POLITECNICO DI TORINO
Repository ISTITUZIONALE

Bridging Communication Gaps: A Low-Cost, Real-Time Sign Language Recognition Platform

Original
Bridging Communication Gaps: A Low-Cost, Real-Time Sign Language Recognition Platform / Buccellato, F., Natale, D., De Sio, C., Azimi, S.. - ELETTRONICO. - (2025). (11th IEEE International Smart Cities Conference Patras (GR) October 6th-9th, 2025) [10.1109/ISC266238.2025.11293256].

Availability:
This version is available at: 11583/3002674 since: 2025-09-02T10:12:17Z

Publisher:
IEEE

Published
DOI:10.1109/ISC266238.2025.11293256

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

15 September 2026

Bridging Communication Gaps: A Low-Cost, Real-Time Sign Language Recognition Platform

Federico Buccellato, Davide Natale, Corrado De Sio, Sarah Azimi
Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin, Italy
federico.buccellato@polito.it, s318967@studenti.polito.it, corrado.desio@polito.it, sarah.azimi@polito.it

***Abstract*— The communication gap between deaf and hearing individuals remains a persistent challenge, limiting participation in daily life and access to essential services. The difficulty of learning sign language and the lack of dedicated tools for translating between signed and spoken languages create barriers to full inclusion in social, educational, and healthcare settings. This paper presents a low-cost, real-time mobile application to facilitate smooth, natural bidirectional communication between spoken and signed language during video calls. The platform incorporates two dedicated translation pipelines: one that converts user gestures captured by the smartphone camera into synthesized speech, and another that transforms spoken input into animated sign language. The mobile application is powered by a backend system built on Node.js and Mediasoup for media management, with Python microservices responsible for running the translation models. Experimental results confirm that the proposed solution achieves real-time responsiveness and translation accuracy, offering a practical and scalable tool to support inclusive communication between deaf and hearing individuals.**

Keywords: Artificial Intelligence, Real-Time Communication, Sign Language Translation, Smart Cities.

I. INTRODUCTION

Bridging the communication divide between deaf and hearing individuals remains a pressing challenge in modern society, particularly within high-stakes domains such as education, healthcare, and public administration. While sign languages serve as fully formed natural languages, their limited integration into mainstream communication tools perpetuates social and linguistic exclusion for deaf communities worldwide. Addressing this gap calls for inclusive, privacy-aware, and real-time technological solutions that enable seamless interaction across different communication modalities.

This work presents a fully integrated, low-cost, real-time communication platform that enables bidirectional interaction between deaf and hearing users. At its core is a neural-powered sign language translation engine capable of translating voice or text into sign language representations and vice versa, using a combination of linguistic preprocessing, neural machine translation (NMT), and skeletal pose generation. This engine is embedded within a mobile application that serves as a communication interface, allowing users to engage in video-based interactions enhanced by real-time sign language translation features. The application is designed to function similarly to common video conferencing tools, with the key innovation being its ability to automatically interpret spoken or signed input and deliver translations in the appropriate modality: text, speech, or sign animation.

To translate speech or text into sign language, the system employs a multistage pipeline including speech-to-text conversion, text-to-gloss translation, gloss-to-pose mapping, and pose-to-video rendering. In the reverse direction, it utilizes 3D Convolutional Neural Networks (3D-CNN) to analyze real-time skeletal pose data captured via camera input, translating it into textual output and, optionally, synthesized speech. The platform supports multilingual sign language variants through the use of standardized Swiss notation and language tagging, enhancing its scalability and generalizability.

The mobile application provides an intuitive, accessible interface that encapsulates the translation engine while offering real-time, interactive communication. Special emphasis has been placed on usability, affordability, and compliance with privacy regulations.

Experimental validation on benchmark datasets, including How2Sign, WLASL, and Jester, demonstrates the platform's high translation accuracy and responsiveness in both communication directions. By merging advanced sign language processing with mobile accessibility, this platform offers a scalable and inclusive solution aimed at transforming everyday interactions between deaf and hearing individuals.

