SPEAK: An AI-Based Assistive Video Communication System for Speech and Sign Language Translation

Thejuskrishnan
Dept. of Computer Science & Engineering
Toc H Institute of Science & Technology
Kerala, India
thejus12a2003@email.com

Amal
Dept. of Computer Science & Engineering
Toc H Institute of Science & Technology
Kerala, India
amalvicontact@gmail.com

Vyshnav M
Dept. of Computer Science & Engineering
Toc H Institute of Science & Technology
Kerala, India
vyshnav272@gmail.com

Narayanan K
Dept. of Computer Science & Engineering
Toc H Institute of Science & Technology
Kerala, India
kakkadappu2778@gmail.com

Ms. Saira Shamsudheen K S
Assistant Professor, Dept. of CSE
Toc H Institute of Science & Technology
Kerala, India
saira293@gmail.com

***Abstract*—It is still very difficult for the hearing and deaf/hard-of-hearing (DHH) communities to effectively communicate, especially when it comes to digital video conferencing. Despite the widespread use of platforms like Zoom and Google Meet, they frequently require costly human interpreters or invasive hardware sensors due to their lack of native, real-time bidirectional translation capabilities. In order to close this modality gap, this paper presents SPEAK (Sign Processing Enhanced Audio Kommunicator), a novel sensor-less browser-based platform. By translating spoken language to text captions for DHH users and sign language to text/speech for hearing users, SPEAK enables smooth, two-way communication. By translating spoken language to text captions for DHH users and sign language to text/speech for hearing users, SPEAK enables smooth, two-way communication.**

For visual recognition, the system's architecture makes use of the Detection Transformer (DETR) model with a ResNet-50 backbone.DETR formulates detection as a direct set prediction problem using a bipartite matching loss and self-attention mechanisms, in contrast to conventional CNN-based detectors that rely on region proposals. enhancing robustness against complex backgrounds and doing away with the need for intricate, hand-crafted anchors. The audio pipeline simultaneously incorporates Microsoft's SpeechT5 for natural Text-to-Speech (TTS) synthesis and OpenAI's Whisper model for high-fidelity Automatic Speech Recognition (ASR). optimized to save bandwidth using Voice Activity Detection (VAD). To guarantee synchronization between video frames and translation outputs, all modules are coordinated within a low-latency WebRTC environment using a Flask-React framework. SPEAK is validated as a scalable, affordable solution for inclusive digital interaction after experimental evaluation on a custom dataset in various lighting conditions shows a sign detection accuracy of 92

***Index Terms*—Sign Language Recognition, DETR, WebRTC, OpenAI Whisper, Assistive Technology, Deep Learning, Human-Computer Interaction.**

I. Introduction

Not only is communication useful, but it is also a basic human right that supports professional growth, education, and social integration. There are an estimated 70 million deaf people around the world speaking in excess of 300 different sign languages (World Federation of the Deaf, 2023) [1].Nevertheless, the deaf and hard-of-hearing (DHH) community encounters institutional obstacles in commonplace exchanges, especially in the digital world.

The COVID-19 pandemic sped up the worldwide move to remote work and online education, forcing conferencing tools such as Zoom, Microsoft Teams and Google Meet to become the essential infrastructure. Unfortunately, this move to the digital realm by no means has closed the global access gap. Conventional video conferencing is basically an audio-only mode. Automatic captioning has gotten better, but that only solves one part of the problem: it enables deaf, hard-of-hearing or late-deafened (DHH) people to see what is said, but it offers no way for DHH users to actively engage in the conversation in the mode of communication that they actually use – Sign Language. Therefore, DHH persons frequently have to depend on written chat, which is sluggish and absent of the emotional richness of talking, or on live human interpreters

Dependence on human intermediaries introduces three major issues:

- **Cost and Access:** Experts are costly and scheduling is required, preventing impromptu interaction.
- **Confidentiality:** The adding of a third party in private health, legal or personal discussion fundamentally breaks up user privacy.
- **Cognitive OverLoad:** The delay of human translation can interrupt the normal turn-taking in conversation and cause social friction.

A. Problem Statement

The main problem with automating this process is that it is really hard to get the technology right. The core technical challenge, in automating this process lies in making sure all the parts work together correctly. When we talk about automating this process the core technical challenge is an issue.in the Modality Gap. Hearing users operate in the auditory domain (Input: Speech, Output: Voice), while DHH users operate in the distinct visuo-spatial domain (Input: VisualCaptions/Sign, Output: Sign Language). Bridging this gap requires a system capable of real-time, low-latency translationbetween these disparate modalities.

