screen 2 will have another button that says “PRACTICE”. Once this button is pressed, the user will be taken to Screen 3.
- **Screen 3:** In this screen, the camera interface will be opened for the user to record the practice gesture. The video will be captured for **5 seconds**, and the video will be saved with the following filename format: "GESTURE_PRACTICE_(practice number)_USERLASTNAME.mp4". Screen 3 will have another button that says "UPLOAD". Once this button is pressed, the user will be able to upload the gesture to a local server. Moreover, clicking this button will take the user back to Screen 1.
- The local server is created with the help of the Flask server. Here Flask is used as a local server

After having these functionalities in the app, this app is used like this:

1. Each team member will perform each gesture at least three times and upload three correct versions of each gesture for a total of 60 videos each.
2. These recorded videos are saved in the local system with the help of Flask server

Objectives of part 2 of the project are:

To develop an online RESTful application service in Flask that accepts skeletal humans pose key points of an ASL sign video and return the label of the sign as a JSON response. The key points are generated using TensorFlow’s PoseNet.

After this part, using these key points, develop four (4) machine learning (ML) models (one by each team member) that can classify the listed ASL signs. There are 6 gestures are used,

1. buy - https://www.signingsavvy.com/search/buy
2. fun - https://www.signingsavvy.com/search/fun,
3. hope - https://www.signingsavvy.com/search/hope,
4. really - https://www.signingsavvy.com/search/really,
5. communicate - https://www.signingsavvy.com/search/communicate,
6. mother - https://www.signingsavvy.com/search/mother

Assumptions

1. In the project we used API 29 and Flask is used as a local server.

Resources Used

- Android Studio
- Flask Server
- TensorFlow's PoseNet
- Python 3.6.9 or above
- OpenCV for Python
- Node.js
- Python predefined libraries (Sklearn for ML models)

Task distribution

The initial challenge for this project was obtaining data from the videos in a useful format. Utilizing Tensorflow's posenet we were able to develop a rough approximation of a person's pose in each frame of a video clip. Posenet developed a skeleton figure with 17 key points representing the overall body position. A result is a JSON dictionary which contains the x and y coordinates of each key point in the frame as well as a confidence level for the skeletal model. This JSON file simplified the gesture performed in a video to a series of two-dimension data points. Those data points were then used as inputs to our Machine Learning models.

For this project, we have used a Recurrent Neural network to perform multivariate classification. A Support Vector Machine classifier and Random Forest classifier were also both developed. Their accuracies were significantly lower than the neural network, thus the neural network was the preferred model in this case.

The models for classification were hosted on a Flask based server. Each model was serialized into .pkl files via python's pickle module. The data from a video is converted to the JSON format as previously mentioned. It was then transferred to the flask server by an Http post request. The Data was then run through the various models which made a prediction on what gesture was being performed. Finally, the accuracy of each model was determined based on known labels for the data.

Data generation:

For the videos we generated in the first part of the project, we generate frames by using the function provided in this project. Since we are going to use LSTM, the time series should be of fixed length to feed the model. So irrespective of the number of frames for each video, they are squeezed by selecting 30 equally spaced frames. The key points are generated for these frames using the Pose net library as mentioned. Once we have the key points, we are going to generate a csv from the data in the key points. The values in the csv include the distances of elbows, shoulders and wrists with respect to nose for each frame. This ensures the lateral movement of a person in the frame does not impact the model and also reduces the input features to optimum level. All the data in the CSVs is concatenated along with their labels.

Training the Model:

Once we have the data ready in the CSV, we divide the data in the csv into a feature_set containing sets of information for each video. So, each set in the feature_set consists of 30 values belonging to a single video with its label. Since LSTM only accepts 3-dimensional input, the data is reshaped to fit the LSTM model using numpy libraries. The feature_set dimensions at this point will be number of videos * 30 * 6. Now we shuffle the data using the shuffle function from scipy to remove the linearity in the input data which has all videos belonging to one class grouped together.

Once we had the input data and labels ready, we started to build the model. We used Keras libraries to build our models. The idea we had is to create a keras model with LSTM layers stacked with dense layers with outer layer consisting of the same number of the neurons as the number of labels. Since the labels consist of categorical data with multiple values, we used label encoder and one hot encoder to convert the one row of categorical data into 6 rows. These encoders are also stored since they will be used to decode the prediction in future. Our models were built on different architecture with different layers, different activation functions, different cost functions and different optimizers. We trained our models with a batch size of 10 and 200 epochs and validation split of our choice between 0.1 and 0.25.

