Real-Time Bidirectional Indian Sign Language Translation Using Deep Learning and Hand Gesture Recognition

Mrs. Anjana H.S.[1], Charan H.R[2], Dhanush G.G[3], Hemanth Kumar.K.P[4], Mahanthesh Kumar C.A[5]

[1]Assistant Professor, Department of Information Science and Engineering, East West Institute of Technology, Bengaluru, Karnataka, India, anjanahs@ewit.edu.in

[2]UG Scholar, Department of Information Science and Engineering, East West Institute of Technology, Bengaluru, Karnataka, India, hcharan656@gmail.com

[3]UG Scholar, Department of Information Science and Engineering, East West Institute of Technology, Bengaluru, Karnataka, India, ddhanushgg@gmail.com

[4]UG Scholar, Department of Information Science and Engineering, East West Institute of Technology, Bengaluru, Karnataka, India, hemanthkumar7p@gmail.com

[5]UG Scholar, Department of Information Science and Engineering, East West Institute of Technology, Bengaluru, Karnataka, India, mahantheshkumar354@gmail.com

Abstract—Communication between users of sign language and the wider hearing population remains constrained by the limited availability of systems that support two-way, real-time interaction. Indian Sign Language (ISL) is used by a large population of deaf and hard-of-hearing individuals in India, yet many existing computer-vision-based ISL systems address only isolated gesture recognition and do not provide a corresponding pathway for a non-signing user to respond in ISL. This paper presents a web-based, real-time, bidirectional ISL translation system that integrates three communication modes: Sign-to-Text, Text-to-Sign, and Speech-to-Sign. The Sign-to-Text module captures webcam video, applies the MediaPipe Hand Landmarker framework to extract twenty-one landmarks per detected hand (up to two hands), normalizes these landmarks relative to the wrist and hand scale, and combines them into a 126-dimensional feature vector that is classified using a TensorFlow/Keras deep learning model, with predictions communi-cated to the client in real time over a WebSocket connection. The Text-to-Sign and Speech-to-Sign modules map typed or spoken input, respectively, to a library of ISL sign videos, with spoken input first transcribed to text through speech recognition. The system is implemented with a React/Vite/TypeScript frontend, a FastAPI backend with SQLAlchemy/SQLite persistence, and JWT-based authentication, and additionally provides translation history and a usage dashboard. The classifier targets 110 ISL sign classes, of which 108 have valid training data in the current landmark dataset, and the platform separately maintains a library of 151 sign videos for text- and speech-based sign generation. This paper details the system architecture, dataset and preprocessing pipeline, model design, and evaluation methodology, while explicitly distinguishing implemented functionality from directions identified for future work. The trained model attains a validation accuracy of 92.42% on a held-out split of the same dataset distribution used for training; this figure is reported strictly as validation performance and is not equivalent to, and should not be interpreted as, independent test-set accuracy or real-world webcam recognition accuracy, since no independent test dataset has yet been constructed for this project.

Keywords—Indian Sign Language, Sign Language Recognition, Deep Learning, Hand Gesture Recognition, Computer Vision, MediaPipe, TensorFlow, Real-Time Translation, Accessibility

I. INTRODUCTION

Communication underlies nearly every aspect of human social, educational, and professional life. For deaf and hard-of-hearing individuals in India, Indian Sign Language (ISL) serves as a primary means of expression, conveyed through hand shapes, hand movement, and non-manual markers such as facial expression. ISL is a distinct visual-gestural language with its own grammar and is not mutually intelligible with English, Hindi, or other spoken Indian languages. Because fluency in ISL is uncommon among the hearing population, a substantial communication gap persists between signers and non-signers, limiting access to education, healthcare, employment, and everyday interaction for ISL users.

Sign languages, more broadly, function as complete natural languages, and the accessibility challenges faced by their users are increasingly being addressed through technology. Over the past decade, advances in computer vision and deep learning have made automated sign-language recognition substantially more tractable. Approaches have progressed from hand-crafted feature descriptors combined with classical classifiers to convolutional neural networks (CNNs) and recurrent architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. The introduction of landmark-extraction frameworks such as MediaPipe has further simplified this pipeline: rather than processing raw video frames directly, a system can operate on a compact, structured representation of hand or body pose, which reduces input dimensionality and can simplify downstream classification.