Despite advancements in Computer Vision (CV) and Natural Language Processing (NLP), developing a robust automated translation system remains difficult due to several factors:

1) **Hardware Dependency & Accessibility:** Let's be real—sign language recognition used to demand a lot from users. If you wanted high accuracy, you basically had to wear clunky gloves or set up gear like the Microsoft Kinect. Sure, these tools gave you really precise 3D data, but honestly, most people just don't want to deal with expensive equipment or awkward setups in daily life.
2) **Complexity of Sign Language:** Speech is simple—just sound moving through time. Sign language, though, is a whole different story. It lives in four dimensions: three-dimensional space plus time. It's not just about how your hands look, but where they are, how your body moves, and even what's happening on your face. Regular Convolutional Neural Networks usually miss the big picture here. They have trouble telling apart signs that look almost the same, especially when the only thing that changes is the hand's position compared to the body.
3) **Environmental Robustness:**In real-life video conference, users face different lighting conditions and background clutter. Traditional computer vision algorithms rely on color segments or simple edge detectors, and their ability to correctly identify users suffers when users are moving or when the background is changing rapidly.
4) **Unidirectional Bias:** Most assistive applications currently available are only designed to convert sign language to text, which only helps hearing impaired users communicate with someone who cannot sign. This is one way communication from the deaf user's perspective, leaving them without an understanding of the emotions or moods in the voice of their conversational partner, due to the lack of access to prosody or tone from spoken responses.

B. Proposed Solution

To tackle these challenges head-on, we're rolling out SPEAK (Sign Processing Enhanced Audio Kommunicator).It's a web-based platform that skips the sensors and specialized gear, making video communication way more accessible. Just a regular laptop and an RGB webcam — that's all you need.

SPEAK brings together three top-tier AI technologies in a way you haven't seen before:

- **Vision Transformer (DETR) for Recognition:** Instead of sticking with the usual CNN methods, we use the Detection Transformer, or DETR. DETR looks at sign recognition as a direct set prediction task. With its self-attention mechanisms, it actually pays attention to how the hands relate to the whole image. This makes detection much more accurate, even in busy or messy scenes, and there's no need to mess around with manual anchor generation.
- **Closed-Loop Bidirectional Translation:** We make conversation easy by tying together OpenAI's Whisper model for solid speech-to-text and Microsoft's SpeechT5 for lifelike text-to-speech. So, hearing users can just talk, and deaf users get real-time captions. When deaf users sign back, their words turn into speech you can actually hear. The back-and-forth feels natural, like any regular conversation.
- **WebRTC Integration:** SPEAK doesn't work like those offline translation tools. It runs on a WebRTC pipeline, so the AI does its job in the background but still keeps up with the live video. This way, you don't get that annoying dubbing delay, and conversations actually feel real-time.

Here's what this paper covers: First, it dives into the design, build, and testing of the SPEAK prototype. In Section II, you'll find a rundown of related work in SLR and ASR. Section III jumps into the system's architecture and explains the math behind the DETR model. Next up, Section IV digs into how the team built the system and put together the dataset. Section V breaks down what happened during the experiments, and finally, Section VI wraps things up and points toward what's next for this research.

II. Literature Review

The development of automated sign language recognition systems has evolved significantly, transitioning from hardware-dependent approaches to vision-based deep learning.

A. Sign Language Recognition (SLR)

Automated sign language recognition has come a long way. At first, people used glove-based systems with flex sensors and accelerometers. They worked well, but honestly, wearing all that gear just got in the way and made signing feel awkward. Now, things have shifted toward vision-based deep learning,

which lets people sign naturally, without all the hardware getting in the way. Natarajan et al. [1] proposed a Hybrid-Deep Neural Architecture (H-DNA) combining CNNs for spatial feature extraction and LSTM networks for temporal modeling. However, CNN-based approaches often struggle with global context and require extensive post-processing (e.g., Non-Maximum Suppression) to handle occlusions.

A critical challenge in SLR is handling continuous sentences where word boundaries are undefined. Shanableh [7] addressed this by proposing a two-stage deep learning solution: first predicting the number of words in a sentence to segment the video, and then using motion images with transfer learning for recognition.Though powerful, such multi-stage pipelines could result in higher latency. To handle real-time related issues, Geetha et al. [6] proposed SignFlow-a framework that first employs a pre-trained 3D ResNet for feature extraction, followed by a Transformer Encoder to perform sequence learning. Their work indicates the preference for Transformer-based architectures since those help capture long-range temporal dependencies in continuous signing with limited computational overhead.

