Isolated American Sign Language Recognition from Video Streams

Pranav Tandra (tandra.b@northeastern.edu), Tarun Thandu (thandu.t@northeastern.edu)

Abstract

This research paper addresses the communication gap between deaf children and their hearing parents by developing a machine learning model that could be integrated into a mobile application to teach parents sign language, specifically American Sign Language (ASL). The paper begins by highlighting the challenges faced by hearing parents in acquiring ASL proficiency and the negative consequences for deaf children, such as Language Deprivation Syndrome. To bridge this gap, we propose a smartphone application that combines ASL sign videos with written English words in an interactive game-like format. Additionally, they integrate a sign language recognizer into the app to allow users to practice signing themselves. The paper reviews previous works in sign language recognition, including the use of depth sensors, hidden Markov models (HMMs), and convolutional neural networks (CNNs), highlighting the advancements made in the field.

In terms of work, we preprocess a large dataset of ASL sign videos by normalizing the spatial coordinates of landmarks and filling missing values and perform exploratory data analysis and visualization to gain insights into the dataset's structure and characteristics. The paper then discusses the implementation of three models for sign language detection: Long Short-Term Memory (LSTM), Transformer, and Gated Recurrent Units (GRU). All the three models use custom embeddings, but the transformer also used positional and CLS encodings. The paper concludes by summarizing the contributions of their research and outlining potential future directions, such as dataset expansion, real-time applications, and multi-modal approaches, to further improve ASL sign recognition systems and support the language learning process of deaf children.

Introduction

Congenital hearing loss is a significant global issue that affects millions of individuals. In the United States alone, approximately 33 babies are born with permanent hearing loss daily, with around 90% of them being born to hearing parents who may not possess proficiency in ASL. The communication gap between deaf children and their hearing parents poses a significant challenge, as many parents lack knowledge of sign language. This puts deaf babies at risk of Language Deprivation Syndrome, which hampers their natural language acquisition during critical developmental years, leading to adverse effects on their relationships, education, and employment prospects. Learning American Sign Language (ASL) can significantly contribute to addressing this problem. However, learning sign language presents numerous challenges. Studies indicate that acquiring ASL is as difficult for English speakers as learning Japanese. The process demands substantial time and resources, which many parents may not have at their disposal, particularly if they are working long hours to make ends meet.

Recognizing the potential of games in facilitating learning, a smartphone application can be developed to make the learning process interactive and enjoyable. To facilitate the development of such applications, this paper aims to propose a Machine Learning model to recognize ASL signs. Integrating a sign language recognizer into such an application would empower players to practice signing themselves instead of solely watching videos, fostering meaningful connections between deaf children and their parents. The ultimate objective of our project is to enhance the learning process and boost the confidence of players aspiring to communicate through sign language with their loved ones.

Literature Review

Over the course of several years, there have been numerous attempts to address the challenge of recognizing sign language using various machine learning techniques. A few pivotal works are mentioned below.

The early work by Starner and Pentland [1] focused on real-time sign language recognition using a depth sensor. This study explored the potential of using depth information to recognize ASL signs, laying the foundation for subsequent research in the field.

Building on this, Vogler, Metaxas, and Soong [2] introduced the application of hidden Markov models (HMMs) to recognize ASL signs. Their work demonstrated the feasibility of using HMMs as a framework for recognizing sign language gestures and paved the way for further advancements.

Jemni and Bouzid [3] delved deeper into fingerspelling recognition in ASL using HMMs. By specifically targeting fingerspelling, which is an important aspect of ASL, their study contributed to improving accuracy in recognizing individual letters and words. Koller, Ney, and Bowden [4] explored the usage of time-of-flight cameras for ASL recognition, leveraging the depth information provided by these cameras to enhance sign detection and tracking.

