Isolated Sign Language Recognition (ISLR)

Mothi Gowtham Ashok Kumar
Spring 2023 – INFO-H518

Faculty – Sunandan Chakraborty, *Ph.D.*
Luddy School of Informatics, Computing, and Engineering, Indianapolis

Introduction:

Each day, 33 infants are born in the U.S. with permanent hearing impairment, and most of these infants (about 90%) are born to parents who can hear and may not have knowledge of American Sign Language (ASL). As a result, deaf infants may suffer from Language Deprivation Syndrome, a condition characterized by the inability to acquire language naturally during their critical learning years. This syndrome can have severe consequences for their relationships, education, and employment prospects.

Acquiring sign language can be challenging, with American Sign Language (ASL) being as difficult for English speakers as learning Japanese. Learning sign language requires both time and resources, which many parents may not have. Although parents may wish to learn sign language, their long work hours to make ends meet can make it difficult to find the necessary time and money for classes. Additionally, classes may be located far away from their homes. [2]

Problem Description:

The steep learning curve to learn ASL poses a threat towards differently abled individuals as well as able bodied enthusiasts. We would like to formulate a deep learning application that would enable easy learning and provide feedback to individuals on their signs and poses. Our project uses MediaPipe, a product by Google to take snapshots of recorded videos of 21 participants performing ASL signs for 250 unique words and store them as a parquet file, providing spatial information on the face, hands, and body of the participants. We will use this data to train our deep learning models on recognizing these isolated signs which will form the basis for our application.

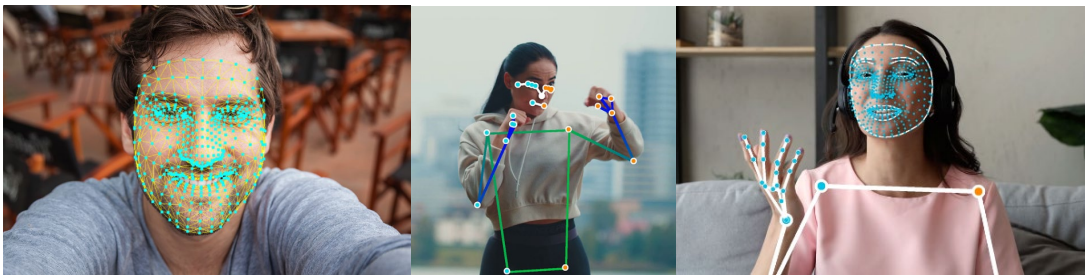

Fig 1. Landmark indexes of MediaPipe [1]

Description of Data:

The files are in the "**train_landmark_files**" directory and are named after the participant and sequence IDs, with a ". parquet" extension. These files contain data about landmarks, which were extracted from raw videos using the MediaPipe holistic model. Not all frames have detectable hands, however. It is important to note that landmark data will not be used to identify individuals or store unique biometric identification. The data is not intended to facilitate any form of identity recognition.

The data in the files includes several columns:

- "frame" refers to the frame number in the raw video,
- "row_id" is a unique identifier for each row,
- "type" specifies the type of landmark (e.g., face, left hand, right hand, pose), and
- "landmark_index" is the number associated with each landmark location.
- [x/y/z] columns contain the normalized spatial coordinates of each landmark, with only the [x/y] values provided for model inference. The MediaPipe model is not fully trained to predict depth, so the [z] values may be ignored.

The "**train.csv**" file contains information about

- the path to each landmark file ("path"),
- unique identifiers for the data contributor ("participant_id"),
- landmark sequence ("sequence_id"), and
- the label for the sequence ("sign").

https://www.kaggle.com/competitions/asl-signs/data

Data Transformations:

We perform multiple transformations of the data based on the model implemented. For the Fully connected neural network model we create a feature generator class which will pull information of landmark coordinates of the face, left hand, right hand and body pose from the parquet files (all frames/rows, each file represents a sign) as separate arrays. We then extract the mean and standard deviation of each array then concatenate the arrays.

The data is stored as NumPy arrays and stored as a NumPy file. In this method we reduce the data from ~50GB to ~3GB. Of course, we would lose information of each frame's landmark coordinate. Here we have stored only the mean and standard deviation of landmark index point that exists in a file. We do this to test how the FCNN model works on just the mean and standard deviation of the landmark index point calculated over each parquet file.

For the sequence-to-sequence model we reshape the data in window sizes of 32. Here we extract landmark points (x, y, z coordinates) related to lips, left hand and right hand which we find is able to get us good accuracy after multiple trial and errors. Landmarks of pose and other facial features are not very useful in predicting a sign.

For both types of transformations, the null values are replaced with zeroes.

Modeling:

Model 1 – Fully Connected Neural Network: To start with we used a Fully Connected Neural Network on the dataset created by calculating the mean and standard deviation of the landmark indices. We use this model to achieve baseline accuracy.

Activation = “Gelu”, “dropout”, “Batch Normalization”, epoch: 100 with early stopping, batch size = 64
Optimizer: adam, Loss: categorical_crossentropy

Accuracy: 55.17%
Macro-averaged Precision: 57.16%
Macro-averaged Recall: 55.14%
Macro-averaged F1-score: 53.84%

Model 2 – RNN: We build a simple RNN model on the landmark indices of lips, right hand and left-hand coordinates.

Activation = “ReLU”, “dropout”, “Layer Normalization”, epoch: 500 with early stopping, batch size = 256
Optimizer: adam, Loss: categorical_crossentropy

Accuracy: 19.14%
Macro-averaged Precision: 18.98%
Macro-averaged Recall: 19.35%
Macro-averaged F1-score: 17.02%

Model 3 – LSTM: We build on the last model and employ an LSTM model on the same indices as used for the RNN.

