Real-Time American Sign Language Interpreter using Transformer and LLM

Aparna Bharati Suresh,
Applied Data Science Department,
San Jose State University
Email: aparnabharathi.suresh@sjsu.edu

Aryama Ray,
Applied Data Science Department,
San Jose State University
Email: aryama.ray@sjsu.edu

Shravani Dattaram Gawade,
Applied Data Science Department,
San Jose State University
Email: shravani.gawade@sjsu.edu

Sujata Deepraj Joshi
Applied Data Science Department,
San Jose State University
Email: sujatadeepraj.joshi@sjsu.edu

***Abstract*—Sign language is gaining widespread popularity, not only as a bridge between the deaf community and non-signers, but also as a critical tool for individuals with autism, speech impairments, and other disabilities who face challenges in vocal communication. According to the Modern Language Association, American Sign Language (ASL) was the third most studied language in U.S. colleges and universities in 2021, reflecting its growing importance in accessibility, education, and human-computer interaction. Previous work in continuous sign language interpretation has demonstrated that mid-level sign to gloss recognition improves translation performance. However, solely transformer-based architecture for sentence level-sign recognition and translation is computationally complex often requiring days of training. In this project we developed a novel approach of generating sentence level interpretation from Isolated Sign Videos. We used transformer Encoder architecture for Sign recognition and for text generation we leveraged GPT4 tokenizer, a well-known LLM for text generation. We trained the model using Google's Isolated American Sign Language (GISLR) Dataset [13] . This dataset is a collection of 94479 files MediaPipe landmark files. The landmarks were extracted from raw RGB video sequences with the MediaPipe holistic model [16] . Mediapipe holistic can extract hand, pose, face landmarks in structured format. The raw video data captured in this way makes it easy for model to learn gesture representation. The framework operates in real time on standard hardware and provides both gloss and contextual sentence outputs. We evaluated the sign-to-gloss model using Word Error Rate (WER) and the gloss-to-text stage using BLEU scores. The system achieved state-of-the-art performance, including a BLEU-1 score of 70.71% on sentence generation.**

***Index Terms*—sign language interpreter, MediaPipe, Transformer Encoder, GPT4, gloss prediction, BLEU score, WER, ASL translation, realtime inference**

I. Introduction

Sign language is a structured visual language that employs hand gestures, facial expressions, and body posture to convey meaning. For the Deaf and hard-of-hearing community, it is an essential mode of communication. In recent years, sign language has also gained significance in broader applications such as assistive technologies for individuals with speech impairments, autism, or temporary communication disabilities. However, the language barrier between signers and non-signers remains a challenge in achieving fully inclusive communication. With advances in deep learning, there is a growing push toward building real-time, automated sign language translation systems that can bridge this gap effectively and affordably.

Sign Language Recognition (SLR) systems can be broadly categorized based on the input type and level of linguistic granularity. At the alphabet level, static hand postures are used to represent letters, commonly seen in finger spelling. At the word level, isolated gestures represent individual vocabulary units. Sentence-level recognition typically involves continuous video sequences with complex temporal dependencies between gestures, body movement, and facial expressions. Depending on the modality, inputs can range from static images and short video clips (isolated sign recognition) to continuous video streams (sentence-level translation).

Early works in vision-based gesture recognition relied on handcrafted features and rule-based systems, but recent research has moved towards deep learning-based methods that provide better generalization and robustness across signers and environments. For instance, Sharma and Singh proposed a CNN-based approach for static sign classification with high accuracy on isolated gestures. In contrast, systems like DeepASL and Camgoz et al.'s Sign Language Transformer (SLT) target continuous recognition and translation using recurrent and attention-based neural architectures. While these systems achieve high performance, they are often computationally expensive and require large, annotated datasets.

To address these challenges, we propose a hybrid approach that begins with Isolated Sign Recognition and extends to sentence-level translation, leveraging the modularity and interpretability of a two-stage pipeline. Our system uses MediaPipe Holistic, a real-time perception framework by Google, to extract 3D landmark coordinates of hand, face, and body joints from input videos. These landmarks—capturing critical gesture and pose information—serve as compact, noise-resilient representations of sign language, reducing the reliance on raw pixel data.

