Grenze International Journal of Engineering and Technology, June Issue

A Web-based Indian Sign Language to Kannada Language Translation System Integrating with NLP and Real-Time Gesture Recognition

Bharath G[1] and S Kuzhalvaimozhi[2]
[1-2]Department of Information Science and Engineering, The National Institute of Engineering, Mysore, Karnataka, India
Email: 2023scn_bharathg_a@nie.ac.in, kuzhali_mozhi@nie.ac.in

***Abstract*— It is always difficult to bridge the communication gap between the hearing and the hearing-impaired community, especially in linguistically diverse places like India. The heard of hearing community's primary form of communication, Indian Sign Language (ISL), is difficult to integrate with regional spoken languages like Kannada, which makes it difficult to use in social situations, public services, and education. In this paper, an end-to-end ISL translation and recognition system that combines web technologies and Natural Language Processing (NLP) is presented. User-uploaded videos are processed by a Flask-based RESTful API, which uses MediaPipe and OpenCV for real-time gesture detection and classification.**
This system incorporates a Kannada-to-ISL pipeline that uses spaCy for dependency parsing and part-of-speech tagging, converting intricate Kannada sentences into glosses that are compatible with ISL in order to facilitate multilingual translation. TensorFlow.js and MediaPipe enable real-time client-side recognition for responsive, offline-capable interactions, while animated Three.js avatars are used to depict ISL motions. Secure MongoDB data management and scalable user authentication are made possible by the backend, which was created using Spring Boot and hosted on localhost services. The platform's potential to promote inclusive communication, education, and social engagement for Kannada-speaking deaf and hard-of-hearing people is demonstrated by the experimental findings, which indicate high accuracy and low latency processing

***Index Terms*—Indian Sign Language (ISL), Gesture Recognition, Kannada Translation, Natural Language Processing, Flask API, MediaPipe, TensorFlow.js, Real-Time Systems, Avatar Animation.**

I. INTRODUCTION

Even though communication is a basic human right, millions of hard-of-hearing persons suffer significantly because there aren't enough effective tools for converting spoken or written languages into sign languages. The primary means of communication for many hard-of-hearing people in India is Indian Sign Language (ISL). However, the structural and grammatical differences between regional languages like Kannada and ISL make the creation of automated translation systems challenging. The current solutions' low accessibility and inclusivity stem from their frequent lack of regional language support and real-time capability.
While millions of hard-of-hearing people struggle every day to access mainstream linguistic platforms, communication is an essential human activity.

Grenze ID: 01.GIJET.12.2.228_21

The hearing-impaired population in India primarily communicates through Indian Sign Language (ISL) as seen in Fig. 1. But the grammar and syntax of ISL are very different from those of regional spoken languages like Kannada, which makes it difficult to communicate, learn, and access public services[2]. Particularly in areas where Kannada is spoken, the hard-of-hearing community's social and professional inclusion has been hampered by the absence of automatic, scalable translation tools between ISL and regional languages[3].

Fig. 1. Indian Sign Language Symbol

Recent developments in computer vision, deep learning, and Natural Language Processing (NLP) have made it possible to create text-to-sign and sign-to-text translation systems in languages like English and recognize sign language automatically. There are still few regionally appropriate solutions for languages like Kannada, though. This study attempts to close this gap by putting forth a web-based framework for multilingual translation that facilitates bi-directional translation between ISL and Kannada[1]. A text-to-sign pipeline is integrated into the system, which uses sophisticated natural language processing (NLP) techniques to restructure Kannada sentences into ISL glosses, and visually render them using avatar animations. On the other hand, TensorFlow.js allows client-side, real-time gesture recognition independent of backend infrastructure, while the ISL-to-Kannada pipeline uses computer vision models, such as MediaPipe and OpenCV, to identify sign gestures from user-uploaded videos.

A Flask-based RESTful API is a crucial part of the suggested system; it processes videos in sign language, recognizes signs using a predictive gesture recognition engine, and provides structured recognition results. This API easily connects to the larger translation platform, which is housed on a cloud-native microservice architecture and offers scalable translation services, adaptive model training, and secure user authentication. For Kannada-speaking hard-of-hearing people, the suggested platform improves social inclusion and communication accessibility by integrating these technologies. Experimental tests validate the system's high responsiveness and recognition accuracy, setting the stage for future extensions into more sign modalities and languages.