II. RELATED WORKS

Over the past two decades, a wide range of technological solutions have been developed to bridge the communication divide between deaf and hearing individuals. Early systems often relied on text-based messaging or speech-to-text transcription, enabling deaf users to participate in conversations via manual typing or automated subtitles. These approaches, however, typically supported only one-way communication and lacked non-verbal expressivity [1].

More advanced platforms have attempted to directly interpret or produce sign language using wearable sensors, depth cameras, or computer vision techniques. For instance, the SignAloud gloves project by University of Washington researchers employed sensor-embedded gloves to recognize specific hand gestures and map them to spoken words [2]. SignAll developed a multi-camera system for translating American Sign Language (ASL) into English in real time, relying on skeletal tracking and natural language processing [3]. Similarly, KinTrans offers real-time sign-to-voice translation using depth sensors and skeletal data, though these platforms typically require fixed hardware setups that reduce portability [4].

In parallel, avatar-based tools such as JASigning, SiMAX, and PAULA have focused on generating sign language from text or speech input using 3D character animations. These systems rely on formal sign transcription languages like HamNoSys or SiGML to generate synthetic sign sequences [5][6]. While useful for static content delivery (e.g., in museums or public service announcements), these tools often

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

lack support for live interaction or reverse translation (sign-to-speech).

Despite progress, these platforms generally suffer from at least one of the following limitations: unidirectional functionality (text-to-sign or sign-to-text only), hardware dependency, or limited language and dialect support. Furthermore, most are not designed for real-time use on mobile devices, which significantly reduces their accessibility and scalability in everyday communication scenarios.

On the academic front, Sign Language Processing (SLP) research has evolved from isolated sign recognition to more complex tasks such as Continuous Sign Language Recognition (CSLR) and Sign Language Translation (SLT). Early SLT systems used statistical or rule-based approaches but have since been replaced by deep learning architectures, particularly sequence-to-sequence models employing Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) [7].

Camgöz et al. [8] demonstrated that incorporating glosses, i.e., semantic labels aligned with sign language structure, improves translation performance. Transformer-based models have further enhanced this by capturing long-range dependencies and context, enabling more accurate gloss-to-text and video-to-text translation [9][10].

In terms of input representation, recent works have prioritized skeletal pose estimation over raw video. Tools such as MediaPipe and OpenPose enable anonymized, low-dimensional data extraction while preserving relevant gestural information. Pose-based models, including 3D Convolutional Neural Networks (3D-CNNs), have shown high efficacy in recognizing dynamic gestures in space and time [11][12].

On the generation side, avatar systems like JASigning and SiMAX continue to rely on rule-based translations using formal markup. However, they are largely unsuited for conversational use and do not offer bidirectional capabilities [5][6].

Our proposed platform builds on these foundations by introducing a fully integrated, real-time, bidirectional communication system for deaf and hearing users. Implemented as a mobile application, the system unifies voice, text, and sign modalities using a translation engine based on NMT, 3D-CNNs, and gloss-to-pose synthesis. It supports multilingual sign variants, anonymized pose, and fluid integration in daily communication contexts, without reliance on external hardware or pre-scripted avatars.

III. Methodology

The proposed methodology enables real-time bidirectional translation between spoken and sign language through a mobile application. The system processes both audio and visual inputs and dynamically adapts the translation pipeline based on the communication direction. When sign language is used, gestures are interpreted and converted into spoken output, while spoken input is translated into animated sign language. The output modality is adjusted according to the receiver's needs, ensuring accessible and natural interaction. The system is optimized to support low-latency and fluid communication in real-world settings.

A. Video-to-Voice Translation Pipeline

For translating sign language gestures into spoken output, we adopted a real-time processing pipeline, illustrated in Figure 1, designed to capture, interpret, and synthesize the visual communication expressed by deaf users. The process begins with continuously acquiring the video stream through the mobile device's integrated camera.