Despite this progress, many existing ISL recognition systems are designed around isolated, one-directional gesture recognition: they convert a signed gesture into text or speech but provide no corresponding mechanism for a hearing, non-signing individual to compose a message and have it rendered back into ISL. Such systems therefore support only one half of a genuine two-way conversation. There is also a recognized need for such systems to operate in real time, since a translation pipeline with substantial latency is of limited practical use in a live conversation.

Motivated by this gap, the present project develops a web-based, bidirectional ISL translation system intended to operate in real time and to support communication in both directions. The system integrates three complementary modes of interaction: Sign-to-Text, in which a webcam captures a user's signing and the system predicts the corresponding text; Text-to-Sign, in which typed text is mapped to a sequence of ISL sign videos; and Speech-to-Sign, in which spoken input is first converted to text through speech recognition and then mapped to ISL sign videos in the same manner as Text-to-Sign. By combining these three modes within a single platform, the system is intended to support two-way communication between a signer and a non-

signer rather than recognition in isolation.

The objectives of this project are as follows: (1) to design and implement a real-time Sign-to-Text recognition pipeline using MediaPipe Hands for landmark extraction and a TensorFlow/Keras model for classification; (2) to implement a Text-to-Sign module that maps user-entered text to a sequence of available ISL sign videos; (3) to implement a Speech-to-Sign module that extends the Text-to-Sign pathway to spoken input; (4) to support low-latency, real-time client-server communication for the recognition pipeline using WebSockets; and (5) to integrate these modes into a single authenticated web application with translation history and a usage dashboard.

The main contributions of this work are as follows:

- A real-time Sign-to-Text recognition pipeline that combines MediaPipe Hands landmark extraction with a TensorFlow/Keras classification model, with predictions delivered to the client over a WebSocket connection.
- A Text-to-Sign module that converts typed text into a sequence of ISL sign videos drawn from a library of available sign resources.
- A Speech-to-Sign module that extends text-based sign generation to spoken input via speech recognition.
- Integration of MediaPipe-based hand landmark extraction with a deep learning classifier for the Sign-to-Text pathway.
- Integration of user authentication, translation history, and a usage dashboard around the core translation functionality, implemented with a FastAPI backend, SQLAlchemy/SQLite persistence layer, and a React/Vite/TypeScript frontend.

These contributions are not presented as unprecedented; related systems combining MediaPipe, recurrent or convolutional models, and speech- or text-based sign mapping have been reported in the literature reviewed in Section II. The contribution of the present work lies in the integration of real-time Sign-to-Text recognition, Text-to-Sign generation, and Speech-to-Sign generation within a single authenticated web application, together with translation history and usage monitoring. The remainder of this paper is organized as follows. Section II reviews related work in ISL recognition and translation. Section III describes the proposed system. Section IV discusses the dataset and preprocessing pipeline. Section V describes the deep learning model used for Sign-to-Text classification. Section VI details the system implementation. Section VII presents the experimental methodology. Section VIII discusses results. Section IX outlines limitations, and Section X concludes with directions for future work.

II. RELATED WORK

This section reviews prior work relevant to Indian Sign Language recognition and translation, covering classical feature-based methods, CNN- and LSTM/GRU-based deep learning approaches, MediaPipe-based landmark pipelines, and sentence-level translation.

Katoch *et al.* [1] proposed a static ISL recognition system for the alphabets (A–Z) and digits (0–9) that combines a Bag-of-Visual-Words representation built from Speeded-Up Robust Features (SURF) with a Support Vector Machine (SVM) and a CNN classifier operating on a live video stream, rendering recognized labels as both text and speech. The authors trained their models on a custom dataset of 28,800 images and evaluated on a held-out test set of 7,236 images. A limitation of this approach is its dependence on a hand-crafted feature extraction stage, which can be less robust to background and lighting variation than end-to-end learned features, and its scope is restricted to static alphabet and digit signs.

Sharma and Anand [2] conducted a comprehensive evaluation of multiple deep learning architectures and optimizers for static ISL recognition, comparing models trained from scratch against fine-tuned pre-trained networks such as ResNet and Inception V3 on a custom image dataset varying illumination, viewing angle, and background. Their study found that a CNN trained from scratch outperformed the pre-trained models evaluated, with ResNet performing best among the pre-trained options and Inception V3 performing worst. A limitation noted by the authors is that optimizer and hyperparameter choice had a substantial effect on performance, indicating that model selection for ISL recognition is not straightforward.