The advent of deep learning brought significant advancements to the field. Pu, Zeng, and Zuo [5] proposed a hybrid CNN-HMM approach for continuous sign language recognition, combining the strengths of convolutional neural networks (CNNs) in visual feature extraction and HMMs for temporal modeling. Kamath, Bremond, and Thonnat [6] developed a real-time ASL fingerspelling recognition system using CNNs, achieving promising results in accurately recognizing individual letters.

Recent survey papers, such as those by Abd-Almageed et al. [7] and Atia and Jelinek [8], have provided comprehensive overviews of sign language recognition using deep learning models. These surveys highlight the progress made in the field, summarize the different techniques employed, and identify existing challenges.

These works have collectively contributed to the development of techniques for detecting English words from ASL signs using machine learning. From early explorations with depth sensors and HMMs to the utilization of CNNs and hybrid models, these studies have paved the way for improved accuracy and real-time recognition, providing a foundation for the development of assistive technologies for deaf children.

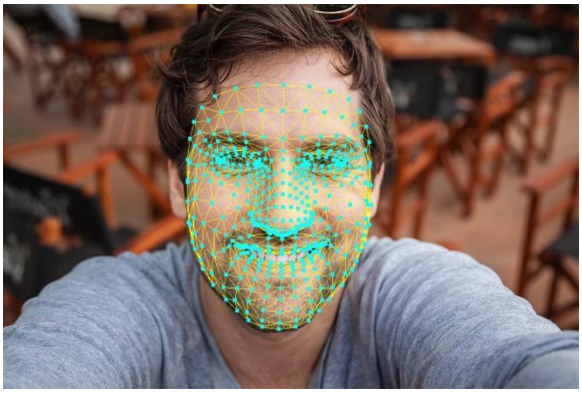
Dataset Analysis

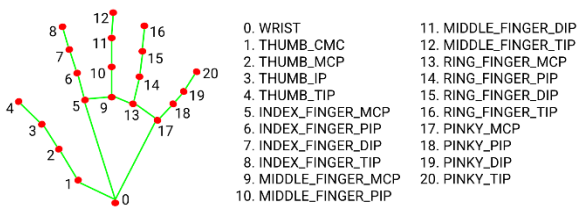
The Isolated Sign Language Recognition corpus (version 1.0) is a collection of hand and facial landmarks generated by Mediapipe version 0.9.0.1 on ~100k videos of isolated signs performed by 21 deaf signers from a 250-sign vocabulary based on the American Sign Language. The dataset is approximately 56 GB in size and is stored in Parquet file format. It contains 94477 parquet files each containing information about a single video sequence which represents an English word. Each row represents a frame in the video. There are 250 classes, where each class label is an English word that represents the ASL sign being performed in the video.

Each frame contains the normalized spatial coordinates of each of 543 landmark points. There are 4 types of landmarks - 'face', 'pose', 'left hand', 'right hand' where face contains 468 landmarks, each hand contains 21 landmarks and pose contains 33 landmarks. Each row in the landmark data file contains 7 columns – 'row id', 'frame number', 'landmark type', 'landmark index', 'x', 'y', and 'z' coordinates of the landmark point, and 543 rows make up a single frame.

The pose, face and hand features in this dataset have been extracted using MediaPipe, a module that utilizes a combination of computer vision and machine learning techniques to detect and track landmarks in real-time. The deep neural network behind this is designed to regress the coordinates of these landmarks, estimating their positions accurately.

The dataset contained video sequences which contained frames with missing values for landmark points. This is expected because not all landmark points would be visible in every frame.





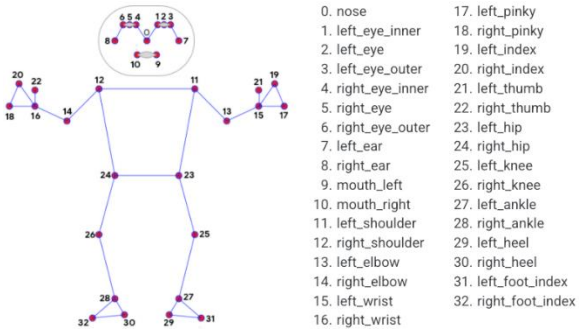


Fig 1. Face, hand and pose landmarks as detected by MediaPipe.