Since sign language unfolds as a continuous temporal sequence, handling gestures' time dimension is critical. To address this, the system adopts a sliding window strategy, illustrated in Figure 1, which enables the dynamic construction of the spatiotemporal representation of each gesture. The data is immediately processed and appended to this evolving representation for every acquired frame.

Once 37 frames have been collected, the complete sequence is forwarded to the next processing stage. The sliding windows are applied with a stride of 15 frames, allowing for overlapping temporal segments that preserve gesture continuity while improving detection responsiveness. The fixed window size of 37 frames was chosen to optimize the balance between temporal resolution and system latency. The video stream is sampled at 12 frames per second, so each window covers approximately 3 seconds of gesture execution. This configuration ensures that gestures are captured in their entirety without introducing noticeable delays, maintaining the fluidity required for real-time gesture interpretation.

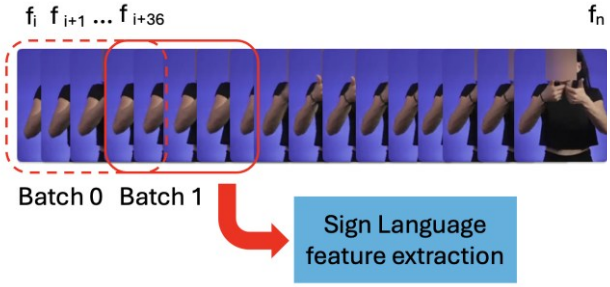

Figure 2: Representation of the sliding window process on a video.

To extract sign language information from the video frames, the system employs the open-source Mediapipe library [11], which is specifically designed for detecting and tracking human body landmarks. More precisely, this library allows the extraction of three-dimensional coordinates (x, y, z) of a predefined set of body keypoints, including hands, elbows, shoulders, and face. These keypoints form the basis

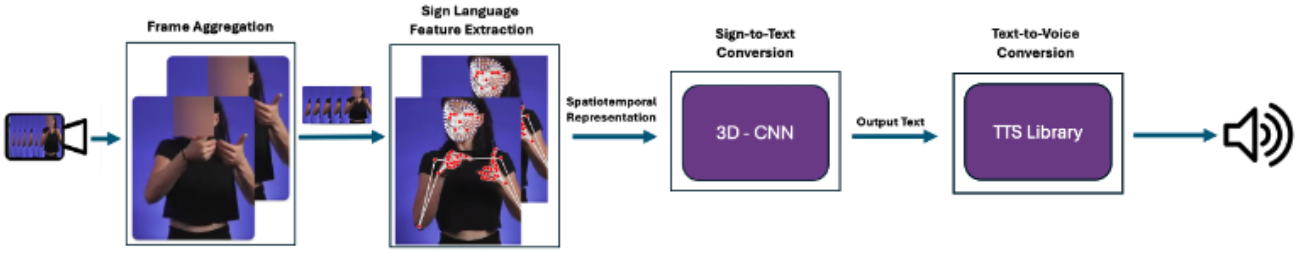

Figure 1: Schematic Representation of the Video-to-Voice Translation Pipeline.

for the skeletal representation of the gesture and are used to construct a structured map of the user's movements.

The processed and collected frames are then transformed into a compact spatiotemporal representation that preserves both the spatial configuration and temporal dynamics of the gesture. This is achieved by organizing the skeletal data into a three-dimensional array that records, for each frame, the detected joint coordinates. This representation enables the system to capture movement, orientation, and relative joint variations, elements that are fundamental for gesture understanding. The resulting representation is normalized to ensure consistency across different users, compensating for variations in body structure and camera position. The overall size of the time series is given by the following structure:

$$Timeseries = Keypoints\ x\ Spatial\ Dimensions\ x\ Frames = 576\ x\ 3\ x\ 37$$

where 576 denotes the number of keypoints extracted by the Mediapipe library, each represented by three spatial coordinates across 37 consecutive frames.