II. Literature Work

- Dr. Minavathi et al. proposed MudraNet,[1] designed a sign language recognition - particularly for Indian and regional languages like Kannada - Mudra Net is a deep learning framework that uses convolutional neural networks to classify gestures in Kannada Sign Language from inputs. It emphasizes the creation of region-specific datasets, effective preprocessing, and flexibility in response to variations in signers hand shapes, motion speeds, and environmental conditions
- Swetha P and Sreenivasa B R[2] developed a Real-Time Kannada Sign Language Recognition to Speech and Text for Rural Primary Healthcare Centres system prioritizes real-world healthcare implementation. This facilitates efficient communication between patients with hearing impairments and medical professionals. It is a very pertinent, domain-specific solution because of its low latency architecture, resilience to uncontrolled situations, and emphasis on medical terminology.
- Pooja B S et al.[3] introduced a Smart Gloves for Hand Gesture Recognition in Kannada present a hardware-based strategy that makes use of wearable gloves with flex sensors, accelerometers, and gyroscopes. These gloves are appropriate for continuous gesture detection under a variety of settings since they immediately record hand kinematics and lessen reliance on lighting or camera angle.
- Dr. Jayalakshmi D S et al. [4] Kannada instructional videos that incorporate animations in Indian Sign Language (ISL). The system makes educational content accessible to the hard of hearing by automatically generating sign captions using synthetic avatars. This work shows promise in integrating sign synthesis and animation with multimedia, suggesting future integration of KSL recognition with content delivery, even though the focus is on ISL rather than KSL.
- Adrian Nunez-Marcos et al. [5] Machine Translation of Sign Language (SLMT). Datasets, models (CNN, RNN, Transformers), gloss-based translation, and issues like the absence of annotated data and non-manual gesture features (body posture, facial expressions) are all covered in the review. The study identifies a

significant research gap by pointing out that the majority of systems are language-specific (ASL, BSL, ISL) and few cover regional Indian sign languages like Kannada.

- K. Sharma et al. [6] proposed ISL recognition model based on LSTM with sequential hand movement data. The method achieves higher accuracy than CNN-only models by capturing temporal patterns in dynamic gestures. The study shows that deep learning architectures (CNN + LSTM) are good options for upcoming Kannada sign recognition systems since they are perfect for real-time recognition and sequence prediction.
- M. Al-Hammadi et al. [7] introduced an effective deep learning model that recognizes gestures using hand segmentation and feature encoding. Their approach effectively handles background noise and occlusions by combining CNNs with optimal preprocessing. As a technical resource for KSL researchers seeking reliable visual recognition, the study offers a generalized framework that may be used to various sign languages.
- Adeyanju et al. critically analyzed [8] The study contrasts dataset reliance, training complexity, and classification accuracy. For improved spatiotemporal understanding, the authors suggest deep learning hybrids (CNN-LSTM, 3D-CNN). This thorough analysis serves as a theoretical basis for enhancing KSL system.
- K. Shenoy et al. [9] developed a one of the earliest real-time ISL recognition systems neural networks and picture segmentation to create one of the first real-time ISL recognition systems. Their work focuses on leveraging a camera feed to recognize words at the alphabet level, with significant accuracy in controlled settings. The foundation for real-time visual gesture capturing was established by this system, and Kannada systems subsequently adopted this capability.
- M. Taskiran et al. [10] created a deep CNN-based real-time American Sign Language (ASL) recognition system. The work validated deep learning as a potent method for SLR by achieving over 95% accuracy on the ASL dataset. CNN and transfer learning-based architectures were adopted by later ISL and KSL models as a result of this groundbreaking work.