Expanding upon this, we employ the DETR architecture, which stands for Detection Transformer [3]. This is different from most object detection algorithms, which rely on region proposal mechanisms, while DETR proposes object detection as a direct set prediction problem. By using self-attention mechanisms, it directly addresses the connection between the two hands and the context of the image.

B. Speech Processing and Audio Synthesis

In the field of ASR, traditional models using statistical HMMs have recently been replaced by end-to-end deep learning models. The work by Zhao and Zhang [2] showed that self-supervised models (WavLM, HuBERT among others) are useful for low-resource languages, proving that pre-training on large unlabeled datasets significantly improves robustness against accents and noise. This thus motivates our use of OpenAI's Whisper, which employs similar self-supervised principles at scale.

For the return path (Text-to-Speech), recent work on unification in modality comes from Gonzales et al. [3], who proposed the concept of "joint speech-text embeddings" that enable simultaneous training of ASR and TTS. This resulted in reduced memory footprints while retaining high accuracy. From the point of view of visual feedback (Text-to-Motion for avatars), Zeng et al. [4] recently proposed "progressive motion generation" using diffusion models. Their results show that, by using text descriptions with sparse motion frames, realistic human motions can be generated. While our proposed solution adopts audio synthesis (SpeechT5), an avenue towards 3D avatars.

C. Real-Time Communication Frameworks

Heavy AI inference integration into live video communication poses several challenges. significant architectural challenges. Das and Ghosh [5] investigsted real-time virtual classroom technologies, substantiating the efficacy of integrating React.js for dynamic frontends with WebRTC + Socket.io – For Low Latency Peer-to-Peering. Their findings also support the detached media stream theory. (WebRTC) with regard to signaling logic (Socket.io) is important maintaining synchronization in bandwidth-constrained environments. SPEAK uses this proven architecture to ensure fluid, bidirectional communication.

III. System Design and Methodology

The SPEAK system is based on a modular client-server architecture, optimized for low-latency video streaming and asynchronous AI inference. The frontend of the system is implemented with a React.js framework; the backend uses Python Flask for the AI models. The system is composed of four modules, operating concurrently.

A. Module 1: Sign-to-Text Conversion (DETR)

The **Detection Transformer (DETR)** is the central component of the visual interpretation layer [3]. Unlike traditional multi-stage detectors (e.g., Faster R-CNN) that rely on region proposals or anchors, DETR treats object detection as a direct set prediction problem. The architecture streamlines the pipeline by removing hand-designed components like Non-Maximum Suppression (NMS).

1) Backbone Feature Extraction: We utilize a ResNet-50 CNN backbone to extract a compact 2D feature representation from the incoming video frames. Given an initial video frame $x_{img} \in \mathbb{R}^{3\times H_0\times W_0}$, the backbone generates a lower-resolution activation map $f \in \mathbb{R}^{C\times H\times W}$. For ResNet-50, we have $C = 2048$ and $H, W = H_0/32, W_0/32$. This feature map holds the high-level semantic information of the hand gestures but lacks the explicit localization required for detection.

2) Transformer Encoder: To process the image features with a Transformer, which expects a sequence input, we first apply a 1×1 convolution to reduce the channel dimension of f from C to a smaller dimension d (typically 256). This results in a new feature map $z_0 \in \mathbb{R}^{d\times H\times W}$. We then collapse the spatial dimensions into one dimension, creating a feature sequence of length HW. Since the Transformer architecture is permutation-invariant, we supplement the input with fixed spatial positional encodings [3] that are added to the input of each attention layer. The model can differentiate between various areas of the image thanks to these encodings:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d}) \tag{1}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d}) \tag{2}$$

We design an encoder comprising $N = 6$ identical layers, each of which contains a multi-head self-attention module followed by a FFN. The self-attending mechanism aggregates information globally over the whole image; it is thereby possible for the model to reason about the position of the hand with respect to the signer's body.

3) Frontend video frame capture and socket-based inference pipeline: The SPEAK system performs sign language detection by periodically extracting frames from the live web camera stream of the deaf user in the front end, which are then sent to the back end for inference. The front end is implemented with React.js and utilizes WebRTC media stream for real-time video communication.