Data preprocessing

• Each parquet file represents a video, and they all have different numbers of frames. Each frame contains spatial coordinate information about 543 unique landmarks, including face, pose and hand landmarks. Hence, each frame is represented by 543 rows. We decided to remove most of the face landmarks since face plays an insignificant role in sign language, bringing the total landmarks from 543 to 88. This makes the data much easier to train given the limited amount of computational resources we possess as the amount of memory required to train a model would decrease drastically. We then pivoted the parquet files and the new parquet files represent each row as a frame and the features are x and y coordinates of the 88 landmark features of the corresponding frame. So, there are a total of 176 features (x0, x1….x87, y1, y2, … y87). We performed *MinMaxScaling* on the data to normalize it to lie in the range of [0,1], and then imputed the missing values with mean.

We then ran the *PaddingAndTruncating* script to make every video sequence have a uniform number of frames. Each video sequence now has 135 frames. We decided upon this number because 95 percent of the samples in the dataset have 135 or less frames. In videos containing more than 135 frames, the excess frames have been removed through uniform sampling as subsequent frames would be very similar and removing one of them would not remove a lot of information. In videos containing less than 135 frames, the extra frames added contain all negative ones to indicate that they add no valuable information.

Proposed Approach

As per the project guidelines to use at least one traditional machine learning model, we tried various models such as Logistic Regression, Support Vector Machines, Random Forest, Naive Bayes and simple Fully Connected Neural Networks to solve this problem. However, none of these models could produce more than one percent accuracy on the validation set. This is because this task is closely related to the problem of activity recognition in videos where the sequential ordering of data is crucial. Hence, we did our study to find models that would work well on sequential data [9-15], and narrowed upon 3 models – Gated Recurrent Units, Long Short-Term Memory Networks and Transformer Encoders.

Transformer

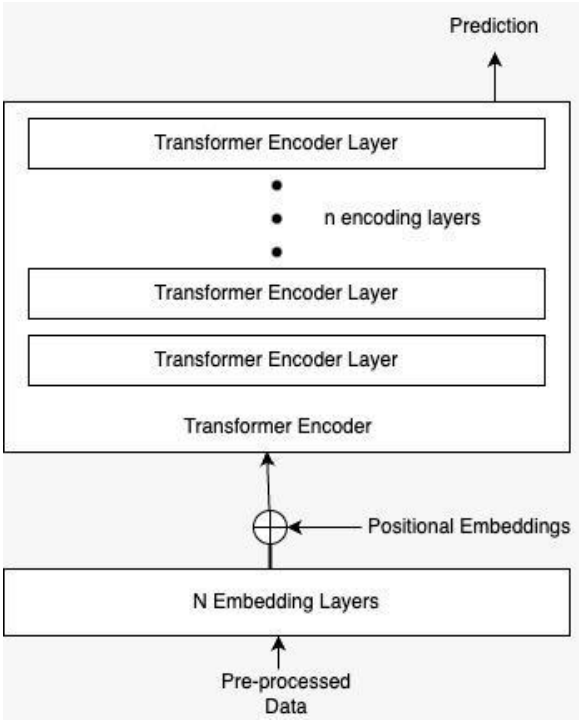


Fig 2. IASL model with Transformer Encoder

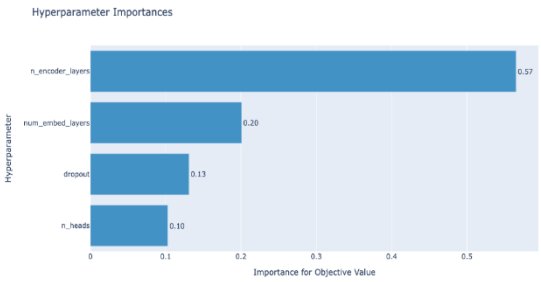


Fig 3. Hyperparameters importance comparison (Transformer)