Unlike traditional SLR pipelines that rely on heavy convolutional or recurrent architectures, we employ a Transformer Encoder model that learns temporal dependencies among the

pose sequences and predicts glosses—linguistic units that represent the meaning of a sign in written form. These gloss sequences are then translated into fluent, grammatically appropriate English sentences using a GPT-4-based language model, bypassing the need for training a separate decoder module. Our approach combines the lightweight computational benefits of isolated sign recognition with the contextual richness of sentence-level translation.

We evaluate our model on the GISLR dataset, which consists of isolated ASL gesture sequences captured from multiple participants. By using standardized metrics like Word Error Rate (WER) for gloss prediction and BLEU scores for sentence generation, we show that our system achieves high accuracy while remaining efficient enough for real-time inference on standard hardware. In doing so, we contribute a scalable, modular, and accessible framework for the next generation of sign language translation tools.

II. Related Work

A. Isolated word level Sign Language Interpretation

S. Sharma and S. Singh [15] proposed a vision-based hand gesture recognition (HGR) system utilizing a custom convolutional neural network (CNN) model, termed G-CNN. The model was trained on a self-constructed Indian Sign Language (ISL) dataset comprising 43 static gesture classes, with a total of 2150 RGB images collected from 50 different signers under diverse lighting and background conditions. In addition, the model's performance was evaluated on the publicly available Jochen-Triesch American Sign Language (ASL) dataset. The proposed G-CNN model, designed specifically for static gesture recognition, achieved classification accuracies of 94.83% for ISL alphabets and 99.96% for ISL static words.

B. Continuous Sign Language Recognition and Translation

Camgoz et al. (2020) [4] introduced a novel end-to-end Sign Language Transformer framework that jointly performs continuous sign language recognition (Sign2Gloss) and translation (Gloss2Text) using the PHOENIX14T dataset. The architecture includes two main components: a Sign Language Recognition Transformer (SLRT) trained with CTC loss for sequence-level gloss prediction, and a Sign Language Translation Transformer (SLTT) decoder trained with cross-entropy loss for spoken language sentence generation. Their Sign2Gloss+text model outperformed the baselines, achieving 21.32 BLEU-4 score. Also, their Sign2Gloss model achieved 24.59% WER for sign to word recognition.

Sincan et al.(2024) [14] combined gpt-3.5-turbo with their isolated sign spotter, which is based on the I3D model, to translate sign language video into spoken level sentences. Instead of gloss-to-text model training they used LLM to generate sentences. Their model achieved a better BLEU score than the traditional transformer model.

III. Dataset

The dataset used for our project is structured to facilitate pose-based gesture classification. It consists of three main components - a CSV file (train.csv) that lists the labels, a JSON file that maps sign names to numerical labels, and a set of parquet files that store the signs data for each sample of the participants. The overall dataset size is around 56 GB.

The train.csv Figure 1 is the primary index for the dataset. Each row in this file corresponds to a ASL gesture sample and includes identifiers that link to the corresponding parquet files along with the ground truth label of the sign performed, such as hello or thank you. This label is provided in text form and represents the gesture class that the sample belongs to.

	path	participant_id	sequence_id	sign
0	train_landmark_files/26734/1000035562.parquet	26734	1000035562	blow
1	train_landmark_files/28656/1000106739.parquet	28656	1000106739	wait
2	train_landmark_files/16069/100015657.parquet	16069	100015657	cloud
3	train_landmark_files/25571/1000210073.parquet	25571	1000210073	bird
4	train_landmark_files/62590/1000240708.parquet	62590	1000240708	owie

Fig. 1. GISLR index file - train.csv

To train a machine learning model, these textual labels were converted into numerical labels using the sign_to_prediction_index_map.json file. This JSON file maps each unique sign name to a numerical class index. For example, the label hello as 23, because the numerical encoding is necessary for supervised learning algorithms that expect numerical targets during training.

The core of the dataset is the set of parquet files, each containing a sequence of pose landmarks of face, hand and pose extracted from gesture videos. Instead of using raw video frames, these files Figure 2 store three-dimensional coordinates (x, y, z) of key points on the hands, faceFigure 3 , and body for each frame of the sign. These landmarks are extracted using a pose estimation model such as MediaPipe [16] , which detects and tracks key joints like fingertips, wrists, elbows, facial features and body postures.

	frame	row_id	type	landmark_index	x	y	z
0	103	103-face-0	face	0	0.437886	0.437599	-0.051134
1	103	103-face-1	face	1	0.443258	0.392901	-0.067054
2	103	103-face-2	face	2	0.443997	0.409998	-0.042990
3	103	103-face-3	face	3	0.435256	0.362771	-0.039492
4	103	103-face-4	face	4	0.443780	0.381762	-0.068013

Fig. 2. Parquet file data for Face landmarks

The dataset includes gesture data collected from 21 different participants, each identified by a unique participant ID. For instance, participant 16069 contributed a total of 4,848 sequences each represented as a separate parquet file. Each sequence corresponds to one instance of a sign being performed.