After the data collection phase, the representation is sent to a Python-based backend server, where the classification step takes place. In this stage, the sequence is provided as input to a three-dimensional convolutional neural network. The 3D-CNN receives as input the spatiotemporal representation of the sign language gesture and outputs the corresponding meaning in text format.

Once classification is completed and the gesture is converted into text, the system activates the final stage of the pipeline, speech synthesis. The textual output is processed using the TTS library, which generates a natural-sounding synthetic voice that is immediately played through the device's speaker. In this way, the gesture performed by the deaf user is made audible to the hearing interlocutor as spoken language.

B. *Voice-to-Video Translation Pipeline*

The voice-to-video translation process begins with the acquisition of spoken language through the mobile device's integrated microphone. To enable continuous and responsive operation, the incoming audio stream is segmented using a sliding window approach, with each window capturing approximately 3 seconds of speech. Each audio segment is then processed by the Whisper library, which converts the spoken input into a textual transcription. The resulting text is then passed to an open-source neural translation toolkit for sign language generation developed by [13]. This toolkit implements a multi-stage neural pipeline designed specifically for text-to-pose translation.

The first stage of their pipeline performs text-to-gloss translation, where the input sentence is converted into a sequence of glosses. Glosses are simplified, language-specific tokens that represent individual signs in a structured and unambiguous form.

Once the gloss sequence is generated, the next stage, known as gloss-to-pose translation, maps each gloss token to a corresponding sequence of body poses. These poses are encoded as three-dimensional skeletal coordinates representing the key joints involved in sign language articulation.

At this point, our system extends the open-source framework by taking the generated pose data and converting it into a visual animation. The pose sequences are printed and visualized using the PoseFormat library [14], which enables the rendering of 3D skeletal movements based on the predicted joint coordinates. These movements are then mapped onto an avatar that performs the corresponding sign language gestures, making the translation visually accessible and easy to interpret for deaf users. The final animated output is displayed directly within the mobile application interface, allowing for real-time hearing-to-deaf communication through sign language visualization. An overview of the complete voice-to-video process is illustrated in Figure 3.

C. *Mobile Application Architecture*

To ensure the proposed translation system can operate effectively in real-world scenarios, all its components have been integrated into a dedicated mobile application. This application enables users to participate in real-time video calls while automatically translating between spoken and sign language. Unlike traditional video conferencing platforms, the app is specifically designed to support accessible communication between deaf and hearing users by embedding the full translation workflow into the call. The design prioritizes accessibility, ease of use, and adherence to privacy regulations.

The overall system architecture of the mobile application is illustrated in Figure 4.

The mobile application interacts with a backend system composed of multiple services. Each service is deployed within a separate Docker container, and the containers are orchestrated using Docker Compose to ensure scalability and modularity.

At the core of the backend is a Node.js server. This server performs two key functions: user management and video call coordination. For user management, it exposes RESTful APIs to handle registration, login, and profile updates. Once a user logs in, the server generates an access token and a refresh token using the JSON Web Tokens standard. The access token is used to authorize API requests, while the refresh token allows the user to obtain a new access token once the previous one expires. To enhance security, Redis is used to

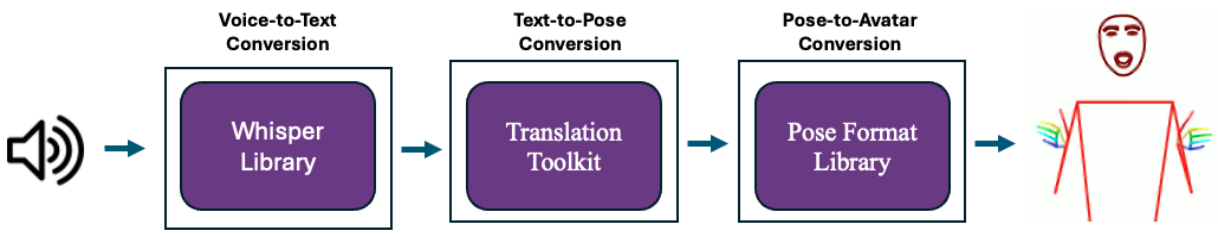

Figure 3: Schematic Representation of the Voice-to-Video Translation Pipeline.