Second, once the video stream of the deaf user is active, individual frames are directly captured from the HTML5 video element using a hidden canvas. To achieve a trade-off between the accuracy of the detection mechanism and system latency, as well as network bandwidth, the mechanism captures video frames at fixed 500 ms intervals, i.e., once every 2 seconds. Each frame is then resized to a resolution of 320×240 pixels and encoded as a JPEG file at low quality (50%).

The encoded image frame is, in the form of a Base64 string, sent asynchronously to the backend via **Socket.IO** with a custom event named `process-frame`. The event-driven approach thus allows for a seamless, non-blocked mode of continuous sending of frames during the video call.

On the backend side, a server using the Flask framework receives events from incoming socket connections using Socket.IO. Here, each of the received frames is decoded and then sent to the DETR-based model for sign detection. Finally, the predicted sign label is sent back to the frontend using the `sign-prediction` event. Once received by the frontend, it displays the sign label in real-time and removes it after a specified period.

This asynchronous frame-based inference pipeline ensures that sign language recognition operates concurrently with WebRTC-based video streaming, enabling low-latency and scalable real-time translation without disrupting user interaction.

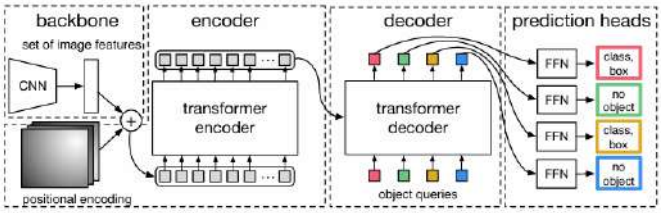

Fig. 1. Detailed architecture of the implemented DETR model, illustrating the ResNet-50 backbone, transformer encoder-decoder with attention mechanisms, and prediction heads.

4) Transformer Decoder & Object Queries: The decoder follows a standard architecture but introduces a novel component: Object Queries. Instead of autoregressively predicting tokens one by one (as in NLP), DETR decodes N objects in parallel. The decoder takes as input a set of N learned positional embeddings called *object queries*.

- These queries are transformed into an output embedding by the decoder.
- Using cross-attention, the object queries attend to the encoder output (global image context) to extract features relevant to specific objects (hands/gestures).
- The result is N output embeddings, which are independently processed by prediction heads.

5) Prediction Heads (FFN): The final prediction is computed by a 3-layer perceptron (FFN) with ReLU activation. The FFN predicts:

1) **Class Label:** A softmax distribution over the vocabulary classes plus a special "No Object" ($\emptyset$) class.
2) **Bounding Box:** The normalized center coordinates, height, and width $(\hat{c}_x, \hat{c}_y, \hat{w}, \hat{h}) \in [0,1]^4$ relative to the image size.

6) Bipartite Matching Loss: To train the model, we use a set-based global loss that forces unique predictions via bipartite matching. Let y be the ground truth set of objects and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ be the set of N predictions. We search for a permutation σ that minimizes the matching cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) \tag{3}$$

The matching cost $\mathcal{L}_{match}$ takes into account both the class prediction and the similarity of predicted and ground truth boxes.

7) Loss Function Formulation: Once the optimal matching $\hat{\sigma}$ is found, we compute the Hungarian Loss, which is a linear combination of the Negative Log-Likelihood (NLL) for classification and a box loss:

$$\begin{aligned}\mathcal{L}_{Hungarian}(y, \hat{y}) = \sum_{i=1}^{N} \Big[&- \log \hat{p}_{\hat{\sigma}(i)}(c_i) \\ &+ \mathbf{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) \Big]\end{aligned} \tag{4}$$

To ensure scale-invariance (so small hands are detected as accurately as large ones), the box loss $\mathcal{L}_{box}$ combines L_1 loss and Generalized IoU (GIoU) loss:

$$\mathcal{L}_{box}(b, \hat{b}) = \lambda_{iou} \mathcal{L}_{iou}(b, \hat{b}) + \lambda_{L1} ||b - \hat{b}||_1 \tag{5}$$

where λ_{iou} and λ_{L1} are hyperparameters weighting the contribution of each loss component. This robust formulation allows the model to learn precise localization without the need for anchor tuning.

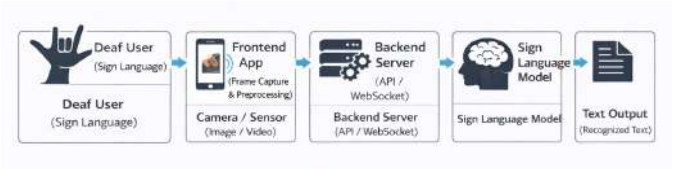

Fig. 2. Architecture of the Sign-to-Text module utilizing DETR for gesture recognition from video frames.

