Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 235 (2024) 2684–2692

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering (ICMLDE 2023)

Gesture-to-Text: A Real-Time Indian Sign Language Translator with Pose Estimation and LSTMs

Shubham Shetty[a]* , Ebrahim Hirani[a], Abhir Singh[a], Reeta Koshy[a]

[a]*Sardar Patel Institute of Technology, Munshi Nagar, Mumbai, 400058, Maharashtra, India*

Abstract

In recent years, there have been notable advancements in technology, deep learning, and pose detection. One significant application of these advancements pertains to the real-time detection of sign language from video sources. The motivation behind this research stems from the pressing societal need to enhance the quality of life for individuals with speech impairments. Given the current prominence of online meetings, exacerbated by the COVID-19 pandemic, there is a growing need for systems that can provide individuals with speech impairments greater independence in communication, eliminating the requirement for a human translator. This research proposal advocates for a solution that leverages PoseNet algorithms for the extraction of key pose points, which are subsequently employed within LSTM models for the predictive modeling of sign language gestures. This research paper aims to make several notable contributions to the field of assistive technology and human-computer interaction. The achieved accuracy stands at an impressive 98%, underscoring the robustness and precision of our proposed system.

Keywords: Sign Language Detection ; Body Pose estimation ; PoseNet ; LSTM ; Deep Learning

1. Introduction

Sign language is considered to be a form of communication used by people with impaired hearing and speech. They use sign language gestures as means of non-verbal communication to express their emotions and thoughts in a better way. But on the other hand, for people who do not understand sign language, they find it extremely difficult to converse. Therefore interpreters who are highly trained in sign language are needed during important appointments be it medical or legal and also during educational training sessions. Over the past years, there has been an increasing

* Shubham Shetty.
E-mail address: shubham.shetty@spit.ac.in

10.1016/j.procs.2024.04.253

demand for interpreters. In recent years, various methods using deep convolutional neural networks (CNNs) have achieved breakthroughs in gesture recognition and recurrent neural networks (RNNs) have shown significant results. Our mission is to create a system that helps people who converse in sign language express their thoughts in a more powerful way using cutting edge technology.
In this time of covid, getting an instructor for a differently abled person for an online conference is difficult. These issues cause impediments for people who converse only through sign language. Thus getting the text for the sign language in a video conference application would be quite helpful. The aim of the paper is to create an API endpoint which can be then used by a video conferencing application that facilitates easier communication with people who do not understand sign language.

We have created a sign language to text converting model that uses pose estimation as temporal encodings for time variant models to output text. We have created a model that is light enough to process information in real-time. The dataset that we have used for the implementation is of Indian Sign Language with around 300 words and 4000 videos. In this study, we use state of the art models like LSTM's to solve the problem of sign language detection for specially abled people.

Rest of the paper is structured as follows: In Section II, we briefly discuss related works. Section III describes the proposed methodology and the various experimentations carried out. Section IV, Results and Discussion. Section V, Conclusion.

2. Related Works

Ying Xie et.al. take video sequences and extract temporal and spatial features from them. They then use Inception, a CNN (Convolutional Neural Network) for recognizing spatial features and then use a RNN (Recurrent Neural Network) to train on temporal features. The accuracy of the proposed model drops with change in facial features and skin tones and more importantly the model does not extract correct features when faces are included in the video. The model used by the authors gave an unsatisfactory performance when clothes of the signers were varied.[1]

Amit Moryossef et.al. do the extraction of optical flow features which is based on human pose estimation and they then use a linear classifier. The authors then use a recurrent model directly on the input, which caused the accuracy to improve up to 91\% accuracy. They create a demo application to demonstrate use in real time.They bank on the pose estimation system to run in real-time on any user's device. This seems to be a difficult task, as even when we perform state-of-the-art pose estimation using OpenPose on a single frame using a GPU, it takes a lot of time. Running this on devices without hardware acceleration like a GPU may be too slow.[2]

Dongxu Li et.al. create a new dataset called WLASL with more than 2000 words. On this new dataset, Dongxu Li et.al. run two types of models: holistic visual appearance based approach, and the 2D human pose based approach. They end up getting an accuracy of 62.63\% at top-10 accuracy on 2,000 words/glosses. They do not take into consideration the spatial-temporal information in a video. The authors of this paper have not utilized word-level annotations.[3]