In total, the GISLR dataset has 250 distinct ASL words signed by 21 participantsTable I . The label distribution of our data is very balanced, with each class having a similar number of samples (around 350 - 420)Figure 4 . This even spread ensures the model will learn all signs fairly without

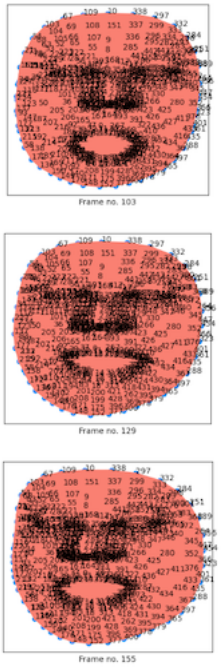

Fig. 3. Face landmarks from Mediapipe

TABLE I
GISLR DATASET SUMMARY

Total Participants	Total ASL Isolated Word Signs	Total Sample Size
21	250	94477

bias toward specific classes, leading to better overall accuracy & generalization.

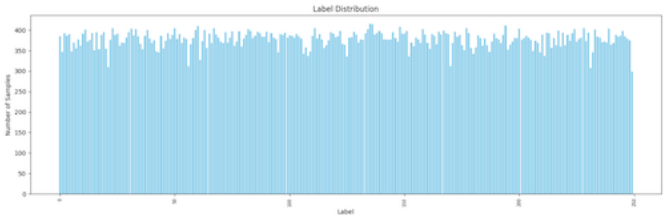

Fig. 4. Sign Label distribution from GISLR dataset

IV. Data Preprocessing

For our project, we performed the pre-processing in three steps. First, the dataset was organized using the three input files - a CSV file (train.csv) containing sign labels (example, hello), a JSON file mapping sign names to corresponding numeric class identifiers, and a set of .parquet files containing landmark data extracted from videos. These landmarks represent the position (x, y, z) of keypoints across the hand, face and body for each frame in a recording. The preprocessing pipeline began by loading these files using standard data-handling libraries such as Pandas and PyArrow. The CSV and JSON files were merged to replace sign names with numerical class labels, which are more suitable for training classification models. This cleaned dataset was then saved as final_train.csv.

After this we treated the .parquet files. To reduce noise and improve model performance, the z-coordinate was removed, as it typically represents relative depth rather than true 3D position and is known to vary significantly due to factors like camera angle and lighting. The pipeline also filters the data to retain only a subset of informative landmarks, specifically those likely associated with essential facial expressions and hand movements. These include landmarks such as the nose tip (index 0), areas around the eyes and cheeks (indices 9, 11, 13, 14, 17), and critical facial landmarks near the mouth and eyebrows (indices 117, 118, 119, 199, 346, 347, 348). Additionally, other columns such as row_id and type are removed. The remaining data was reshaped from a row-per-point format into a row-per-frame format, consolidating all x and y coordinates into a single feature vector per frame. The processed landmark data was stored in a new directory titled train_landmark_files_preprocessed.

In the second step, we normalized all the features onto a similar scale. This is achieved through a two-pass z-score normalization process. In the first pass, the pipeline calculates the global mean and standard deviation for each feature (i.e., each coordinate across all selected landmarks) across the entire training dataset. In the second pass, each individual file is normalized by subtracting the mean and dividing by the standard deviation, thus ensuring that the data conforms to a standard normal distribution.

The third step of the preprocessing addressed the issue of varying sequence lengths across video samples. Since deep learning models require uniform input dimensions, all sequences are standardized to a fixed length of 135 frames. Sequences longer than 135 frames are uniformly downsampled to retain representative frames, while shorter sequences are padded with a value of -1 to ensure the consistency across the dataset without losing temporal information.

In summary, the preprocessing pipeline transformed the diverse raw inputs into a uniform dataset. By converting labels, filtering and reshaping landmarks, normalizing features and standardizing sequence lengths, the pipeline ensured that the input data is both computationally efficient and semantically meaningful for subsequent training stages in sign language recognition.