• The data was processed in such a way that each input sample to the transformer has a fixed length, since one limitation of the transformer architecture *"Attention is all you need"* [13] is that it accepts only fixed length context. The preprocessed data was sent to a custom embeddings network to learn the embeddings vector from the 88 coordinate points. The embeddings network consists of 4 sequences of fully connected layers, ReLU activation and a normalization layer. Then, the learnt embeddings are concatenated with the positional embeddings from the positional encoder. We also added a CLS embedding. After that, the concatenated data was sent into a transformer encoder which consists of 2 Transformer Encoder layers. We used 4 heads for Multi-Headed Attention in the Transformer Encoder. The Transformer Encoder's output of the last frame is considered as the predicted class label. The complete architecture of our model can be seen in Fig 2. Our model for IASLR (Isolated American Sign Language Recognition) is inspired by the BERT model [14].

The number of heads in multi-headed Attention and the number of layers in the embeddings network and the Transformer Encoder were decided by performing hyperparameter tuning using Optuna [16]. We could only run 20 trials of 10 epochs each due to memory constraints arising from the dataset's huge size as we were often faced with the CUDA out of memory error. The hyperparameters that resulted in the least validation loss were selected to train the whole model. The number of transformer encoder layers has more effect than the rest of the hyperparameters as shown in Fig 3.

For training the model we used an *AdamW* optimizer with weight decay of 0.1 and an initial learning rate of 0.0005 with *cosineAnnealingWarmupRestarts* scheduler with cycle value equal to 8. The loss function we used was Cross Entropy Loss since it is a multi-class classification problem.

We obtained a validation accuracy of 64% and mean accuracy of 75% and 64% on the train and validation sets respectively. There's a big difference between Training loss and Validation loss indicated in overfitting as shown in figure 4. Since we had to remove most of the facial landmarks and compress the data from 56GB to 14GB, the accuracies and losses took a hit. Any more than 15GB would be untrainable as the maximum amount of GPU Memory we had was 15GB and would result in a CUDA out of memory error even with the smallest batch sizes.

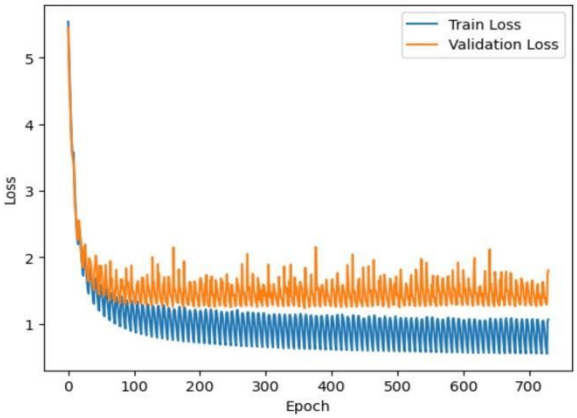


Fig 4. Epoch vs Loss (Transformer)

LSTM

We used the same preprocessed data for the LSTM model that we have used for the transformer model. We used a custom embeddings network in our LSTM model too but here the number of layers in the embedding network is fixed to 2 unlike in the transformer model where the number of layers is a hyperparameter. The learnt embeddings are then sent through a bi-directional LSTM with 5 layers and hidden size of 160.

Again, the hidden size and number of layers in LSTM was decided after performing hyperparameter using Optuna for 20 trials of 10 epochs each. The individual importance of these hyperparameters can be seen in Fig 4.

The optimizer, learning rate, scheduler and loss function are exactly the same as the ones in our transformer model. After training the LSTM model,

we obtained an accuracy of 60% on the validation set and mean accuracy of 91% and 60% on the train and validation sets respectively. We can observe from Fig 4 and 6 that the convergence is much slower in the LSTM model compared to the Transformer model. However, the issue of overfitting persists in this model too. This overfitting is most likely due to the fact we purged a lot of facial landmarks due to the lack of compute resources.