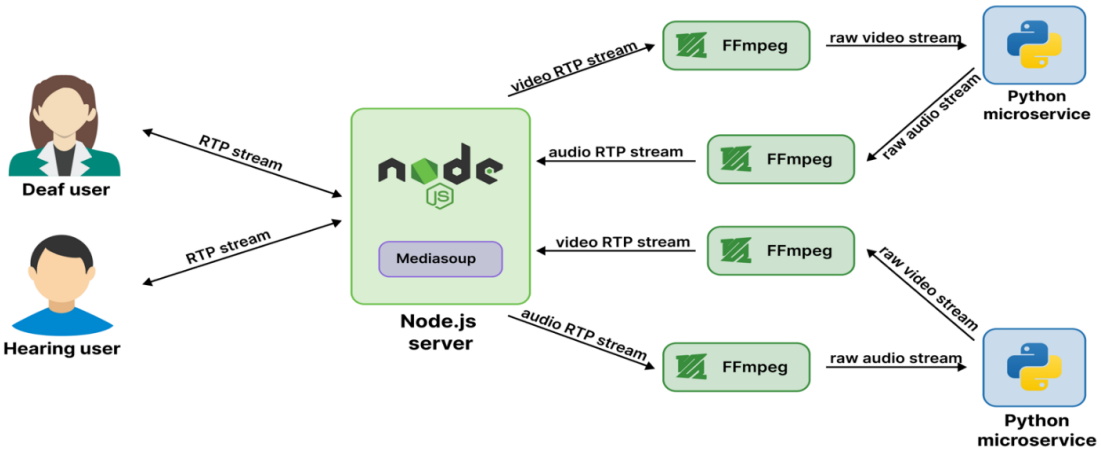

Figure 4: Architecture of the mobile application and backend translation services.

store a blacklist of invalidated tokens, ensuring that expired or revoked tokens cannot be reused.

For managing video calls, the Node.js server handles signaling through WebSocket connections and Firebase Cloud Messaging (FCM) [15]. WebSocket enables low-latency real-time communication between clients, while FCM ensures users receive notifications of incoming calls even if the app is running in the background or closed. Each user who enables push notifications registers their FCM token with the server, which stores it in a PostgreSQL database for future use.

The handling of media streams, audio and video, is managed using Mediasoup [16], a WebRTC Selective Forwarding Unit. Mediasoup receives the media streams from participants and forwards them to the other users in the call. This approach reduces bandwidth usage and ensures efficient and stable communication across different network conditions.

To enable real-time sign language translation during video calls, the system relies on two Python-based microservices. One service handles the translation from sign language to spoken output (video-to-voice), while the other processes spoken input and generates sign language animations (voice-to-video). These microservices communicate with the Node.js server through FFmpeg processes over TCP sockets.

When a video call starts, the Node.js server launches two FFmpeg processes, one for each translation direction. These processes receive RTP media streams from Mediasoup, decode them into raw audio or video data, and forward them to the appropriate Python microservice. Each microservice then applies its translation model to the incoming stream. Once the translation is completed, the result is sent back to FFmpeg, which re-encodes the output into RTP format and injects it into the ongoing call via Mediasoup.

In addition to real-time translation and video calling, the mobile application provides a number of features designed to enhance overall usability. For instance, it includes a password recovery mechanism that allows users to regain access to their account if they forget their credentials. This process involves sending a 6-digit verification code to the user's email and issuing a temporary access token, which enables the user to set a new password securely.