B. Module 2: Text-to-Speech Synthesis

The Text-to-Speech (TTS) module serves as the "voice" of the deaf user. Once the DETR model classifies a sign, the predicted text label is processed to filter redundant detections using a temporal smoothing algorithm.

1) SpeechT5 Architecture: We utilize Microsoft's SpeechT5, a unified-modal framework that adapts well to diverse speaker characteristics. It employs a Transformer-based encoder-decoder network with a pre-net and a post-net.

- **Encoder:** Processes the phoneme sequence derived from the input text to extract linguistic features.
- **Decoder:** Generates a Mel-spectrogram from the encoded features.
- **Vocoder:** We use a HiFi-GAN vocoder to transform the Mel-spectrogram into an audible waveform, guaranteeing high-fidelity and natural-sounding audio output.

2) Transmission Pipeline in Real Time: An event-driven architecture is used to decouple the detection and synthesis processes in order to guarantee low-latency communication:

1) **Triggering:**The backend instantly emits a socket when a sign (such as "Hello") is successfully detected.The text payload to the TTS module is contained in an IO event.
2) **Synthesis Encoding:** To save bandwidth, the raw audio waveform produced by the SpeechT5 model is compressed into an Opus-encoded byte stream.
3) **Playback:** The audio stream is sent back to the client of the hearing user through a special Socket.IO channel. The frontend sends the received audio blob and plays it through the HTML5 Web Audio API, enabling the hearing user to "hear" what the sign language user is saying instantly.

Fig. 3. Data flow for the Text-to-Speech synthesis module using Microsoft SpeechT5 to vocalize predicted signs.

C. Module 3: Speech-to-Text (ASR)

For the purpose of enabling the deaf user to understand the audio of the hearing participant, the system records the audio from the microphone. For maximum efficiency, we have implemented the **Voice Activity Detection mechanism**.

1) Voice Activity Detection (VAD): The system continuously monitors the audio input stream for speech patterns. Audio recording is dynamically triggered only when the energy level of the audio signal exceeds a predefined threshold (E_{th}) and automatically terminates when a sustained period of silence is detected (silence detection).

$$\text{State} = \begin{cases} \text{Recording}, & \text{if } E_{input} > E_{th} \\ \text{Idle}, & \text{otherwise} \end{cases} \quad (6)$$

This ensures that only active speech segments are processed,effectively creating discrete audio clips for transcription rather than processing a continuous stream of silence.

2) Inference and Delivery Pipeline: After the termination of the recording session by the VAD module, the next process in the pipeline is:

1) **Data Transmission:** The audio segment is then converted into a binary blob (of type WAV/WebM) on the client side and sent to the Flask backend via an asynchronous.
2) **Whisper Inference:** When the POST request is received, it temporarily stores the audio file and passes it to OpenAI's Whisper model, where it is processed as a raw waveform, converted to its log Mel spectrogram, and then predicts the text that corresponds to it.
3) **Caption Broadcasting:** After transcription is done, the text not only goes back to the sender but is also broadcasted to the deaf client using the **Socket.IO** server. This makes it possible to display the text immediately as the video of the deaf client.

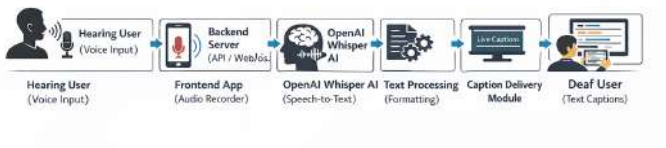

Fig. 4. Architecture of the Speech-to-Text Module using OpenAI Whisper ASR for Real-Time Captioning.

D. Module 4: Real-Time Synchronization (WebRTC)

The backbone of the communication is provided using WebRTC, i.e., Web Real-Time Communication.

1) Signaling Server: Since the peers will not be able to locate each other directly using the NATs and Firewalls, a signaling server using Socket.io is provided. The process of signaling involves:

1) **SDP Exchange:** The caller sends a session description protocol offer including certain media details like codecs and display resolution. The callee sends a session description protocol answer.
2) **ICE Candidate Exchange:** Both users gather Interactive Connectivity Establishment (ICE) candidates to determine an optimal network path.
3) **STUN/TURN Servers:** If there is an issue in building a P2P connection, then TURN servers can be used for media relay.