In this study, Gaolin Fang et.al. proposed a temporal movement model that is able to recognize adjacent signs and handle the transition between those transitions. They use devices to track the movements involved in making signs, that include cybergloves as well as trackers placed on the wrists as well as the back of the user(this one is used as a reference for the other two). A temporal clustering algorithm is used that is able to identify pertinent points in order to identify sign language. These clusters are created by training TMMs through HMMs for identifying transitions. Requires equipment and trackers which is not practical in our case. Requires clustering, hence, needs the whole set of frames for every single prediction. This is not practical and an approach that summarizes previous frames in a single state object is more practical for real time operations.[4]

Akira Fukui et.al. in the method proposed use a Multimodal Compact Bilinear Pooling technique, where the image and language representations are allowed to interact with each other and soft attention is used later to make an attended visual representation. This model is used to predict answers to questions. The process of generating attention maps for real-time processes like sign language translation is very hardware intensive. The attention to relevant areas in an image is very successful. It fits perfectly to the question answering system proposed but does not translate well to real time tasks like Sign Language Detection. [5]

In this study , W. Du et.al. have proposed a recurrent pose-attention network (RPAN) , with a innovative pose-attention mechanism to learn pose-related features adaptively at every time-step action prediction of RNNs. These features are then fed into the human-part pooling layer to construct a highly-discriminative pose-related representation for temporal action modeling. This study proposed by W. Du et al. did not indicate real time inference times of the model, which is important for real time applications.[6]

In this study, B Gebre et.al. have proposed a Random-Forest based sign language identification system. The system proposed by the authors uses low-level visual features and is based on the hypothesis that sign languages have varying distributions of phonemes (hand-shapes, locations and movements). They have evaluated the system on two sign languages – British SL and Greek SL and achieved average F1 scores of about 95\% – indicating that sign languages can be identified with high accuracy using only low-level visual features. The model gives a low test set accuracy, indicating some level of overfitting.[7]

Srujana Gattupalli et.al. propose multiple pose estimation models which are trained and evaluated over the ASLID dataset. The study proposed uses 5 methods to predict different points on the body. A 3D deformable model tracking system is proposed and applied to American Sign Language (ASL). The model is quite resource extensive and a good degree of Facial features for ASL can be recognised by the head orientation alone instead of expressions most of the times.[8]

In this study , Sang-Ki Ko et.al. have introduced the KETI (Korea Electronics Technology Institute) dataset which consists of 14,672 videos using which they developed a neural network model for translating sign videos into natural language sentences by utilizing the human keypoints extracted from the body, hands and face. Normalization is applied on the obtained human keypoint vector using the mean and standard deviation of the keypoints which is then used as input to the translation model based on the sequence-to-sequence architecture. All experiments were conducted with three different sets of feature vectors containing: a) only manual features, b) only facial features, or finally c) a combination of both manual and facial features. Sang-Ki Ko et.al. conclude saying that the amount of increase in accuracy is not extensive when facial expressions are considered. Their results show that head pose was more important than facial expressions and lip patterns.[9]

Eeva Jacobs et.al focus on the relation of facial expression with Sign languages. Although necessary to convey one's intent, the use of facial expressions is similar to that used in verbal communication. Since the hand-signer's face will also be visible to the video conference participants in our case, we believe that taking facial expressions into consideration for the model will not be necessary.[19]

D. Kothadiya et.al introduce an exciting approach combining GRU and LSTM units to effectively recognize and translate Indian Sign Language (ISL) gestures into English words. The model demonstrates high accuracy and real-time performance on a custom ISL dataset, leveraging InceptionResNetV2 for feature extraction. However, limitations include untested generalization to other sign languages and diverse real-world conditions, potential delays introduced by the paraphrasing tool affecting real-time usability, a relatively small dataset, and the need for further evaluation metrics and scalability considerations. [23]

3. Methodology

3.1. Dataset Description

The INCLUDE dataset [22] that we are using has 4292 videos (the paper mentions 4287 videos but 5 videos were added later). There are 3475 videos in the train split of the dataset, while the testing split of the dataset has around 817 videos.Each video in the dataset is a recording of 1 ISL sign, signed by deaf students from St. Louis School for the Deaf, Adyar, Chennai.