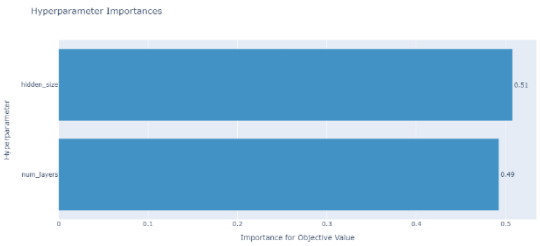


Fig 5. Hyperparameter Importance (LSTM)

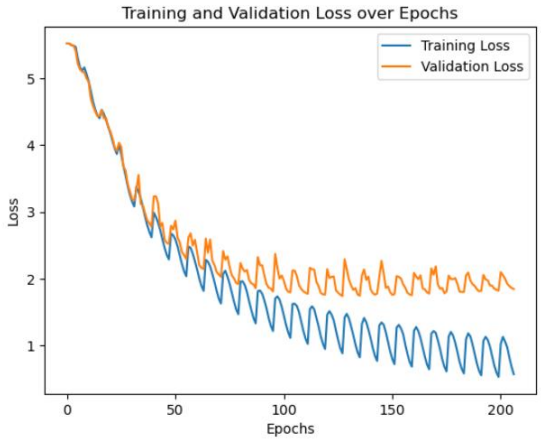


Fig 6. Epoch vs Loss (LSTM)

GRU

The same preprocessed data is used for GRU as used by Transformer and LSTM, whereas the same kind of custom embeddings network used in the LSTM model with 2 layers is used here too. The GRU we used in this model has 5 layers with a hidden size of 128.

The hidden size and number of layers are chosen using the same hyperparameter tuning process as in the LSTM model. The individual importance of these hyperparameters can be seen in Fig 7.

Again, the optimizer, loss function, learning rate and scheduler are the same as in both the Transformer model and the LSTM model. The validation accuracy is 58%, mean train accuracy is 76% and mean validation accuracy is 58%. The training curve of GRU can be seen in Fig 8.

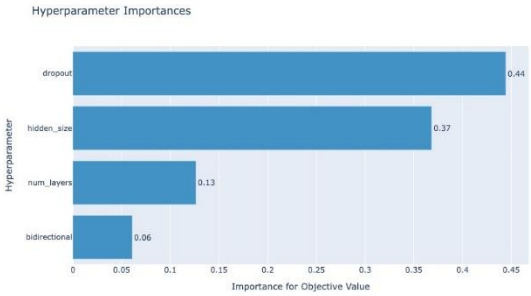


Fig 7. Hyperparameter Importance (GRU)

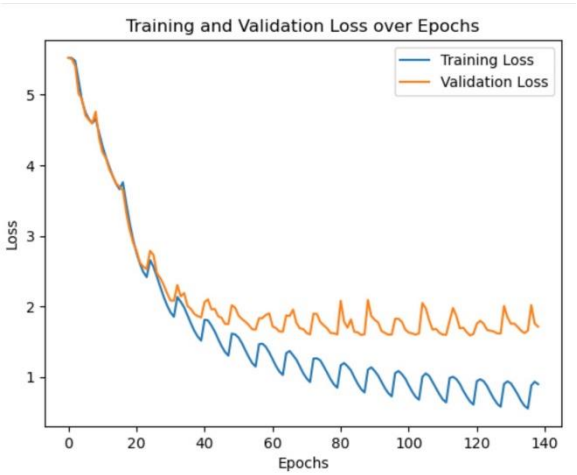


Fig 8. Epoch vs Loss (GRU)

Traditional Machine Learning Classifiers

We also trained our data on other Traditional Machine Learning Classification algorithms such as Support Vector Machines, Logistic Regression, Naive Bayes, Random Forests and K-Nearest Neighbors but they performed very poorly on our data.