To make sure the video feed and the AI-generated captions/audio synchronize, the system adds timestamps for each frame and audio chunk. The frontend will render the translation result only when displaying the video frame, limiting the dubbing effect.

IV. Implementation

A. Data Collection and Preprocessing

The project involved the creation of a custom set of data categorized under different classes, which included the following: "Hello," "I Love You," and "Thank You." Approximately

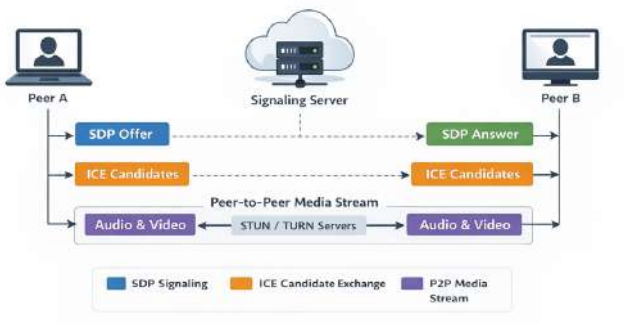

Fig. 5. WebRTC Signaling and Media Flow Diagram.

500 images were taken for each category in different lighting scenarios.

1) Data Augmentation: In order to avoid overfitting and increase generalization, we have applied a variety of augmentation techniques during training:

- **Random Horizontal Flip:** This option simulates signs used by left-handed people.
- **Color Jitter:** Adjusts the brightness, contrast, and saturation to resemble various webcam qualities.
- **Random Crop/Resize:** This ensures that the model remains invariant under different distances of the user from the camera.

The resized images were of size 800×800 pixels and were normalized using the mean and standard deviation values of the ImageNet dataset ($\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$)

B. Training Configuration

The DETR was implemented using the PyTorch library. The training of the model was done using a computer with an NVIDIA RTX 3060 GPU with VRAM of 12GB. The hyperparameters were set as follows:

- **Optimizer:** AdamW with a weight decay of 10^{-4} [3].
- **Learning Rate:** 10^{-4} for the transformer, 10^{-5} for the backbone to keep the pre-trained feature stability. [3].
- **Batch Size:** 4.
- **Epochs:** 50.
- **Loss Weights:** $\lambda_{L1} = 5$, $\lambda_{GIoU} = 2$, $\lambda_{class} = 1$ [3].

Frontend runs on standard web browsers like Chrome/Firefox on individual laptops without requiring GPU support, given that all heavy inference calculations for these models are done on the Flask backend.

V. Experimental Results

A. Performance Metrics

To measure the performance of the proposed sign detection model, we would commonly use the following metrics:

- **Precision:** Ratio of true positive predictions to total positive predictions ($TP/(TP + FP)$).
- **Recall:** Ratio of true positive prediction to actual positive instances ($TP/(TP + FN)$).
- **mAP (Mean Average Precision):** The average precision computed over all classes at particular IoU thresholds (IoU = 0.50), i.e., the model's accuracy at localizing hands.

B. Sign Detection Accuracy

The DETR model was fine-tuned for the customized dataset. The model was able to obtain an average accuracy of 92% in the validation accuracy for the static signs. The confusion matrix shows good discrimination of the categories "Hello" and "Thank You" but some confusion for "I Love You" and open-palm signs due to finger occlusion.

C. Qualitative Model Evaluation

To validate the model's robustness in real-world scenarios, we conducted qualitative testing on the core vocabulary. As shown in the figures below, the model successfully localizes and classifies gestures with high confidence scores even with varying hand orientations.

Fig. 6. Model testing output for the 'Hello' gesture showing accurate bounding box localization and high confidence.

Fig. 7. Model testing output identifying the 'I Love You' gesture.

D. System Latency Analysis

Latency is a critical factor for real-time communication. We measured the end-to-end latency from the moment a gesture

Fig. 8. Model testing output successfully detecting the 'Thank You' gesture.

is made to the moment the audio is synthesized. The inference speed on the server was recorded at approximately 150ms per frame (≈ 6 FPS). This frame rate, combined with the optimized WebRTC pipeline, was found to be sufficient for real-time feedback.