The authors have also made a smaller version of the dataset available which is called as INCLUDE50 which has 766 train videos and 192 test videos. We use the INCLUDE Dataset in our study and not the INCLUDE50. INCLUDE has various category of words and we prioritized the words in the category Adjectives, Days and Time, Greetings and Places. We used the aforementioned categories from the entire INCLUDE dataset. [22]

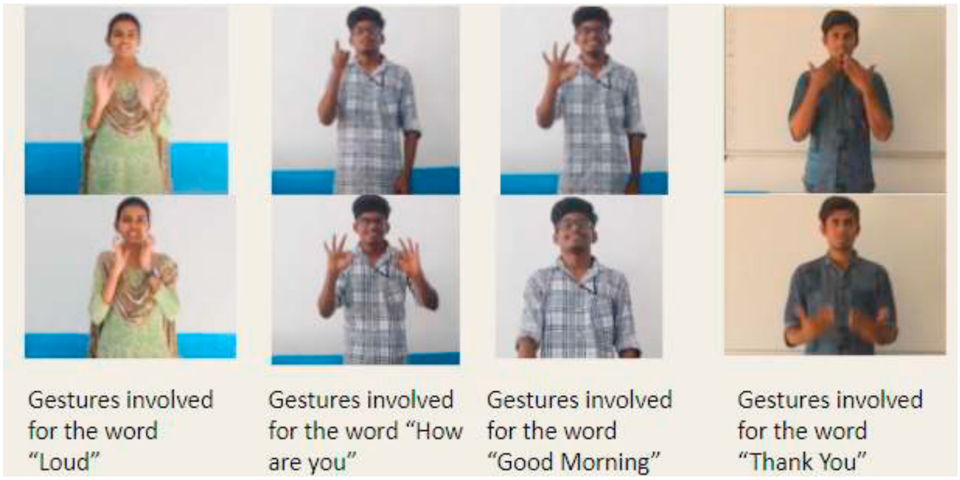

Gestures involved for the word "Loud"
Gestures involved for the word "How are you"
Gestures involved for the word "Good Morning"
Gestures involved for the word "Thank You"

Figure 1: Dataset Examples

3.2. Experimentation

Initially, we tried PoseNet model for pose estimation and then fed the normalized key points directly to an LSTM model in the form of timeframes. This methodology although the fastest at inference gave us an accuracy of 23\% on 20 classes. When we tried to improve it, the model tended to overfit and would only output the class with the highest representation.

The second implementation that we tried was using the HRNet model [20] for pose-estimation. We use 33 keypoints for the feature extraction phase in this implementation. After this, we pass these points through a CNN model and extract features. These features are then passed through the temporal attention based model for classification [21]. This model gave an accuracy of 92\% on 20 classes. A drawback that we discovered in this model was that the operations were not applied on single frames but on a set of frames. This would not only slow down the inference process significantly, but would also require all the frames at a time which is not a convenient approach for our pipeline as each new frame would require us to run operations on all the frames.

Later we tried the mediapipe hands model and used only the keypoints of a person's hands and we fed this to the Time Series model. In our first attempt, we only used the frames for which keypoints for both the hands were detected. This caused the model to lose a lot of information as many frames were skipped. This approach gave us an accuracy of 51\% and would again overfit when we would try to make major changes to improve its accuracy. Later we also used frames that also detected keypoints for a single hand and missed the detection for the other one. This improved the accuracy significantly as information from almost every frame was used in the model. This approach gave us an accuracy of 98\% on 43 classes. This approach is not only very accurate but also quick at inference time and is our current method of choice for the pipeline.

Another area that we experimented upon was tuning the hyper parameters of a particular model. This process was done for each and every model mentioned above. After trying out various values for the dropout parameter, dropout when set to 0.2 slowed the model down considerably when making inferences. On the other hand when we increased the dropout to a higher value, we experienced over-fitting of the model. After a lot of experimentation a dropout of 0.8 was finally used. All the models behaved in a similar manner when dropout was increased or decreased.