Users can also manage their personal profile directly within the app, including updating their name, profile picture, and adjusting preferences such as push notifications or accessibility settings. Furthermore, the application offers contact management tools that let users create, modify, or delete contacts. All changes are automatically synchronized with the backend system, ensuring consistency and a seamless experience across different devices.

This architecture enables a continuous and seamless exchange of translated content during the video call. It allows both translation services to function in parallel with the communication channel, ensuring that the user experience remains fluid and uninterrupted.

IV. Experimental Results

This section presents the evaluation and experimental results for the proposed bidirectional translation system. To perform the testing, the backend server was deployed on a machine equipped with an AMD Ryzen 5 3600 processor, 16 GB of RAM, and an NVIDIA RTX 2060 GPU. This configuration provided the necessary computational resources to execute the translation models and media processing in real time.

For the mobile-side evaluation, the application was tested on two Android smartphones, a Samsung Galaxy A04s and a Samsung Galaxy A20e. These devices were selected to represent common, low-to-mid-range hardware and to ensure the system could operate efficiently under realistic usage conditions.

The system's performance was assessed in both translation directions, from sign language to spoken output (video-to-voice) and from spoken input to sign language animation (voice-to-video).

To evaluate the effectiveness and reliability of the application, the following metrics were adopted:

Accuracy

This metric reflects the model's overall effectiveness in correctly identifying both positive and negative instances across all classes. It is computed by dividing the sum of true positives (TP) and true negatives (TN) by the total number of cases, which includes true positives, true negatives, false positives (FP), and false negatives (FN):

$$\text{Accuracy}=(TP+TN)/(TP+TN+FP+FN) \quad (1)$$

Class-specific Precision

This metric reflects the model's overall effectiveness in correctly identifying both positive and negative instances across all classes. It is computed by dividing the sum of true positives (TP) and true negatives (TN) by the total number of cases, which includes true positives, true negatives, false positives (FP), and false negatives (FN):

$$\text{Precision}=\text{TP}/(\text{TP}+\text{FP}) \quad (2)$$

Character n-gram F-score (CHRF)

CHRF is a metric used to evaluate translation quality at character level. This makes it particularly effective for assessing short or morphologically rich outputs, such as glosses in sign language translation.

CHRF calculates a balanced F-score based on character-level precision and recall, allowing it to capture small variations that may not affect word-level metrics:

$$CHRF = \frac{(1+\beta^2) * Precisionchar * Recallchar}{\beta^2 * Precisionchar + Recallchar} \quad (3)$$

where $Precisionchar$ is the percentage of correctly predicted characters out of all characters in the generated output, and $Recallchar$ is the percentage of correctly predicted characters out of all characters in the reference. The parameter β adjusts the relative importance of recall over precision. In our evaluation, CHRF was used to compare gloss sequences across different translation configurations, demonstrating the advantage of multilingual models over bilingual ones

A. Video-to-Voice Translation Results

TheVideo-to-Voice translation task is based on a gesture recognition model based on a 3D Convolutional Neural Network (3D-CNN) whose detailed structure is illustrated in Figure 5. The model was trained using the Jester dataset, a publicly available collection of over 148,000 labeled video clips, each depicting a single person performing isolated hand gestures in front of a camera. The dataset includes 27 gesture classes and was specifically designed for dynamic gesture recognition.

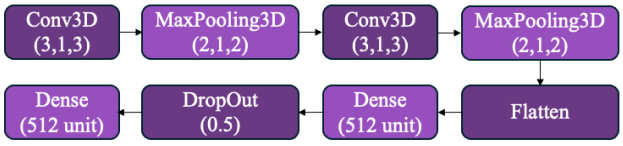

Figure 5: Architecture of the mobile application and backend translation services.