Combinations of LSTM Dense networks:

Model-1: 2 LSTM Layers(150,100), 5 Dense Layers, Adagrad optimizer, Leaky ReLU activation, Categorical_Crossentropy loss function, Drop Out= 0.1

Model-2: 2 LSTM Layers(100,100), 4 Dense Layers, Adam optimizer, ReLU activation, Categorical_Crossentropy loss function, Drop Out= 0.1

Model-3: 2 LSTM Layers (150,100), 6 Dense Layers, SeLU activation function, Adam optimizer, Categorical_Crossentropy as loss function, Drop Out = 0.1

Model-4: 2 LSTM Layers(100,50), 4 Dense Layers, Adam optimizer, ReLU activation, Logcosh loss function, Drop Out= 0.1

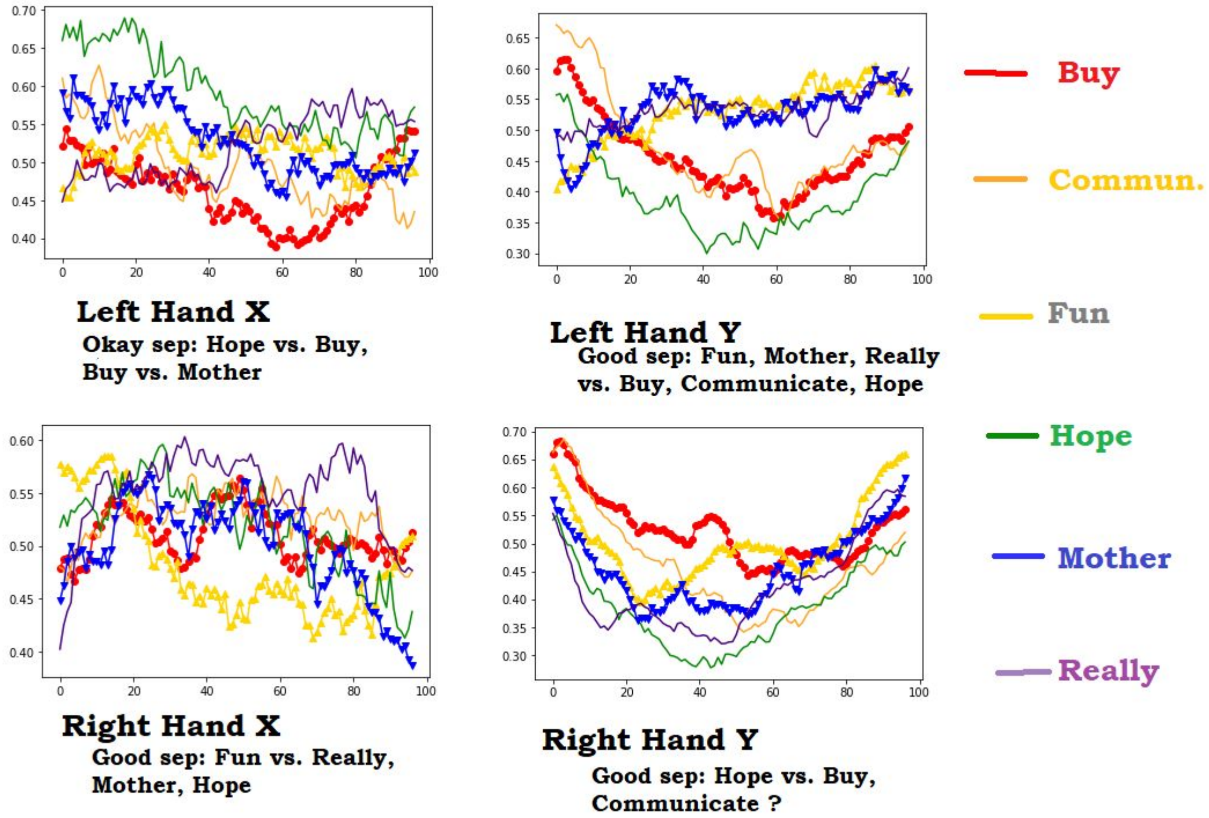
Challenges

As part of the whole exercise, several challenges were encountered that acted as different decision points leading to the current shape of the model. Some of the challenges are enumerated below:

1. **Right data**: One of the most preliminary questions was related to the correctness and the distribution of the data available. The training data provided contained a lot of variations in terms of the duration of the recording. For example, the fastest clip lasted less than a second and the longest one seemed to last for over 3 seconds. This data needed to be normalized in order for comparing equivalent moments during the gesture expression. Several experiments were conducted on normalizing and aggregating the data and inspecting it visually in a graph in order to observe patterns and compare the similarities and the variations.
2. **Right parameters**: Each converted keypoint, as described above, contains 17 different independent parameters, which are coordinates of 17 different parts of the body. Correctly chosen parameters will lead to more accurate models. It is impractical to try with multiple combinations and come up with the most accurate result, owing to a large number of combinations. Therefore, reasonable assumptions were made and the most appropriate coordinates were chosen. For example, in a video displaying hand-based gestures, the location coordinates of the abdomen might not be applicable.
3. **Right Model**: Once the above two are sorted, the most important decision that needed to be made was on choosing the model itself. The objective is to propose a model that accurately predicts the gesture. Some of the experiments included some SVM based analyses and some more based on neural networks. During the experiments, neural network-based models turned out to be accurate at predicting the gesture based on the key points.

Data visualization:

For this project the data was plotted and visualized to look for possible features to use in classification. Some of the plots generated follow:



Left Hand X

Okay sep: Hope vs. Buy, Buy vs. Mother

Left Hand Y

Good sep: Fun, Mother, Really vs. Buy, Communicate, Hope

Right Hand X

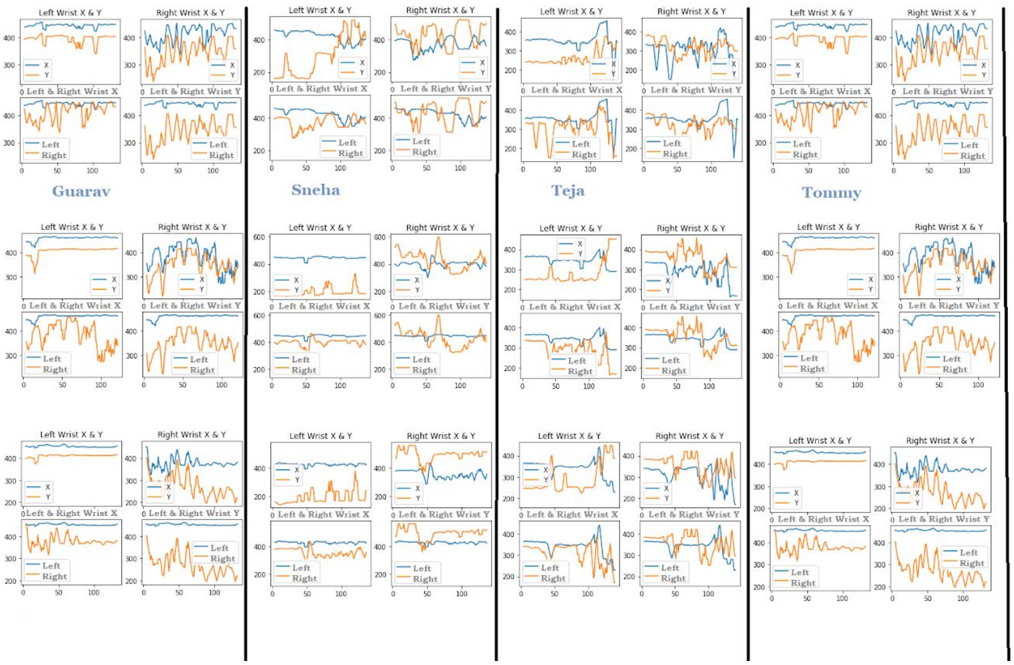
Good sep: Fun vs. Really, Mother, Hope

Right Hand Y

Good sep: Hope vs. Buy, Communicate ?

This plot shows the average position of each hand during a gesture. Each colored line represents a different Gesture. The Y position of the left hand shows a particularly good divide between several gestures, indicating a possible decision boundary.

The next plot shows the left and right wrist movements for each individual person across each video of the Buy gesture. Each Column shows the movement for a person across three different videos. Clockwise from top left to bottom the plots are as follows: Left wrist X & Y movement, Right Wrist X & Y movement, Left and Right wrist X movement, Left and Right wrist Y movement.



Plots of Wrist movements: All four members.

Buy Gesture.

Each line is a video clip from that person.

Each Row has four plots from Top Left to Bottom Right:
1: Left Hand X and Y movement
2: Right Hand X and Y movement
3: Both Hands, only X movement
4:Both Hands, only Y movement