V. End to End Method

This system aims to translate sign language video to gloss sequences and eventually to produce whole sentences through the application of deep learning modelsFigure 5 .

The pipeline consists of the following steps:

- Feature Extraction through Mediapipe

It starts off using video input that includes gestures in the form of sign language. Mediapipe's [16] real-time perception library is employed for the retrieval of the landmark key points for body, hand, and face. The key points indicate the spatial coordinates of the frame and are saved as efficient Parquet files.

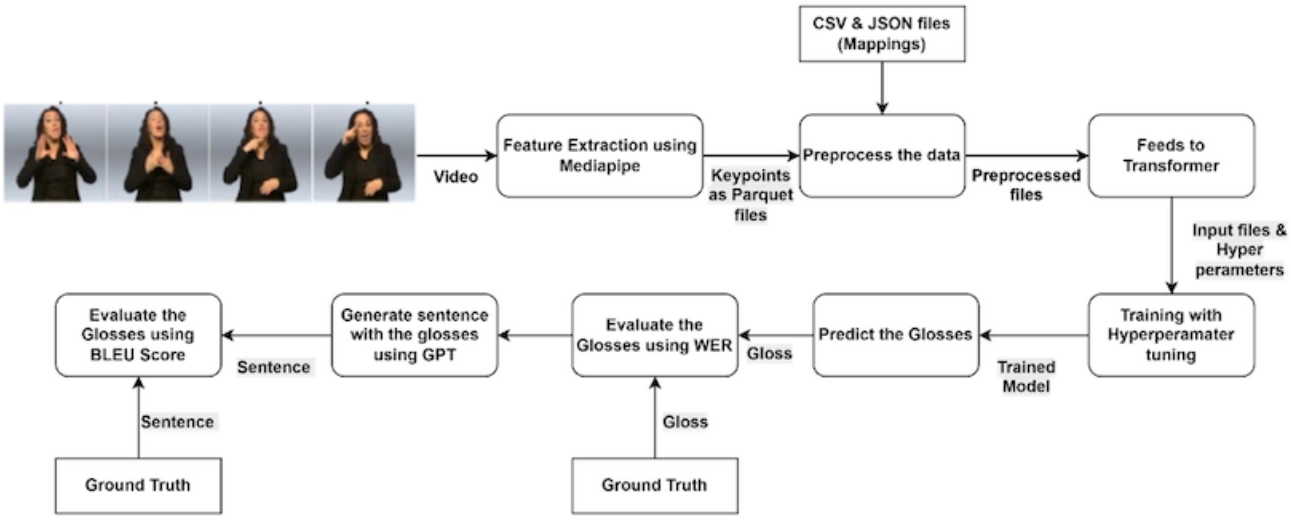

Fig. 5. Sign Language Gloss and Sentence Generation Pipeline

• Pre-processing the data

The key points that are extracted are enriched and aligned through metadata mappings that are supplied in CSV and JSON formats. This process ensures the standardization and labeling of keypoint data. It produces a series of preprocessed datafiles that comprise structured temporal pose information.

• Input to the Transformer Model

The preprocessed data and hyperparameters are fed into a Transformer model. The model learns to translate the temporal sequences of pose key points to corresponding gloss sequences (word-level signs).

• Hyperparameter Tuning for Training

The model is trained with techniques such as clipping of the gradients, scheduling of the learning rate (CosineAnnealingWarmRestarts), and cross-entropy loss with label smoothing and weighting of classes. It is optimized using the AdamW optimizer and improved in performance through hyperparameter tuning systematically.

• Gloss prediction

The model is employed after being trained to predict glosses for new input videos. These predicted glosses act as an in-between representation of visual gestures and spoken language.

• Evaluation using WER

The glosses predicted are also tested against the ground truth gloss annotations for their quality and accuracy using Word Error Rate (WER) [3] .

• Sentence Generation using GPT [17]

The predicted glosses are subsequently input into a language model that is based on GPT and produces fluent and semantically correct sentences in natural language.

• Final Evaluation by BLEU Score [12]

They are evaluated against the ground truth sentences through BLEU Score, which quantifies the quality of the machine-made sentences against human-written ones.

Overview This end-to-end process allows for the conversion of sign language clips into grammatically accurate sentences, going through the major stages of pose extraction, gloss prediction, and language generation. It incorporates the use of deep learning for bridging the gap between visual signals and language comprehension to provide a strong base for sign language translating systems.