The training process was conducted across multiple epochs, and the model's performance was evaluated using standard metrics such as test accuracy and F1-score, whose detailed results are reported in Table I.

The best outcome was obtained at epoch 47, when the model reached a test accuracy of 70.02% and an F1-score of 71.73%. Training was stopped at this point based on an early stopping criterion, which monitored validation loss to prevent overfitting and improve generalization to unseen samples. These values confirm that the model is able to distinguish between visually similar gesture classes with reasonable accuracy, even under real-time conditions.

While the Jester dataset does not represent full sign language sequences, it served as a practical starting point for building and evaluating the first version of our gesture classification pipeline demonstrating its suitability for integration into the mobile application while maintaining high performance.

This model can be fine-tuned using domain-specific datasets containing continuous and linguistically structured sign language, allowing the system to handle real-world signing more accurately.

Table I: Test accuracy and precision of the 3D-CNN model on Jester at different epochs.

Epochs	***Accuracy(%)***	***Precision(%)***
20	67.49	69.36
30	69.96	71.57
40	69.37	71.20
47	70.02	71.73

B. Voice-to-Video Translation Results

To evaluate the performance of the Voice-to-Video translation component, we replicated the evaluation experiment proposed by the creators of the translation toolkit described in [13]. The focus was placed on the text-to-gloss and gloss-to-text translation stages, using the Public DGS Corpus, a large dataset comprising over 60,000 aligned pairs of glossed German Sign Language (DGS) and corresponding German sentences.

Two model configurations were evaluated: a bilingual model trained on a single translation direction (either DGS→German or German→DGS), and a multilingual model trained on both directions simultaneously. The CHRF metric was used to measure translation quality, as it captures character-level precision and recall, making it especially suitable for gloss-based sequences where small variations can significantly affect meaning.

The results, summarized in Table II, show that the multilingual model achieves a higher CHRF score in the German-to-DGS direction (32.12 vs. 30.39) compared to the bilingual setup. In the reverse direction (DGS→German), the bilingual configuration performed slightly better (27.56 vs. 26.72), although the difference is minimal. These outcomes are aligned with those reported in [13], confirming that the multilingual approach provides better generalization and performance in one direction without a significant trade-off in the other.

Table II: CHRF scores comparing bilingual and multilingual models for DGS↔German translation.

Configuration	***DGS→DE CHRF***	***DE→DGS CHRF***
Bilingual (DGS→DE)	27.56	-
Bilingual (DE→DGS)	-	30.39
Multilingual	26.72	32.12

C. Mobile Application Performance

To evaluate the runtime efficiency of the bidirectional translation mobile application, we measured the average

execution time of the main processing stages during real-time operation on the deployed mobile application. This analysis helps to identify which stages are the most computationally demanding and ensures that the system can meet the timing requirements for continuous interaction. Table III presents the average duration for each processing stage in both translation directions.

Table II: Average execution time for the main processing stages in both translation directions.

Processing Stage	***Video-to-Voice***	***Voice-to-Video***
Trasmission Time (App→Service)	0.028s	0.025s
Preprocessing	0.846s	0.034s
Model Inference	0.336s	0.786s
Trasmission Time (Servise→App)	0.028s	0.025s
Total	1.238s	0.870s

As shown in the table, the most time-consuming stage in the Video-to-Voice direction is preprocessing, which takes on average 0.846 seconds. This is likely due to the use of the MediaPipe framework for extracting landmarks and assembling them into a spatiotemporal tensor of 37 frames. By contrast, the 3D-CNN inference is relatively fast, with an average execution time of 0.336 seconds, allowing gesture classification to proceed in near real-time.

In the Voice-to-Video direction, the computational load is distributed differently. Here, the preprocessing phase is minimal, requiring only 0.034 seconds. The inference stage, which involves generating the full sequence of 3D skeletal poses and rendering them into avatar animation, is the most computationally expensive part, taking 0.786 seconds on average.