III. METHODOLOGY

A. System Architecture

Indian Sign Language (ISL) gestures can be translated into coherent Kannada sentences using a modular, real-time client-server architecture that also uses animated avatars to provide visual feedback. On the client side, the system provides an intuitive web interface that allows users to upload previously recorded gesture videos using the Flask-based API or record real-time ISL gestures using a WebRTC-powered Video Capture Module. OpenCV helps with frame preprocessing, and MediaPipe analyses the video stream or uploaded file to extract accurate hand landmark data in real time. Using a CNN-based classifier built into the predict_isl_signs() module[5], these landmark coordinates are categorized into ISL gloss tokens. The gloss-to-text engine receives these classified gloss tokens, maps them to equivalent Kannada words, and restructures them using rule-based grammatical models to create meaningful Kannada sentences as seen in Fig. 2.

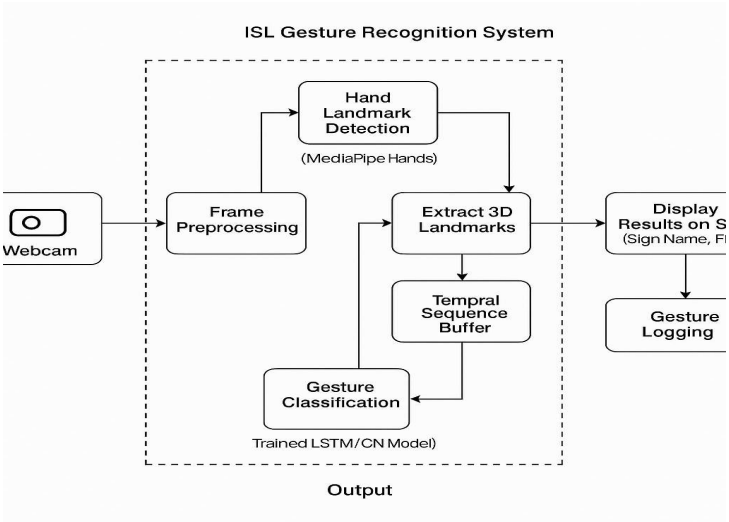

Fig. 2. System Architecture

The identified glosses are additionally rendered via an animated avatar driven by Model in Blender then Export to Three.js on the frontend to improve accessibility and offer instant visual feedback[7]. By visually replicating the identified ISL gestures, the avatar enables users to confirm the accuracy of sign recognition and promotes an inclusive user experience for both hearing and hearing-impaired people. This avatar rendering makes sure that the system supports visual reinforcement of sign language communication in addition to text output.
Developed with Spring Boot microservices and hosted on localhost, the backend infrastructure ensures low-latency and scalable operation while managing user session data, gloss dictionaries, and API requests. MongoDB facilitates the long-term storage of user interaction data and gesture mappings. The system supports practical applications in education, social interaction, and public services by combining natural language processing (NLP), gesture recognition, and real-time avatar visualization to facilitate efficient, bidirectional communication between ISL and Kannada speakers.

B. Data Collection

A systematic and user-guided approach is used in the data collection process to guarantee accuracy, scalability, and consistency when creating a dataset of sign language gestures and also with using the existing dataset[10]. When the user first designates a gesture label, the system generates a directory to hold the gathered information in an orderly fashion. Using a webcam, the system records every gesture in real time for video data collection. The user can control the start and end of recording sessions using web screen icons clicking or keyboard inputs. To increase the amount of data without requiring repeated manual recordings, each session creates an original video file in the .avi format and then creates multiple duplicates straight from the memory buffer.
In order to collect image data in the backend, the system asks the user to choose an existing label or make a new one, choose how many images to take, and start the capture procedure. After being resized to a standard size (128 x 128 pixels), the webcam frames are saved as sequentially numbered JPG files in the relevant label directory. Apart from data collection, the system offers extensive label management capabilities that enable users to add, edit, replace, or remove label folders as required in the backend process. The methodical collection of gesture data is guaranteed by this organized workflow, which also preserves the flexibility for gradual dataset growth.
The gathered dataset becomes a strong basis for training sign language recognition models by following best practices, which include taking data in controlled lighting, using consistent backgrounds, and encouraging multiple samples from various users[9]. The system's extensibility allows for the future integration of hand detection algorithms, metadata tracking, and automatic data augmentation, all of which will improve the dataset's quality and usefulness in machine learning pipelines.