VI. Neural Network Architecture Design & Implementation

To recognize American Sign Language (ASL) signs from the landmark data, we developed a deep learning model based on the Transformer Encoder architecture. The model takes as input a sequence of hand, face, and body keypoints extracted from videos and classifies the entire sequence into one of 250 ASL sign classes.

The processing began with an embedding block, implemented as a multi-layer perceptron (MLP). This block includes several layers of linear projections, layer normalization and ReLU activations. It first projects the raw landmark feature vectors from the original input size to a higher-dimensional space, allowing the network to extract richer spatial representations. It then reduces the dimension back to the original size to ensure the compatibility with the Transformer stack while preserving the learned features.

To retain the ordering of the input sequence, positional encoding was added to the embedded vectors. Additionally, a special classification token (CLS) is prepended to the sequence. This learnable vector serves as a summary token, allowing the network to focus its attention on producing a holistic representation of the entire sequence.

The sequence was then passed through a stack of Transformer Encoder layers. Each layer contains a multi-head self-attention mechanism followed by a position-wise feed-forward network. The self-attention component enabled the model to attend to all frames in the sequence simultaneously, capturing both short-term interactions (example, hand shape changes) and long-range dependencies (example, coordinated hand motion across time). The feed-forward layers applied the non-linear transformations, while dropout regularization was used for reducing the overfitting. The number of encoder layers used were 2 and attention heads used were 4 during our training process.

After processing, only the final CLS token output was extracted from the Transformer. This token now has the entire sequence's semantic content. It is then passed through a fully connected (linear) output layer that maps the CLS vector to a 250-dimensional space, corresponding to the number of ASL sign classes. The output is a probability distribution over all possible signs, and the class with the highest score is selected as the model's prediction. As illustrated in Figure 6 , this architecture enables end-to-end recognition of sign language from raw landmark inputs.

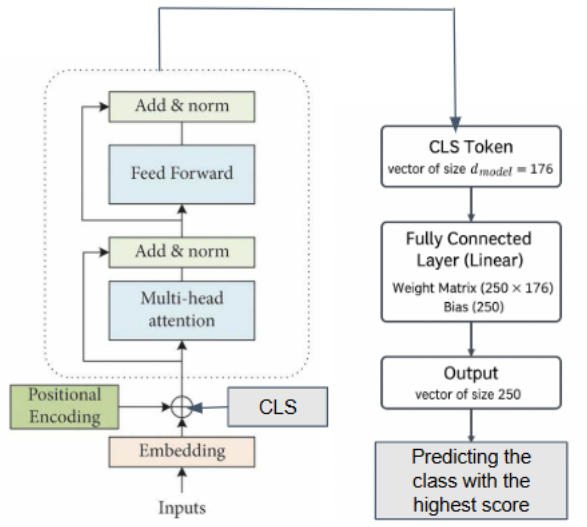

Fig. 6. Sign Encoder-based Transformer Architecture

Our Transformer Encoder architecture is particularly well-suited to this task due to its ability to handle variable-length sequences through self-attention and leverage parallel computation. Compared to traditional recurrent or convolutional models, the Transformer model proved to be better at understanding how movements change over time and recognized the complex hand and body shapes, which made it a powerful framework for ASL sign classification.

VII. Training and Evaluation

A. Training

For ASL Gloss prediction we trained Transformer Encoder on GISLR dataset. All the components of this Encoder were built using the Pytorch framework. We used Cross Entropy Loss with label smoothing and class weights to compute the loss and applied gradient clipping (max norm = 0.5) to stabilize training. The model parameters were updated using the AdamW optimizer, while a Cosine-Annealing warm restarts scheduler dynamically adjusted the learning rate from an initial value of 0.0005. This training loop was executed for 500 epochs to ensure convergence and optimal generalization. The train and validation loss Figure 7 showed that both train and validation dropped eraly at the beginning of the training due to learning rate. Both the loss curves showed oscillation due to OneCycleLR learning rate. However, validation loss was significantly fluctuating and model is slightly overfitting