In both pipelines, the transmission time between the mobile device and the Python microservices is extremely low and can be considered negligible when compared to the overall processing time. As expected, the model inference remains the most demanding component in both directions.

Considering that the system processes a video window every 1.3 seconds in the Video-to-Voice direction and an audio window approximately every 3 seconds in the Voice-to-Video direction, the observed execution times are within the available time budget. This confirms the feasibility of the system for real-time use in practical mobile communication scenarios.

V. Conclusion and Future Works

In this work, we proposed a real-time, low-cost translation system for bidirectional communication between deaf and hearing individuals. The platform integrates a neural-based translation engine within a mobile application, enabling real-time video calls with automatic interpretation between spoken language and sign language. The system includes both a gesture recognition pipeline based on 3D-CNNs and a sign generation module using text-to-gloss and gloss-to-pose conversion. Experimental results confirmed the feasibility of the approach for both communication directions, demonstrating its responsiveness and usability in real-time scenarios. Future developments will focus on improving platform availability and model performance. From a deployment perspective, the mobile application will be extended to support iOS devices, and new features will be added to enhance usability and cross-platform compatibility. On the translation side, the current gesture recognition model, actually trained on the Jester dataset, will be fine-tuned using sign language-specific datasets to improve accuracy and adapt the system to more complex, continuous signing scenarios. These enhancements aim to make the system more robust, inclusive, and suitable for real-world use.

References

[1] Shezi, Mandlenkosi & Ade-Ibijola, Abejide. (2020). Deaf Chat: A Speech-to-Text Communication Aid for Hearing Deficiency. Advances in Science, Technology and Engineering Systems Journal. 5. 826-833. 10.25046/aj0505100.

[2] N. Azodi and T. Pryor "SignAloud gloves," Multilingual Magazine, University of Washington, 2016. [Online]. Available: https://multilingual.com/signaloud-gloves/

[3] SignAll Inc., "Real-Time Sign Language Translation System," [Online]. Available: https://cordis.europa.eu/article/id/411590-first-system-to-automatically-translate-sign-language

[4] KinTrans Inc., "KinTrans: Real-Time Sign Language to Voice Translation," [Online]. Available:https://dallasinnovates.com/kintrans-sign-language-tech-translates-movements-text-voice/

[5] R. Elliott, J. Mackenzie, and M. Glauert, "Linguistic Modelling and Language-Processing Technologies for Avatar-Based Sign Language Presentation," Universal Access in the Information Society, vol. 9, no. 4, pp. 295–311, 2010.

[6] T.Hanke,"Representing sign language data in language resources and language processing contexts" in Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC), Lisbon, Portugal, 2004.

[7] S. Stoll, N. C. Camgöz, O. Koller, and R. Bowden, "Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018, pp. 10023–10033.

[8] N. C. Camgöz, O. Koller, S. Hadfield, and R. Bowden, "Neural Sign Language Translation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018, pp. 7784–7793.

[9] B. Saunders, N. C. Camgöz, and R. Bowden, "Progressive Transformers for End-to-End Sign Language Production," in Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 2020, pp. 687–705.

[10] S. Yin and J. Read, "Better Sign Language Translation with STMC-Transformer" in Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Seattle, WA, USA, 2020, pp. 3461–3471.

[11] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," arXiv preprint arXiv:1906.08172, 2019.

[12] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2017, pp. 7291–7299.

[13] A. Moryossef et al., "An Open-Source Gloss-Based Baseline for Spoken to Signed Language Translation," arXiv preprint arXiv:2305.17714, 2023.

[14] A. Moryossef et al., "pose-format: Library for Viewing, Augmenting, and Handling .pose Files," arXiv preprint arXiv:2310.09066, 2023.

[15] Moroney, Laurence. (2017). Firebase Cloud Messaging. 10.1007/978-1-4842-2943-9_9.

[16] Mediasoup [Online]. Available: https://mediasoup.org/documentation/