3.3. Final Approach

The dataset used in this study has limited number of videos per word which proved to be a limitation when we tried training our model. To overcome this problem, we augmented our dataset by using 20 versions of each video. Each video consisted of about 70\% of the total frames of that video as for each version, we dropped frames with a probability of 0.3.

Initially we apply pre-processing to the videos from the dataset before feeding it to the model for training. In the pre-processing phase, we use the Hands module from mediapipe to extract 42 keypoints (21 keypoints in each hand) from each frame. Now for each frame we store the X, Y and Z coordinates detected by the Hand Detector which then acts as the input for our model for the training phase. Some signs only use one hand of the signer and the other hand is not always visible, so to keep the data uniform for the model we append an empty array of 21 coordinates in the keypoints array as representation of the other missing hand. We save these keypoints as JSON files which have video names as the keys and the keypoints as values. These JSON files are feeded to the model as training data.

Since the positions of the keypoints not only depend on the word-sign but also on the distance of the camera from the person and the size/height of the person, which are factors that are variable, we try to eliminate these by normalizing all the points along every single axis, i.e., we make sure that along each axis (x, y, z) for a set of keypoints for a frame, the maximum coordinate value is 1 and the minimum coordinate value is 0. The normalization for 1 axis is independent of that of the other axes which allows us to map each keypoint in a scale of 0-1 along each axis.

For creating the deep learning model we use TFLearn. Since TFLearn can be defined as a modular and transparent deep learning aspect used in TensorFlow framework and the main motive of TFLearn is to provide a higher level API to TensorFlow for facilitating and speeding up new experiments. We start off by loading the training data on which we use the LabelEncoder from sklearn.

The first layer in the model is an Input Data layer which is basically a Core layer in TFLearn. We use this layer for feeding (aka inputting) data into the network. The next two layers in the model are Long Short Term Memory