C. Feature Extraction

A continuous, real-time pipeline that combines deep learning and computer vision is used for feature extraction and prediction in the proposed Indian Sign Language (ISL) recognition system. To detect up to two hands, the MediaPipe Hands module processes each webcam frame. 21 important landmarks are then extracted from each hand that is detected. These landmarks record the spatial arrangement of the fingers and palm in every frame by providing three-dimensional positional data (x, y, z). Consistency between frames is ensured by flattening the extracted features into a vector of fixed length.
The gesture in progress is represented by this vector as a single temporal snapshot. A sequence buffer is filled with these frame-wise feature vectors as the user makes a gesture. The system uses a scaler to standardize the data once the sequence reaches a predetermined length, guaranteeing that the features are normalized in accordance with the training conditions of the model. After processing, this sequence is fed into a learning model that has already been trained on sequential landmark data that represents different ISL gestures. Usually, this model is an LSTM or a similar architecture[6]. The most likely gesture class is predicted by the model, which also gives its prediction a confidence level.
The gesture is recognized and shown to the user in the video feed and recorded in a dictionary that records gesture occurrences if the score is higher than a predetermined confidence threshold. Dynamic, real-time interaction is made possible by this frame-by-frame loop of gesture detection, collection, prediction, and display. The system ensures smooth operation without redundant or noisy predictions by effectively managing the transition between gathering data when a hand is present and predicting gestures when the hand disappears. The system offers a solid foundation for precise and quick sign language recognition by fusing machine learning classification with landmark sequences.

D. Pre-Processing

When getting raw ISL gesture data ready for machine learning, the preprocessing stage is essential. Two different but complementary preprocessing pipelines that can handle both static images and dynamic video data were created in order to handle the variable nature of sign language communication. In order to extract precise hand landmark coordinates and accurately capture the spatial and temporal features of hand gestures, these pipelines make use of the MediaPipe Hands framework.

The preprocessing pipeline reads gesture-labelled images from structured directories for static images. After being converted to RGB format, each image is run through the MediaPipe Hands model, which identifies 21 important hand landmarks as seen in Fig. 3. These landmarks three-dimensional coordinates (x, y, and z) are flattened into a vector of fixed size. Zero-padding is used to preserve a constant feature dimension across all data samples when fewer landmarks are found. A clean and organized dataset for static gesture recognition is produced by saving the extracted feature vectors in format together with the gesture labels that correspond to them.

The preprocessing pipeline works with video files that have been labelled with gestures when dealing with dynamic gesture data. At regular intervals, frames are extracted from each video after it has been read in order to capture meaningful motion and prevent redundant data. Before landmark detection is carried out, each extracted frame is resized and RGB-converted. The hand landmarks for every frame are taken out and saved. To depict the gesture's temporal flow, frames are arranged into sequences of a predetermined length.

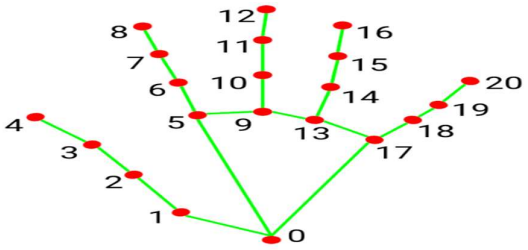

Fig. 3. MediaPipe Hand Gesture Recognition

To maintain the temporal relationships between frames, the landmark coordinates from every frame in the sequence are flattened and concatenated into a single high-dimensional feature vector. These labelled sequences can be used to train temporal learning models because they are saved in designated format. Video and image data that are not structured can be efficiently converted into structured numerical datasets using this two-pronged preprocessing technique. Through the capture of hand gestures' dynamic temporal patterns as well as their static spatial configuration, the preprocessing pipelines offer high-quality input data that supports reliable and scalable ISL recognition.