For training on these models, we have not used the preprocessed data that we had used for Transformers, LSTMs and GRUs. Instead, we used the original unprocessed data, extracted the 176 features from that

data and padded and truncated with 0s instead of -1, performed Standard Scaling and then we made all the parquet files have a maximum length of 50 instead of 135. Having 50 still represents the data accurately since 80% of the data have length of 50 or less.

After this preprocessing step we performed hyperparameter tuning to choose the number of components on which to perform PCA (Principal Component Analysis). The number of principle components we have chosen is 13 as it represents all the data with a variance of 90% or more retained.

After PCA we flattened the data into a single row and applied the above-mentioned Vanilla ML classifiers. But the result was quite disappointing as all the models performed very poorly with validation accuracy of around 1%. This is because when we flattened the data all the temporal ordering between the data was lost which is quite important in these types of problems.

Analysis

The Transformer model is slightly superior to both the LSTM and GRU Models since it has around 4 percent higher accuracy on the validation set. Moreover, although overfitting occurs in all the models, it is not as high in the Transformer as it is in the rest of the models, since there's almost a 30 percent difference between train and validation set accuracies in the LSTM model, whereas in the Transformer model, it is only about 10 percent.

The mean per class accuracies of both the test set and the validation set are higher in the transformer model compared to LSTM and GRU.

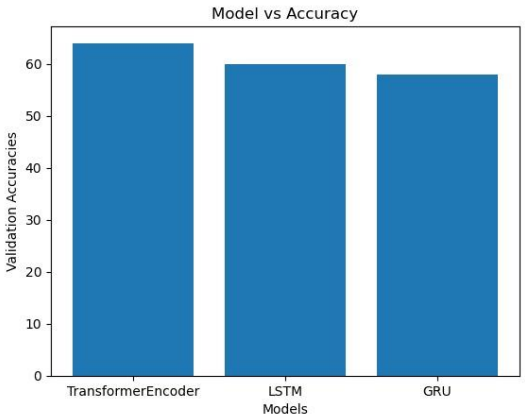


Fig 9. Models vs Accuracy

Finally, the running time of the Transformer is much lower than the running time of LSTM and GRU. Transformer took about 28 seconds per epoch and had its highest validation accuracy of 72% at epoch 224, whereas LSTM took about 95 seconds per epoch and had its highest validation accuracy of about 61% at epoch 192 and the GRU took about 75 seconds per epoch and had its highest validation accuracy of about 65% at epoch 128. The transformer also converged faster than the LSTM and the GRU.

Mean Accuracies	Transformer Encoder	LSTM	GRU
Training Set	75	91	76
Validation Set	64	60	58

Fig 10. Mean class accuracy

Conclusion

In this project, we explored the use of machine learning techniques for detecting English words from ASL signs in videos, aiming to provide a valuable tool to support deaf children in their language learning process. Through an extensive review of related works, we observed significant advancements in this field, with the development of various approaches including Transformers, LSTMs, GRUs, and hybrid models. These techniques have demonstrated promising results in accurately recognizing hand and facial landmarks associated with ASL signs, enabling real-time analysis and interpretation of sign language gestures.

In this paper, we tried to solve the problem of Isolated American Sign Language Recognition from landmark data of video files by using several models. The models that worked best for the problem are the ones that capture sequential dependency in the data such as Transformer Encoder, LSTM, and GRU. Other models like SVM, Logistic Regression, Naive Bayes, and Random Forest which do not have the ability to capture this dependency are unsuitable for solving this problem as they have very low prediction scores. The best model for this task is the Transformer Encoder model, which is similar to the BERT model but does not use any pre-training and masking. Transformer

Encoder has the best prediction score, the best loss, and the best training and convergence time.

These findings highlight the potential of machine learning approaches in enhancing the communication and educational experiences of deaf children, providing them with a means to bridge the gap between sign language and spoken/written language.