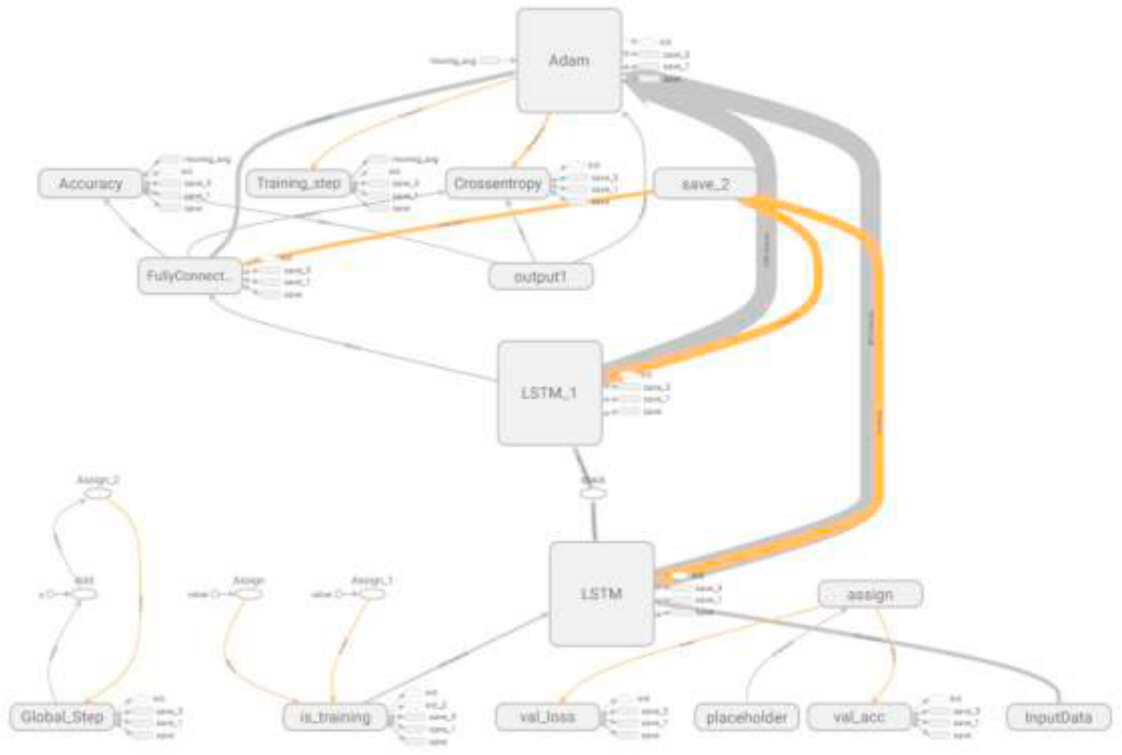


Figure 2: Model Architecture Graph

Recurrent layers which act as the memory of the model. Both the LSTM layers have 128 units. We add a dropout of 0.8 for the first LSTM layer as making it less would slow down the training and increasing it would cause the model to overfit. We also provide the return_seq Boolean as True since we want the entire sequence rather than the last sequence only. The next layer is the Fully Connected layer with the activation function as softmax. Following this layer we add a regression layer, which is used to apply a regression to the provided input. This layer requires us to specify a TensorFlow gradient descent optimizer that will minimize the provided loss function and we use the Adam optimizer for this task. Finally, we pass the network to a Deep Neural Network function in TFLearn and create the model.

All tables should be numbered with Arabic numerals. Every table should have a caption. Headings should be placed above tables, left justified. Only horizontal lines should be used within a table, to distinguish the column headings from the body of the table, and immediately above and below the table. Tables must be embedded into the text and not supplied separately. Below is an example which the authors may find useful.
For training, we split the data as 80\% for training and 20\% for testing. we train our model on the data for 40 epochs with a batch size of 32.

We validate the performance of our model by testing it on a couple video samples recorded by us which are not part of the dataset that the model is trained on.

After training the word level sign language model, we use it on a stream of videos for different words, to simulate how sign-language is used in real-life where words after words are to be detected to form a sentence. To achieve this, we create a pipeline that takes a sequence of frames one by one. Each frame is first passed through the pose-detection model to locate keypoints, and these keypoints are then fed to the LSTM model to predict the words. To avoid false predictions, we set a threshold of 80\% and a successful detection requires 5 consecutive outputs for the same word before confirmation. This setting produced the best results in our study. Furthermore, we wait for 20 frames to stack-up before we feed them to the LSTM model which also helps in false positives in our case.

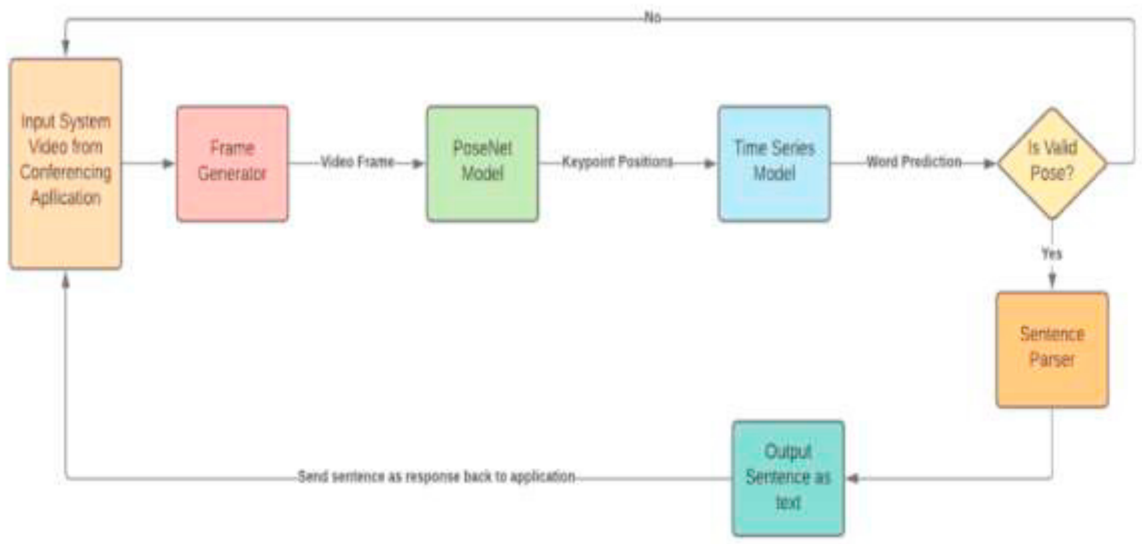
After a successful word prediction, the state of the model is reset, so that it is able to predict the next word without any influence from the model states for the previous word. This allows the model to output a sequence of words for the given sequence of frames.