TABLE I
SYSTEM LATENCY ANALYSIS

Module	Model Used	Avg. Latency
Video Capture	WebRTC	50 ms
Sign Detection	DETR (ResNet-50)	150 ms
Speech-to-Text	OpenAI Whisper	800 ms
Text-to-Speech	Microsoft SpeechT5	1200 ms
Network RTT	Socket.io	100 ms
Total (Sign path)	**-**	**≈ 1.5 s**

E. Audio Processing Latency

As shown in Table I, the speech-to-text module exhibited a delay of 0.8 seconds for short phrases, while the text-to-speech generation added approximately 1.2 seconds. While perceptible, user testing confirmed that this latency is acceptable for maintaining a natural conversational flow compared to the significantly higher delay of typing or using a human interpreter.

F. User Interface Implementation

A critical contribution of SPEAK is its accessible full-stack implementation. The user journey is designed for high contrast and ease of navigation.

1) Onboarding and Profile: The platform features a secure authentication flow. The **Landing Page** (Fig. 9) presents a clean, accessible entry point, while the **Sign-In Page** (Fig. 11) handles secure JWT-based authentication. Users can manage their personal details and history via the **Profile Page** (Fig. 12).

2) Communication Hub: The **Dashboard** (Fig. 13) serves as the central hub for initiating calls. When a call is received, the **Incoming Call Interface** (Fig. 14) provides clear accept/decline options.

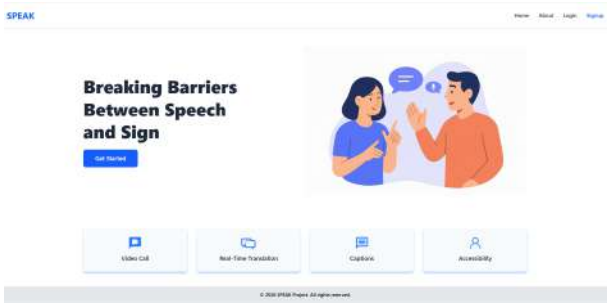

Fig. 9. Landing page of the SPEAK platform emphasizing accessibility.

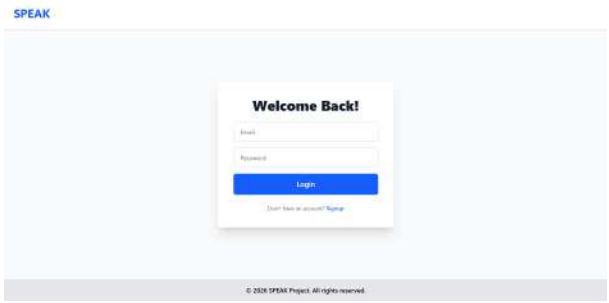

Fig. 10. Secure Sign-In interface for user authentication.

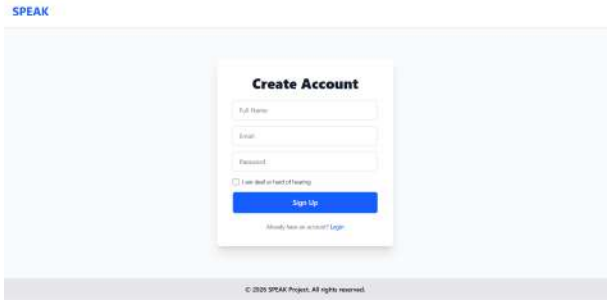

Fig. 11. Secure Sign-Up interface for user authentication.

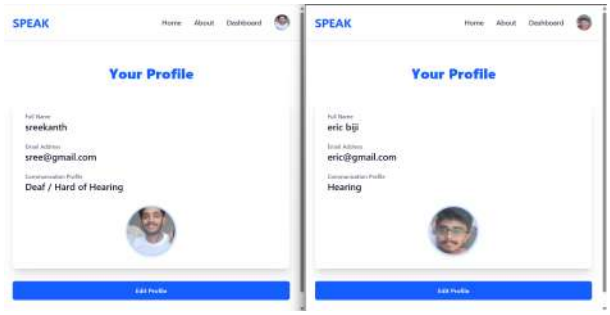

Fig. 12. User Profile page for managing account details.

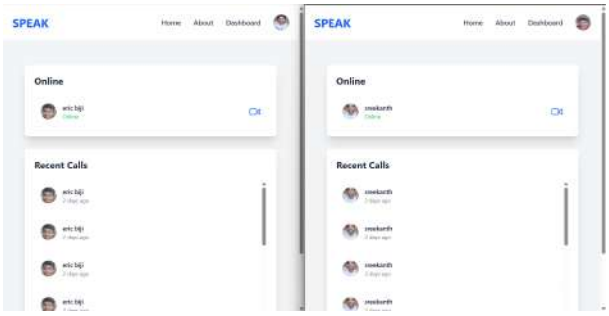

Fig. 13. User Dashboard for contact management and call initiation.