Future Work

Through this project, we were able to make substantial progress in the detection of English words from ASL signs. However, there are several avenues for future research and improvement. Some potential directions for future work include:

1. Dataset Expansion: Expanding the existing datasets with a wider range of ASL signs and linguistic variations can improve the robustness and generalization of the trained models.

2. Real-time Applications: Further optimization of the algorithms and models is needed to achieve real-time performance, enabling seamless integration into real-world applications such as interactive learning platforms or assistive devices.

3. Multi-modal Approaches: Investigating the integration of multi-modal data, such as combining video inputs with depth or thermal information, can enhance the accuracy and reliability of ASL sign recognition systems.

4. Gesture Variation and Adaptability: Addressing the challenges posed by inter-person and intra-person gesture variations, including different signers and variations within a single signer, is crucial for building more adaptive and personalized systems.

5. Long-term Gesture Understanding: Exploring the use of recurrent neural networks (RNNs), long short-term memory (LSTM) networks, or transformers for capturing temporal dependencies and enabling long-term gesture understanding can lead to more sophisticated ASL recognition systems.

6. User Interface and Feedback: Designing intuitive user interfaces and providing immediate feedback to users can enhance the learning experience and engagement of deaf children, promoting effective language acquisition.

References

1. Starner, T., & Pentland, A. (1995). Real-Time Sign Language Recognition Using a Depth Sensor. In Proceedings of the Fifth International Conference on Computer Vision (ICCV) (pp. 511-517).

2. Vogler, C., Metaxas, D., & Soong, F. (1998). Recognizing American Sign Language Using Hidden Markov Models. In Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG) (pp. 641-646).

3. Jemni, M., & Bouzid, M. (2001). Fingerspelling recognition in American Sign Language based on Hidden Markov Models. In Proceedings of the International Conference on Pattern Recognition (ICPR) (Vol. 1, pp. 954-957).

4. Koller, O., Ney, H., & Bowden, R. (2013). American Sign Language Recognition With a Time-of-Flight Camera. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP) (pp. 297-304).

5. Pu, J., Zeng, Y., & Zuo, W. (2016). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. In Proceedings of the International Conference on Image Processing (ICIP) (pp. 2657-2661).

6. Kamath, C., Bremond, F., & Thonnat, M. (2018). Real-Time American Sign Language Fingerspelling Recognition Using Convolutional Neural Networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 410-426).

7. Abd-Almageed, W., et al. (2020). Sign Language Recognition Using Deep Learning Models: A Survey. ACM Computing Surveys, 53(3), Article No. 64.

8. Atia, G. K., & Jelinek, H. F. (2021). Deep Learning-Based American Sign Language Recognition: A Survey. IEEE Access, 9, 25347-25361.

9. Pu, J., Zeng, Y., & Zuo, W. (2017). Fingerspelling Recognition in American Sign Language Using Recurrent Neural Networks. In Proceedings of the International Conference on Multimedia Modeling (MMM) (pp. 319-331).

10. Zhang, S., Yao, S., & Cao, L. (2017). Temporal Convolutional Neural Networks for Sign Language Recognition. In Proceedings of the International

Conference on Multimedia and Expo (ICME) (pp. 543-548).

11. Tripathi, N., & Sharma, G. (2018). American Sign Language Recognition using Long Short-Term Memory. In Proceedings of the International Conference on Recent Advances in Information Technology (RAIT) (pp. 1-6).

12. Li, C., et al. (2022). Transformer-Based Sign Language Recognition. arXiv preprint arXiv:2202.09390.

13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems (NeurIPS), 5998-6008.

14. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Association for Computational Linguistics (ACL), 4171-4186.

15. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 2978-2988).

16. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta,and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.

17. The dataset used for this project has been collected from Kaggle and is made available by Google. See https://www.kaggle.com/competitions/asl-signs/data