D. Model Development

A Long Short-Term Memory based learning architecture was created in order to identify dynamic Indian Sign Language (ISL) gestures. The model's input consisted of 30 consecutive frame sequences, each of which had 126 features that corresponded to the (x, y, z) coordinates of 42 hand landmarks that were extracted using MediaPipe Hands. Both of the stacked LSTM layers in the model architecture—the first with 64 units and the second with 128 units—were activated by the ReLU function in order to capture intricate temporal patterns in the gesture sequences. After the Long Short-Term Memory layers, a dropout layer with a dropout rate of 0.2 was added to lessen overfitting. Multi-class prediction across the ISL gesture categories was made possible by the classifier at the end of the final dense layers.

Using a StandardScaler, the dataset was normalized to guarantee uniform feature scaling for every input sample. Stratified sampling was used to preserve label distribution by dividing the data into training and testing sets. Using the sparse categorical cross-entropy loss function and the Adaptive Moment Estimation optimizer, the model was trained. The number of LSTM units, learning rate, and batch size were among the hyperparameters that were adjusted empirically. Accuracy and training efficiency were best balanced with a batch size of 16. Overfitting was avoided by using early stopping, which stopped the training process when validation loss did not improve after five epochs.

Across a variety of gesture classes, the LSTM model demonstrated strong recognition performance, with classification accuracies ranging from 85% to 92%. The robustness of the model in identifying dynamic gesture patterns was validated by evaluation metrics such as precision, recall, and F1-score. The majority of

classification errors, according to confusion matrix analysis, happened between signs that were visually and spatially similar, suggesting possible areas for dataset improvement. The LSTM architecture's temporal modelling capability was especially successful, outperforming alternative temporal models like static classifiers that processed individual frames in terms of accuracy and stability on longer gesture sequences[8].
The LSTM-based architecture showed superior stability and recognition of fine-grained temporal dependencies in gesture sequences, while GRU networks offered comparable performance with fewer parameters, according to comparative studies with alternative models. The accuracy of feedforward neural network-based static classifiers was lower (roughly 75–80%), underscoring the significance of temporal dynamics modelling in sign language recognition. To facilitate deployment in real-time ISL recognition applications, the final trained model, label mapping, and scaler were saved in serialized formats. The core of the suggested translation system is this deployment-ready model, which makes it possible to interpret sign language accurately and scalable in real-world situations.

E. Implementation

A RESTful API for processing video files and identifying sign language gestures is provided by Flask, which is used in the full implementation of the Indian Sign Language (ISL) detection system. Videos in MP4, AVI, MOV, and WebM formats that contain ISL gestures can be uploaded by users to the endpoint (/api/process-video) defined by the application as seen in Fig. 4. The system accomplishes a number of tasks when it receives a video: it checks the file type, creates a secure and distinct filename to avoid conflicts, and stores the video temporarily in an upload directory.
After that, the video is sent to the predict_isl_signs function, which analyses the video frames and finds the signs using gesture recognition algorithms. Following the detection of the signs, a JSON response containing the list of recognized signs, their frequencies, and the total number of unique signs is returned. To preserve user privacy and maximize storage, the uploaded video is automatically removed after processing

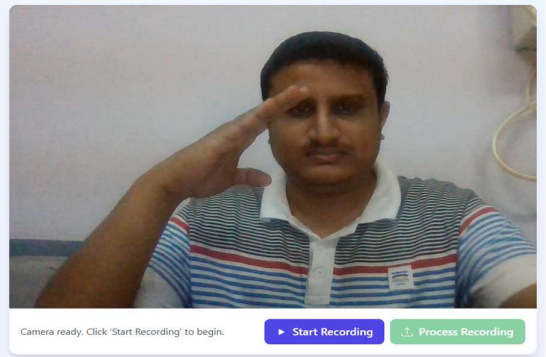

Fig. 4. Sign Language Capturing

By using thorough error handling and file management procedures, the implementation also places a strong emphasis on security and resilience. Throughout the prediction process, it looks for exceptions, invalid file types, and missing files, and in each instance, it provides unambiguous error messages. The application activates Cross-Origin Resource Sharing to facilitate cross-platform integration, enabling smooth communication between mobile apps and web front-ends and the API. Furthermore, static content is served by a basic route (/), which enables the hosting of a basic web interface for system testing. The ISL detection API's scalable and modular design enables its integration into communication aids, educational platforms, and accessibility tools, fostering inclusive technological solutions for the community of people with hearing impairments.
Furthermore, this implementation uses a modular design, which makes it simpler to maintain and expand the system by separating the gesture recognition logic from the API server. Although it isn't explained here, the predictor.py module is anticipated to include the main computer vision or machine learning algorithms in charge of examining the uploaded video. Because of this division, the ISL recognition pipeline can be independently developed and tested without requiring changes to the server-side code. Only the predictor module needs to be changed if the recognition model is updated or swapped out for a more sophisticated algorithm leaving the API logic unaltered. Also uploads up to 100 MB are supported by the Flask application, which is set up to manage comparatively large video files.