Activation = “ReLU”, “dropout”, “Layer Normalization”, epoch: 500 with early stopping, batch size = 256
Optimizer: adam, Loss: categorical_crossentropy

Accuracy: 62.34%
Macro-averaged Precision: 62.59%
Macro-averaged Recall: 62.72%
Macro-averaged F1-score: 61.82%

Model 4 – GRU: To check for further improvement we use a GRU model.

Activation = “ReLU”, “dropout”, “Layer Normalization”, epoch: 500 with early stopping, batch size = 256
Optimizer: adam, Loss: categorical_crossentropy

Accuracy: 66.11%
Macro-averaged Precision: 66.17%
Macro-averaged Recall: 66.53%
Macro-averaged F1-score: 65.61%

The Sequence-to-Sequence models follow the below architecture in Fig 2., The RNN layer is replaceable with LSTM, GRU.

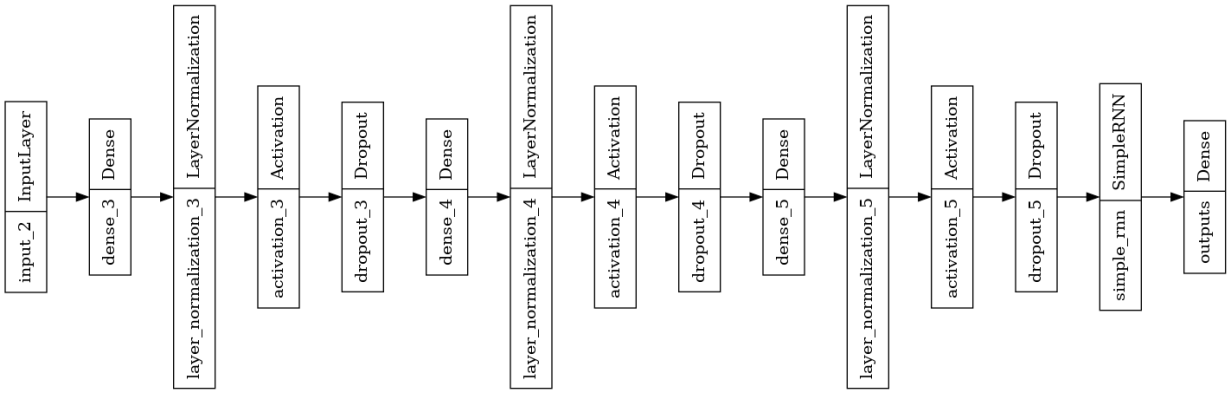

Fig 2. Skeletal Model Architecture [3]

Methodology:

Since the dataset is large and we wanted to experiment on simpler models like fully connected networks the mean and standard deviation of the landmark indices were calculated and used as the input to the network. This was able to provide almost a 55% accuracy. Additionally, the GELU activation function was able to provide better results compared to generally used activation functions like the RELU.

For the sequence-to-sequence model instead of using all landmark indices through trial and error we came up with using only movement of the lips, and hands as input data. This makes sense because for a particular sign to be expressed the hands and lip movements are more meaningful than pose and other facial features.

Also employed are Batch Normalization for the Fully connected network and Layer Normalization in the Sequence-to-Sequence Models. We see our loss decrease further while experimenting with Piece Wise Constant Decay function as a learning rate input to our optimizer. This function returns a callable object that takes the current step and returns the learning rate for that step based on the defined schedule. This function returns a callable object that takes the current step and returns the learning rate for that step based on the defined schedule.

We use multiple methodologies to work out a good accuracy of around 66%.

Result:

	FCNN	RNN	LSTM	GRU
Accuracy	55%	19%	62%	66%
Precision	57%	19%	63%	66%
Recall	55%	19%	63%	67%
F1 Store	54%	17%	62%	66%

Table 1. Model performance comparison

From our experiments we see GRU to perform better compared to other models. One interesting insight to see here is that the FCNN has better performance compared to the RNN model. With feature engineering on the dataset specifically for FCNN the model can perform better than RNN.

We also tested Transformer Model in the same format as the other seq-to-seq models, but the loss did not decrease after the 2nd epoch. Further analysis would be required to identify the causes.

Future Work:

The future scope of the project would include designing a better deep learning model that would provide better performance metrics compared to the ones we had already explored. Implementing a real-time sign language recognition effort will also be required to proceed with the project for our application to be made a reality.

Conclusion:

In conclusion, the Isolated Sign Language Recognition (ISLR) project aimed to develop a deep learning application to recognize American Sign Language (ASL) signs and provide feedback to individuals on their signs and poses. With 33 infants born each day in the U.S. with permanent hearing impairment, the project aimed to help individuals with hearing impairment, as well as able-bodied enthusiasts, to learn sign language more easily. The team used MediaPipe to take snapshots of recorded videos of 21 participants performing ASL signs for 250 unique words and store them as a parquet file, providing spatial information on the face, hands, and body of the participants.

The data was transformed and fed into four models: Fully Connected Neural Network, RNN, LSTM and GRU. The models were evaluated based on accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score. GRU showed the best performance out of the experimental models. Although the models did not achieve high accuracy, they provided a starting point for future research on ISLR. The project highlights the importance of using technology to help individuals with hearing impairment overcome challenges and barriers in their lives.

References:

[1] MediaPipe

[2] Kaggle Competition by Google

[3] Project Source Code

Appendix 1: Contributions from each member

Tasks	Team Members
Dataset selection	Alan Varkey, Mothi Gowtham Ashok Kumar
EDA	Mothi Gowtham Ashok Kumar, Alan Varkey
Testing different model architecture.	Rohan Satya Isaac, Alan Varkey, Mothi Gowtham Ashok Kumar