Since sign language doesn't use preposition and verbs, to form a coherent sentence we need to paraphrase it by adding these words. For example, in sign-language, if a person wants to say "I am alright", they will only sign for the words "I" and "alright". To form a sentence, we need to add the missing verbs and prepositions. To do this, we use a paraphraser implemented on python called Parrot[22]. This allows our model to truly translate sign-language.

3.4. Application

We propose a system which deploys the model in an API endpoint using Flask. The model can be hosted on a cloud storage service from where we can access it easily. This will not only make the API endpoint very light but also make it faster for inferences. We plan to feed the frames one-by-one to the model in real-time using the API and the model will process these frames and will output the corresponding word. To avoid false detections, we plan to use a threshold value, which the model's confidence needs to cross in order to make an acceptable prediction.

Figure 3: Methodology Block Diagram



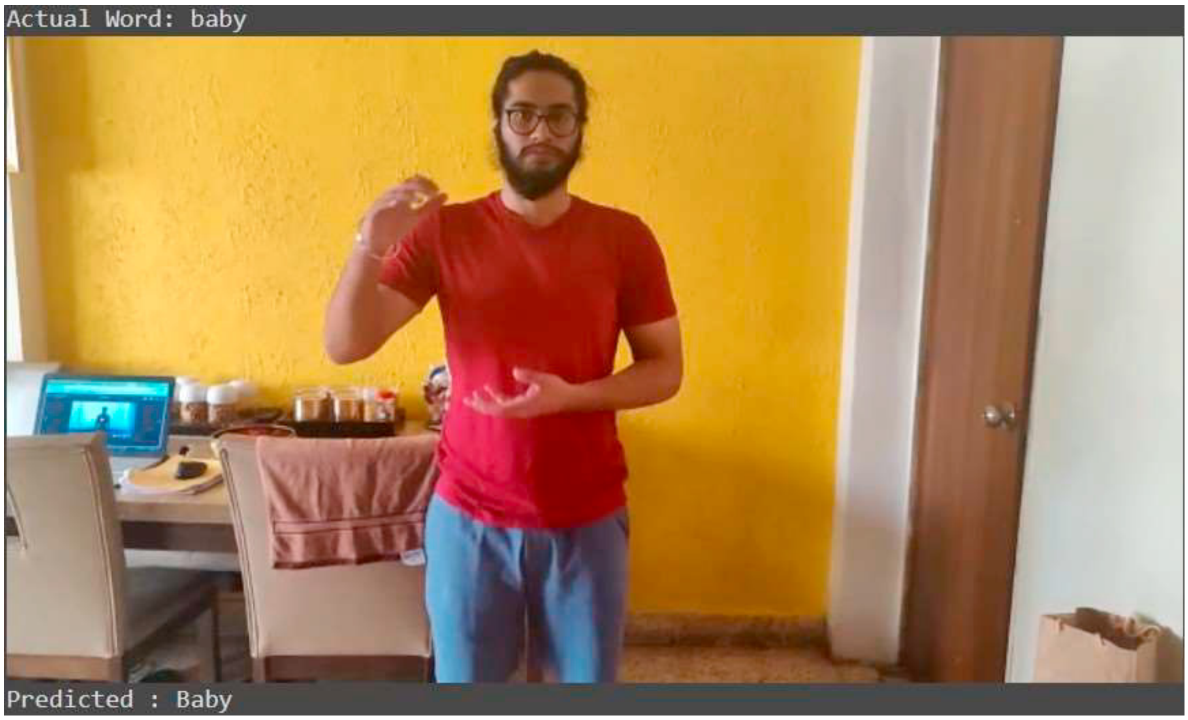
4. Results and Discussion

The model that we created ended up getting an accuracy of 98.44\% on the training data and 98.29\% on the validation data.