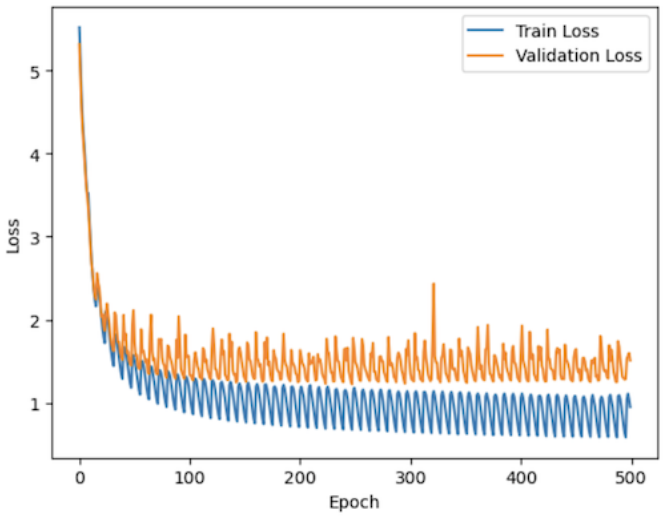

Fig. 7. Losses for GISLR dataset training

We performed hyperparameter tuning using Optuna framework [5] and performed 20 trials to find best hyperparameters for the model. Based on the trial result we ran the training with the parameters of 2 embedding layers, 4 encoder layers, 11 attention heads, and a dropout rate of 0.318 for 300 epochs. The hyperparameter tuned model achieved a training accuracy of 71.5% and a validation accuracy of 64.9% along with 1.5 validation lossTable II .

TABLE II
Model performance comparison between Base model and Hyperparameter Tuned model

	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss
Base model	0.757	0.947	0.651	1.51
Hyperparameter tuned model	0.715	1.1	0.649	1.5

Sign to Gloss model achieved mean per class accuracy of 72.81% on train dataset and 62.72% accuracy on test dataset

for classifying Sign Glosses. Few gloss classes achieved over 90% class accuracy as shown inTable III .

TABLE III
AVERAGE PER GLOSS CLASS ACCURCAY FOR TRAIN AND TEST DATA

Train Set	Mean per Class accuracy	Test Set	Mean per Class accuracy
bug	93.9234	fireman	97.6103
callonphone	90.0864	Icecream	89.9885
brother	87.5508	high	87.0502
sick	89.6268	bird	84.4794
better	87.3155	clown	83.4873

Sentence Generation: For Sentence generation from predicted Sign glosses, we utilized GPT-4 [17] instead using a traditional decoder model. American Sign Language sentences follow different syntactic patterns unlike regular English grammar, instead they follow topic-comment structure or subject-verb-object structure [10] . Keeping these in mind we experimented with a few prompts. Also, for sentence generation we kept the word limit to 5. Our goal was to achieve sentence level interpretation with small sentences.

B. Evaluation

Sign to Gloss: The trained Transformer-based model was evaluated on a held-out test set using both sign classification accuracy and natural language generation quality. We used Word Error Rate (WER) [6] [3] metric to evaluate sign to gloss predictionFigure 8 . For a single word, our model achieved 0% WER. For multiple gloss prediction, we used levenshtein distance and achieved 72.85% WER.

```
Collected 10 unique correct predicted words after 204 attempts.

Summary of correct predictions:
    Sample Index Predicted Word Ground Truth Word
0           1580           nuts             nuts
1           3738          uncle            uncle
2           7892         please           please
3           8205         flower           flower
4           5783        because          because
5           5926             if               if
6           5014          black            black
7            636         pretty           pretty
8           3284            gum              gum
9           3907    glasswindow      glasswindow
```

Fig. 8. Sample of 10 correct Gloss prediction on test data

Gloss to Text: For GPT-based sentence generation, we evaluated sentence quality using BLEU-1 and BLEU-4 [12] scores. We configured the ground truth sentence based ASL grammer and experiemented with various prompts to generate sentences using predicted Glosses from the Sign Table IV .

Our system achieved a state-of-the art BLEU-1 score of 70.71 and BLEU -4 score of 13.41 outperforming Camgoz et al.'s text-to-text based Gloss2Text translation performance (50.69 BLEU-1) and surpassing Sincan et al.'s (Spotter + GPT) [14] sentence generation performance (9.12 BLEU-4). These results Table V established the effectiveness of our two-stage approach for sentence level interpretation from a limited number of Isolated Signs.