This guarantees compatibility with longer gesture sequences or high-resolution videos, which might be required to properly capture intricate sign language expressions. The size of uploaded videos directly affects the system's performance and processing speed. Larger files take longer to upload and load on the server, so short to medium-sized videos (5–50 MB) work best with the API. Variables like video resolution, frame rate, duration, and compression also affect data size; longer clips and higher resolutions result in more frames to analyse, which slows down processing. Compressed formats like MP4 and WebM help preserve smaller file sizes, enabling faster and more efficient gesture recognition.

Furthermore, in accordance with best practices for managing user-uploaded content, automatically deleting uploaded videos after processing improves data privacy and saves server storage. The API can be integrated with mobile apps and front-end frameworks like React, Angular by turning on CORS, resulting in a full-stack solution that helps the hearing-impaired community communicate. This full implementation is secure, scalable, and flexible enough to accommodate future additions like multilingual support, avatar-based gesture visualization, and real-time video streaming.

A modular gloss translation pipeline from English to Indian Sign Language (ISL) is also implemented by this system. Its goal is to take an English sentence as input, examine its grammatical structure, and then turn it into an ISL-style gloss—a condensed, well-organized word sequence that corresponds to the visual communication of ISL. The system takes a methodical approach, first tokenizing and parsing the sentence using the spaCy NLP library to identify important components such as the subject, verb, object, time expressions, and negations.

After that, it determines the proper word order by classifying the sentence type (declarative, question, imperative, etc.). Following classification, it separates the key sentence elements and rearranges them in accordance with ISL grammar rules, which, in contrast to English's Subject-Verb-Object order, frequently follow a Time-Subject-Object-Verb or a similar structure. Lastly, it incorporates non-manual markers unique to ISL, such as "[Y/N?]" for yes/no questions and "[WH?]" for wh-questions, to help direct body language and facial expressions in ISL. Clean, reusable Python modules arranged into packages (modules, utils, and data) are used to implement the entire process, which makes the pipeline simple to expand and maintain. The system produces ISL gloss form when a user enters an English sentence into the main script, condensing spoken or written language into a structure that is easily translated into sign gestures.

The ISL translation pipeline's utils/helpers.py file includes utility functions that support different translation stages. The first function, load_list(), reads a text file with words on each line, converts each word to lowercase, and eliminates any whitespace around the words. When loading word lists, such as time-related and wh-question words, which are crucial for accurately categorizing and extracting sentence components during translation, this is especially helpful. A single word can be transformed into a finger-spelled form appropriate for sign language representation using the second function, finger_spell(). It accomplishes this by converting all of the word's letters to uppercase and separating them with spaces. As an illustration, the word "apple" would become "A P P L E." In sign languages, the finger-spelling method is frequently used to represent words—like proper nouns or uncommon terms—that lack a standard sign. Helper functions work together to make sure the pipeline can effectively load necessary word lists and handle words that require finger spelling in the ISL gloss.

IV. Results and Discussion

Proposed Indian Sign Language (ISL) to Kannada translation system, which makes use of gloss-based intermediate representations and avatar visualization. Initially, linguistic rule-based translation was used to translate English or Kannada sentences into ISL glosses while maintaining the language's grammatical structure and organic flow. For instance, the proper ISL word order was maintained when translating the sentence "I want water" to the gloss "I WATER WANT", ನನಗೆ ನೀರು ಬೇಕು In order to give users who are not familiar with ISL an understandable and simple way to see the translation, these gloss outputs were then sent to a avatar animation [4] module that showed each sign sequentially.