The model created by us works flawlessly even when there is a variation in clothes of the signers which was considered as a limitation in [1]. Also, our model has shown promising decrease in time to compute the sign from a video and does not need hardware acceleration like GPU's which was necessary in [2]. We have also used word level annotations which makes the predictions more accurate. This was considered as a gap in [3].

The inference time on CPU(2 cores @2.3GHz) for each video word pair was about half the actual length of the video, which encourages us to make use of it in real time applications.

All of our testings and experimentations have been summarized for better understanding in the table below. Please note that the mediapipe model has been run in 2 different configurations as mentioned in the section above. The single hands model doesn't discard a frame when just a single hand is detected by the MediaPipe model, while the both hands model actually discards a frame if it detects a single hand as it requires both hands. Our model is able to output the correct word for the videos that we recorded for testing which aren't a part of the dataset. This is demonstrated in the sample output shown in Fig. 4. This shows the model's ability to generalize over different situations.



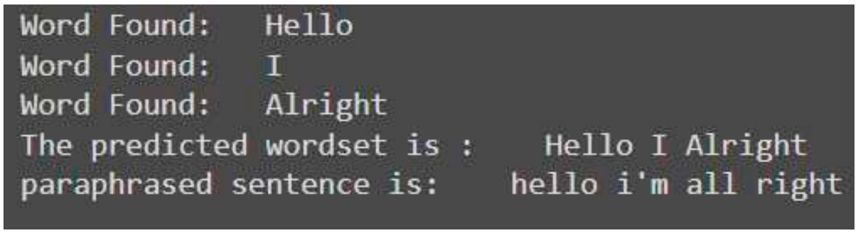
```
Actual Word: baby
Predicted : Baby
```

Figure 4: Sample Output

Furthermore, the paraphrasing library we use is able to stitch words to form sentences as shown in Fig. 5. The paraphrasing tool takes time to form sentences as it requires the whole context to form a sentence and this slows down the process. Inclusion of the paraphraser is not recommended for real-time use.

Table 1. Experimentation Table

Model	Accuracy
PoseNet + LSTM	23%
HRNet + CNN + LSTM	92%
MediaPipe(Both hands) + LSTM	51%
MediaPipe(Single hands also) + LSTM	98%



```
Word Found:   Hello
Word Found:   I
Word Found:   Alright
The predicted wordset is :    Hello I Alright
paraphrased sentence is:    hello i'm all right
```

Figure 5: Paraphrasing model working with sentence predictor

5. Conclusion

This study is an attempt at tackling the difficulties people with disabilities face in social situations. Notably, our model performs admirably even in scenarios involving variations in the clothing worn by signers, a challenge acknowledged in prior research [1] and [23]. Additionally, our system has demonstrated a substantial reduction in computation time for sign recognition from video inputs, eliminating the need for specialized hardware acceleration like GPUs, as previously required [2]. Through our application, we have developed a platform, although in its initial state, will help lessen this issue. We plan to perform more studies and integrate this application in as many systems as possible. This system can be used as an extension that displays texts in the signer's frame to other participants in video-conferencing applications. The system can achieve real-time performance if an efficient paraphrasing system is developed solely for this. The word prediction model does work in real-time and can be used without the paraphraser for real-time applications if required. With better hardware, processes like sentence paraphrasing can be made faster which can make the process truly real-time. Looking ahead, we plan to conduct additional studies to refine and expand our application's capabilities. This includes exploring the adaptation of our system to accommodate other sign languages and integrating it into a variety of communication platforms. By fine-tuning hardware resources and streamlining processes like sentence paraphrasing, we aim to fully unlock the potential of real-time performance, ultimately fostering more inclusive and accessible social interactions for individuals with disabilities.

References

[1] Bantupalli, K., Xie, Y. (2018). American Sign Language Recognition using Deep Learning and Computer Vision. 2018 IEEE International Conference on Big Data (Big Data). doi:10.1109/bigdata.2018.862214

[2] Amit Moryossef and Ioannis Tsochantaridis and Roee Aharoni and Sarah Ebling and Srini Narayanan, "Real-Time Sign Language Detection using Human Pose Estimation", 2008.04637, 2020.