TABLE IV
SENETENCE GENERATION OUTCOME

Predicted Gloss	GPT-4 Prompt
['fast', milk'. "empty' "aunt']	f"You are creating a meaningful, natural-sounding sentence suitable for a sign language performance. f"Use ONLY these words: {', ".join(predicted _words)}. f"Make the sentence visual, imaginative, and easy to express with body language. f"Do NOT introduce new words, but you can rearrange or repeat words creatively if needed. f"Keep it short and vivid."
Generated Sentence	Aunt fast empty milk.
BLEU Score	0.7071

TABLE V
OUR SIGN TO GLOSS MODEL AND GLOSS TO TEXT INTERPRETER SYSTEM EVALUATION COMPARED TO PREVIOUS WORKS

Model	**Dataset**	**WER**	**BLEU-1**	**BLEU-4**
S2G (Camgoz et al.,2020)	Phoenix2014T	24.88	NA	NA
Our Sign Transformer	GISLR	72.85	NA	NA
Sign2(Gloss+Text) (Camgoz et al.,2020)	Phoenix2014T	NA	47.26	22.38
Spotter+GPT (Sincan et al.,2024)	DGS20	NA	38.25	9.12
Transformer+GPT4 (Gloss+Text)	GISLR	NA	70.71	13.41

VIII. TECHNICAL CONTRIBUTION

The proposed two-staged modular pipeline for interpreting American Sign Language into spoken-level English sentence offers enhanced gloss prediction quality along with significantly reduced training time. Transformer Encoder-based gloss prediction model paired with GPT-4 prompting avoids computationally heavy Continuous Sign Language Transformers (CSLR-SLT) [4] or Spotter + GPT pipelines while ensuring gloss prediction and sentence-level accuracy. Our base model is trained on Nvidia GeForce RTX 4090 GPU for 17 hours which is considerably lesser than continuous SLT training time. Since we trained our model with Mediapipe landmark data instead RGB video data, our system is deployable on standard hardware, making it suitable for real-time application.

IX. COMMUNITY CONTRIBUTION

According to the Kansas Department of Health, approximately 33 babies are born with permanent hearing loss every day in the U.S. Early exposure to sign language can significantly reduce the risk of Language Deprivation Syndrome, a condition that can severely impair cognitive and social development [18] . Our sign language interpreter offers a valuable tool for deaf children and their hearing parents to learn ASL together and establish effective communication. American Sign Language is not only essential within the deaf community but is also increasingly adopted by individuals

who experience challenges with vocal communication. A real-time ASL interpreter like ours empowers these communities to interact independently, eliminating the reliance on human interpreters and promoting greater accessibility and inclusion.

X. Conclusion

This project involved an end-to-end complete system for sign language video translation into grammatically consistent natural language sentences. Beginning with raw video input, we utilized a robust feature extraction pipeline via Mediapipe to track hand, body, and facial keypoints. These key points were preprocessed, organized, and utilized for training a Transformer model, which successfully predicted glosses — sign language's intermediate linguistic representation.

To maintain precision, we incorporated metrics of evaluation like Word Error Rate (WER) for gloss prediction and BLEU Score for sentence generation. The predicted glosses were translated into grammatical sentences with the help of a GPT-based language model, culminating the translation. The model was trained and tuned with sophisticated deep learning methods, such as hyperparameter tuning, learning rate scheduling, and gradient clipping, leading to an effective and stable training process. This project illustrates the potential to integrate computer vision and NLP to develop intelligent, multimodal sign language translation systems. This project helps enhance deaf and hard-of-hearing community accessibility and communication and acts as an excellent foundation for future real-time sign language interpretation improvement.

XI. Appendix

We performed real time inferencing using a CPU system with a web camera. For inferencing we limited the words in the sentence to 5. System captured and recognized each gesture and at the end it generated an entire spoken level sentence.

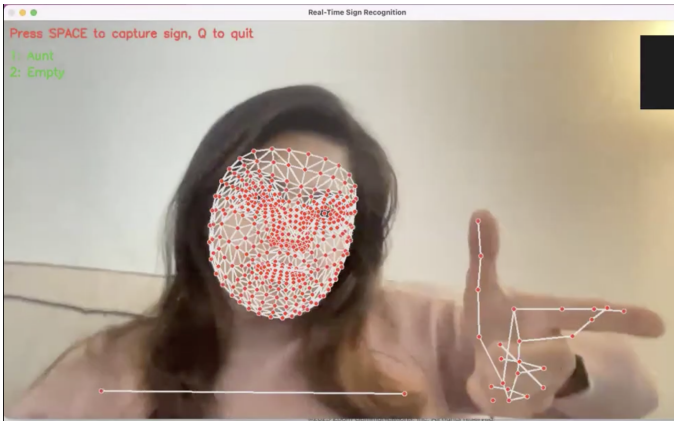

Fig. 9. Real-time Isolated Sign recognition

Our application captured signs and identified the glosses as per signs. Then it pass the recognized sign to GPT-4 prompt and interpret the spoken-level English sentence based on Sign Grammer.