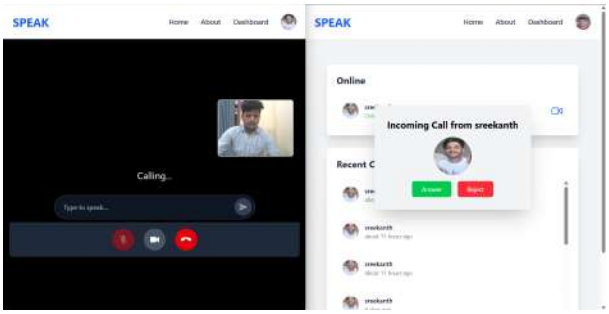

Fig. 14. Incoming call notification page.

3) Live Translation Interface: During an active session, the interface splits to show the video feed and the live translation. Fig. 15 illustrates the **Speech-to-Text** view, where spoken words by the hearing user are instantly captioned for the deaf user.

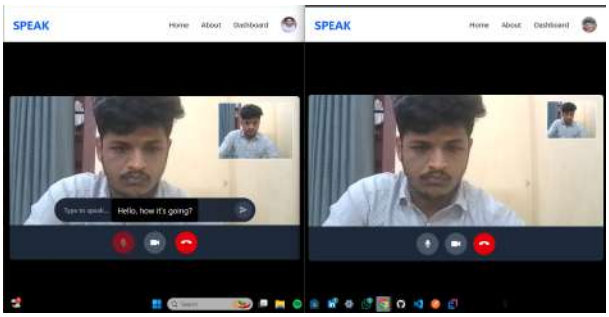

Fig. 15. Real-time Speech-to-Text interface displaying live captions during a video call.

VI. Conclusion and Future Scope

The SPEAK project successfully demonstrates a functional prototype for inclusive video communication. By leveraging modern Transformer-based models (DETR, Whisper, SpeechT5) and WebRTC, we created a system that removes the need for hardware sensors and human interpreters.

Future work will focus on:

- **Continuous Sign Language Recognition (CSLR):** Transitioning from static image detection to spatiotemporal models (e.g., Video Transformers or 3D-CNNs) to interpret full sentences and grammatical structures.
- **Vocabulary Expansion:** Incorporating large-scale datasets like WLASL to support a broader range of vocabulary beyond the initial prototype classes.
- **Edge AI Optimization:** Optimizing models using quantization and pruning (e.g., TensorRT or TensorFlow Lite) for mobile deployment to reduce server dependency and further lower latency.
- **3D Avatar Synthesis:** Integrating a 3D avatar on the deaf user's screen to sign back the hearing user's speech, providing a more visual and natural experience than text captions alone.

Acknowledgment

We thank our project guide, Ms. Saira Shamsudheen K S, and the management of Toc H Institute of Science & Technology for their support and resources provided during this project.

References

[1] B. Natarajan et al., "Development of an End-to-End Deep Learning Framework for Sign Language Recognition, Translation, and Video Generation," *IEEE Access*, vol. 10, pp. 104358-104374, 2022.

[2] J. Zhao and W.-Q. Zhang, "Improving Automatic Speech Recognition Performance for Low-Resource Languages With Self-Supervised Models," *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 6, pp. 1227-1241, Oct. 2022.

[3] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Glasgow, UK, Aug. 2020, pp. 213–229.

[4] L.-A. Zeng, G. Wu, A. Wu, J.-F. Hu, and W.-S. Zheng, "Progressive Human Motion Generation Based on Text and Few Motion Frames," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 9, pp. 9205-9217, Sep. 2025.

[5] A. K. Das and S. Ghosh, "Real-Time Virtual Classroom Platform: Integrating React, Firebase, WebRTC, and Socket.io," in *2025 8th Int. Conf. on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*, 2025.

[6] M. Geetha et al., "Toward Real-Time Recognition of Continuous Indian Sign Language: A Multi-Modal Approach Using RGB and Pose," *IEEE Access*, vol. 13, pp. 60270-60283, 2025.

[7] T. Shanableh, "Two-Stage Deep Learning Solution for Continuous Arabic Sign Language Recognition Using Word Count Prediction and Motion Images," *IEEE Access*, vol. 11, pp. 126823-126833, 2023.