[3] Word-level Deep Sign Language Recognition from Video: A New Large- scale Dataset and Methods Comparison.

[4] Gaolin Fang, Wen Gao, and Debin Zhao. 2007. Large-Vocabulary Con- tinuous Sign Language Recognition Based on Transition-Movement Models . IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 37, 1 (Jan. 2007), 1–9. https://doi.org/10.1109/TSMCA.2006.886347.

[5] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal Compact Bilin- ear Pooling for Visual Question Answering and Visual Grounding . arXiv:1606.01847 [cs] (Sept. 2016). http://arxiv.org/abs/1606.01847 arXiv: 1606.01847

[6] W. Du, Y. Wang, and Y. Qiao. Rpan: An end-to-end recurrent pose- attention network for action recognition in videos. In Proceedings of the IEEE International Conference on Computer Vision, pages 3725–3734, 2017.

[7] Gebre, B.G., Wittenburg, P., Heskes, T.: Automatic sign language identification. In: 2013 IEEE International Conference on Image Processing. pp. 2626–2630. IEEE (2013)

[8] Athitsos. 2016. Evaluation of Deep Learning based Pose Estima- tion for Sign Language Recognition. In Proceedings of the 9th ACM International Conference on Pervasive Technologies- 2016 https://doi.org/10.1145/2910674.2910716 answering system. Appl Intell (2021).

[9] Sang-Ki Ko, Chang Jo Kim, Hyedong Jung, and Choongsang Cho. 2019. Neural Sign Language Translation based on Hu- man Keypoint Estimation . arXiv:1811.11436 [cs] (June 2019). http://arxiv.org/abs/1811.11436 arXiv: 1811.11436.

[10] James Charles, Tomas Pfister, Mark Everingham, and Andrew Zisser- man. 2014. Automatic and Efficient Human Pose Estimation for Sign Language Videos. International Journal of Computer Vision, Vol. 110, 1 (Oct. 2014), 70–90. https://doi.org/10.1007/s11263-013-0672-6

[12] Recognition of Isolated Indian Sign Language Gesture in Real Time, Anup Nandy, Jay Shankar Prasad, Soumik Mondal, Pavan Chakraborty, G. C. Nandi, Communications in Computer and Information Science book series (CCIS, volume 70)

[13] Continuous dynamic Indian Sign Language gesture recognition with in- variant backgrounds by Kumud Tripathi, Neha Baranwal, G. C. Nandi at 2015 Conference on Advances in Computing, Communications and Informatics (ICACCI)

[14] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In Advances in neural information processing systems, pages 1799–1807, 2014.

[15] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In CVPR. 2014.

[16] Y. Yang and D. Ramanan. Articulated pose estimation with flexi- ble mixtures-of-parts. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1385–1392. IEEE, 2011.

[18] Elliott, Eeva & Jacobs, Arthur. (2013). Facial Expressions, Emo- tions, and Sign Languages. Frontiers in psychology. 4. 115. 10.3389/fpsyg.2013.00115.

[19] K. Sun, B. Xiao, D. Liu and J. Wang, ”Deep High-Resolution Represen- tation Learning for Human Pose Estimation,” 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5686-5696, doi: 10.1109/CVPR.2019.00584.

[20] S. Jiang, B. Sun, L. Wang, Y. Bai, K. Li and Y. Fu, ”Skeleton Aware Multi-modal Sign Language Recognition,” 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2021, pp. 3408-3418, doi: 10.1109/CVPRW53098.2021.00380.

[21] Damodaran, P. (2021). Parrot: Paraphrase generation for NLU. (v1.0) [Computer software]

[22] Advaith Sridhar, Rohith Gandhi Ganesan, Pratyush Kumar, and Mitesh Khapra. 2020. INCLUDE: A Large Scale Dataset for Indian Sign Language Recognition. In Proceedings of the 28th ACM International Conference on Multimedia (MM '20). Association for Computing Machinery, New York, NY, USA, 1366–1375. https://doi.org/10.1145/3394171.3413528

[23] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A.-B. Gil-González, J.M. Corchado, Deepsign: Sign Language Detection and Recognition Using Deep Learning, Electronics. 11 (2022) 1780. https://doi.org/10.3390/electronics11111780.