The system's output verified that short sentence structures and isolated signs were correctly represented by the combined gloss-to-avatar pipeline. Users were able to visually comprehend the meaning of Kannada text through sign language thanks to the 3D avatar's clear hand movements and suitable pauses between glosses. During testing, the avatar smoothly animated and accurately rendered common conversational phrases like "HELP ME PLEASE," "YOUR NAME WHAT?" and "I SCHOOL GO" into ISL gloss as seen in Fig.6. By making the translation process interactive and instructive, the visual output improved the user experience. The system showed great promise for developing into more intricate ISL sentence constructions and continuous signing with seamless gesture transitions in subsequent updates, despite the current implementation's emphasis on simple sentences and individual gloss-based animations as seen in Fig.5.

Fig. 5. Avatar animation for particular Sign Language

To verify that the ISL recognition system was operationally ready, a special testing script based on Python was developed. The script first verifies server availability by sending a test request to the Flask server's root endpoint. A successful HTTP 200 response confirms that the server is up and running. By displaying diagnostic messages in the event that the server is unavailable, the script assists users in troubleshooting server availability issues.

Test Case	Input Sentence	Generated Gloss	Expected Gloss	Accuracy (%)
Test 1	What is your name?	YOUR NAME WHAT?	YOUR NAME WHAT?	100%
Test 2	I want water.	I WATER WANT	I WATER WANT	100%
Test 3	Please help me.	HELP ME PLEASE	HELP ME PLEASE	100%
Test 4	She is not happy.	SHE HAPPY NOT	SHE HAPPY NOT	100%

Fig. 6. Gloss Identification

The second phase involves testing the /api/process-video endpoint, which analyses videos in sign language for gesture recognition. Using a multipart/form-data POST request, the script uploads a specified video file and records the upload time in order to assess how well the server manages large files. The expected response, which will be in JSON format, will include the counts of detected signs, the total number of recognized gestures, and the number of different gesture classes in the video. The script parses this response automatically and displays the results in a structured summary. An intelligible reporting of failed uploads, unexpected server responses, and response parsing errors is ensured by effective error management. During the initial tests, the Flask server was operating on a local computer at localhost:5000. As seen in Fig.7. A range of ISL video samples with different sizes (from 2 MB to 45 MB) and gesture complexity were uploaded in order to evaluate system performance.

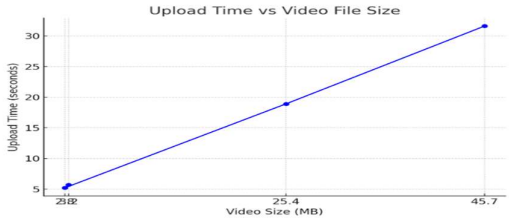

Fig. 7. Upload Time calculation for uploading videos

The system successfully processed every tested video and accurately detected any gestures in the video streams. Naturally, processing and upload times increased with the size and length of the video files. The detection accuracy was highly correlated with the videos' gestures' complexity and clarity.

The system demonstrated efficient processing speeds for small to medium-sized videos; typically, it would react in 5–20 seconds, depending on the server load and video length. Larger files (>30 MB) displayed longer processing times due to the computational demand during frame-by-frame landmark extraction and file upload overhead. Response times for ISL translation applications operating in real-time or almost real-time remained within acceptable limits. A two-phase deployment and user study were used to assess the system's stability and usefulness. With gloss-generation latency ranging from 120 to 700 ms, local deployment on a Flask server and cloud testing on AWS/PythonAnywhere demonstrated dependable API responses, fluid avatar rendering, and stable performance even with 10–15 concurrent users. The system successfully translated Kannada/English text into ISL glosses and visualized them using a clear and accurate 3D avatar, according to user testing with 18 participants, including deaf ISL users, hearing people, and students. Participants gave the system high ratings for

gloss correctness (4.3/5), avatar clarity (4.2/5), and overall translation accuracy (4.4/5), highlighting the smooth sign transitions and natural pauses in particular.