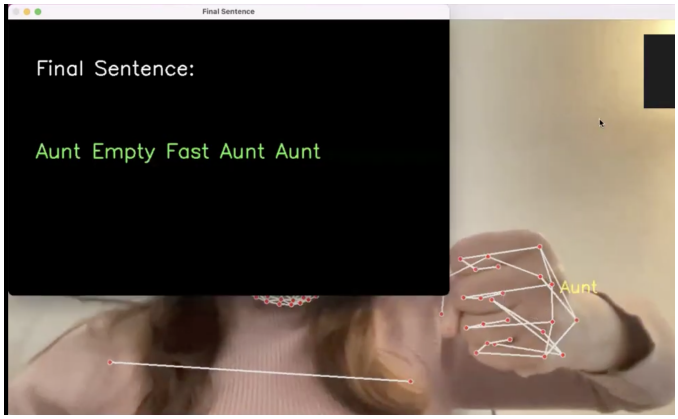

Fig. 10. Real-time Sign Language Interpretation by our application

References

[1] K. Yin and J. Read, "Better Sign Language Translation with STMC-Transformer," in *Proc. 28th Int. Conf. Computational Linguistics (COLING)*, 2020, pp. 5975–5989.

[2] M. Boháček and M. Hrúz, "Sign Pose-based Transformer for Word-level Sign Language Recognition," in *Proc. IEEE/CVF Winter Conf. Applications of Computer Vision (WACV) Workshops*, 2022, pp. 182–191.

[3] C. Morris, V. Maier, and P. Green, "From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition," in *Proc. 8th Int. Conf. Spoken Language Processing (INTERSPEECH)*, 2004, pp. 501–504.

[4] N. C. Camgoz, O. Koller, S. Hadfield, and R. Bowden, "Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 020–10 030.

[5] O. Team, 2025. [Online]. Available: https://github.com/optuna/optuna (accessedMay10

[6] J. P. Woodard and J. T. Nelson, "An information theoretic measure of speech recognition performance," in *Workshop on Standardisation for Speech I/O Technology*, 1982.

[7] A. Alnuaim, M. Zakariah, W. A. Hatamleh, H. Tarazi, V. Tripathi, and E. T. Amoatey, "Human-Computer Interaction with Hand Gesture Recognition Using ResNet and MobileNet," *Computational Intelligence and Neuroscience*, vol. 2022, no. 8777355, 2022.

[8] D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A. Gil-González, and J. M. Corchado, "DeepSign: Sign language detection and recognition using Deep learning," *Electronics*, vol. 11, no. 11, pp. 1780–1780, 2022.

[9] C.-Y Lin and F. J. Och, "ORANGE: a method for evaluating automatic evaluation metrics for machine translation," in *Proc. 20th Int. Conf. Computational Linguistics (COLING)*, 2004, pp. 501–507.

[10] A. Bloom, 2025. [Online]. Available: https://www.aslbloom.com/blog/asl-sentence-structure\#:\textasciitilde:text=ASL\%20uses\%20a\%20topic\%2Dcomment

[11] B. Fang, J. Co, and M. Zhang, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1802.07584

[12] K. Papineni, S. Roukos, T. Ward, and W.-J Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Computational Linguistics (ACL)*, 2002, pp. 311–318.

[13] Kaggle, 2025. [Online]. Available: https://www.kaggle.com/competitions/asl-signs/discussion/395380

[14] O. M. Sincan, N. C. Camgoz, and R. Bowden, 2024. [Online]. Available: https://arxiv.org/abs/2403.10434

[15] S. Sharma and S. Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," *Expert Systems with Applications*, vol. 182, no. 115657, 2021.

[16] Google, "MediaPipe Solutions Guide," *Google AI*, 2025.

[17] S. Yenduri, "GPT (Generative Pre-Trained Transformer)-A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions," *IEEE Access*, vol. 12, pp. 54 608–54 649, 2024.

[18] 2025. [Online]. Available: https://www.kdhe.ks.gov/887/Hearing-Loss-Facts