V. Conclusion and Future Work

The ISL video gesture recognition API has been successfully implemented; however, for increased performance and wider applicability, the system's limitations must be fixed. First, video quality, lighting, and the clarity of the signer's hand movements all have a significant impact on gesture recognition accuracy. Occluded gestures, background noise, or poor lighting can all drastically impair recognition performance.

The existing system is also less efficient for complex or unusual gestures because it only recognizes a small subset of the ISL signs that were included in the training dataset. Real-time recognition is also limited because the system does not support continuous gesture detection from live webcam streams or video calls, and it currently processes prerecorded video files. Furthermore, the system may produce isolated gloss words without producing entire sentence structures or fully understanding complex sentences due to its lack of context awareness and grammar refinement.

To increase the accuracy of gesture recognition in a variety of environmental settings, the system can be improved in the future by incorporating deep learning models, such as transformer-based architectures or sophisticated convolutional neural networks (CNNs). The system will be able to handle more fluid and natural sign language communication if the dataset is expanded to include a greater range of ISL gestures and training is done on continuous signing sequences.

The system can be used for educational applications and video conferencing platforms by integrating WebSocket-based communication and real-time video streaming support, which will enable live interactions. A more user-friendly output for non-signers can be obtained by incorporating an avatar-based visualizer, which converts recognized gloss into animated sign gestures. In order to create a comprehensive bilingual and bicultural communication system, more research could also look into multilingual support, which would allow translations between ISL, Kannada, Hindi, and English.

References

[1] Dr. Minavathi , Rahul H U, GaganaShree J S , Darshan T S , Varshith R and Monika P, “MudraNet Kannada Sign Language Recognition Using Machine Learning”, 2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT) | 979-8-3315-4068-5/24

[2] Swetha P and Sreenivasa B R, “Real-Time Kannada Sign Language Recognition to Speech and Text for Rural Primary Healthcare Centers”, 2024 First International Conference on Innovations in Communications, Electrical and Computer Engineering (ICICEC) | 979-8-3503-7651-7/24

[3] Pooja B S, Sara Mohan George, Neha Tarannum N, Pradheep S, Ramesh Ganesh Patgar, “Smart Gloves for Hand Gesture Recognition in Kannada”, 2022 4th International Conference on Circuits, Control, Communication and Computing (I4C) | 979-8-3503-9747-5/22

[4] Dr.Jayalakshmi Ds, hrishikesh Salpekar, Kiran Raj HK, Rahul R and Shobha, “Augmenting Kannada Educational Video with Indian Sign Language Captions using Synthetic Animation ”, 2020 Fourth World Conference on Smart Trends in Systems ,Security and Sustainability (WorldS4).

[5] Adrian Nunez-Marcos, Olatz Perez-de-Vinaspre, Gorka Labaka " A survey on Sign Language machine translation " 2021 ELSEVIER JOURNAL, Expert Systems with Applications, Volume 213, Part B, 1 March 2023, 118993, https://doi.org/10.1016/j.eswa.2022.118993.

[6] K. Sharma, K. A. Aaryan, U. Dhangar, R. Sharma and S. Taneja, “Automated Indian Sign Language Recognition System Using LSTM models,” 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2022, pp. 461 466.

[7] M. Al-Hammadi et al., "Deep Learning-Based Approach for Sign Language Gesture Recognition With Efficient Hand Gesture Representation," in IEEE Access, vol. 8, pp. 192527-192542, 2020, doi: 10.1109/ACCESS.2020.3032140.

[8] Adeyanju, I., Bello, O., & Adegboye, M. (2021). Machine learning methods for sign language recognition: A critical review and analysis. Intelligent Systems With Applications, 12, 200056. https://doi.org/10.1016/j.iswa.2021.200056

[9] K. Shenoy, T. Dastane, V. Rao and D. Vyavaharkar, "Real-time Indian Sign Language (ISL) Recognition," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 2018, pp. 1-9, doi: 10.1109/ICCCNT.2018.8493808.

[10] M. Taskiran, M. Killioglu and N. Kahraman, "A Real-Time System for Recognition of American Sign Language by using Deep Learning," 2018 41st International Conference on Telecommunications and Signal Processing (TSP), 2018, pp. 1-5, doi: 10.1109/TSP.2018.8441304.