Subramanian *et al.* [3] proposed an integrated MediaPipe-optimized Gated Recurrent Unit (MOPGRU) model for ISL recognition, in which MediaPipe Holistic extracts pose, hand, and face landmarks that are processed by a modified GRU cell combining the update and reset gates and replacing standard activations with exponential-linear-unit and softsign variants. This work is directly relevant to the present project as it demonstrates the effectiveness of pairing MediaPipe-based landmark extraction with a recurrent architecture for gesture classification. A limitation is that the model targets isolated sign or word-level recognition rather than continuous, sentence-level translation.

Nehra *et al.* [4] presented an ISL recognition and translation system combining MediaPipe hand-landmark extraction with an LSTM network for gesture classification, and additionally supporting spoken English or Hindi audio captured through a microphone, recognized via a speech-to-text API, and mapped to corresponding ISL sign videos. This system closely parallels the present platform in combining MediaPipe-based recognition with a speech-to-sign pathway; a limitation is that it was demonstrated on a bounded gesture/word set rather than as a fully integrated, authenticated, multi-user web application.

Aparna and Geetha [5] proposed a sign recognition pipeline using a pre-trained CNN for spatial feature extraction followed by a stacked LSTM architecture to capture spatio-temporal information from ISL video sequences, with an additional LSTM layer stacked to improve recognition accuracy. The authors note that, at the time, deep learning approaches capturing temporal structure were comparatively underexplored for ISL relative to static, frame-based recognition. A limitation of the stacked CNN-LSTM design is the added computational cost introduced by cascading two learned temporal layers, which can complicate real-time deployment on resource-constrained hardware. (This paper is included in place of a previously circulated but bibliographically unverifiable CNN-LSTM reference; see the note at the end of this section.)

Navendu and Sahula [6] proposed a word-level ISL recognition approach combining the MediaPipe framework with a hybrid LSTM-GRU network intended to capture temporal dependencies in gesture sequences more effectively than either recurrent unit alone. The system was evaluated on the INCLUDE ISL

video dataset and reported a test accuracy of 89.50%, described by the authors as a modest but definite improvement over prior comparison models. A limitation identified by the authors is that gains over existing individual LSTM- or GRU-based models, while consistent, were incremental, suggesting that word-level ISL recognition on diverse vocabularies remains challenging.

Rawat *et al.* [7] proposed a sequential LSTM model combined with MediaPipe Holistic for recognition of both dynamic and static ISL hand gestures, extracting face, hand, and body landmarks across video frames and classifying the resulting temporal sequences with a two-layer LSTM network using categorical cross-entropy loss and early stopping. A limitation of this study is that the recognition dataset was collected under a constrained set of recording conditions (a fixed frame count and resolution per gesture sample), which may limit generalization to more varied real-world conditions.

Damdoo and Kumar [8] conducted an integrative survey of ISL recognition and translation literature published between 2011 and 2024, covering machine-learning-based, deep-learning-based, and gesture-based recognition techniques as well as the development of ISL datasets and dictionaries. The survey highlights a persistent scarcity of large, standardized, sentence-level ISL datasets suitable for continuous sign-language translation, a gap directly relevant to systems, including the present one, that currently operate at the isolated-sign or word level rather than the full sentence level.

Damdoo, Kumar, and Gogoi [9] introduced ISH-NEWS, described as the first ISL dataset constructed for sentence-level Continuous Sign Language Translation (CSLT), comprising 4,222 sentence videos covering more than 6,500 words, and evaluated a transformer-based translation model against this dataset, reporting that data augmentation improved the model's BLEU-4 score by 8.46 points over the non-augmented baseline. This continuous, sentence-level translation task is discussed here as related and future-facing work; it is distinct from, and not part of, the isolated-sign Sign-to-Text recognition implemented in the present system (see Section IX and Section X).

Note on reference verification: a candidate reference attributed to "S. Bhamare and S. Bhamare" could not be independently verified against a reliable scholarly source at the time of writing and has therefore been omitted in favor of the verified CNN-LSTM reference by Aparna and Geetha [5], in accordance with this paper's reference-verification policy.

Table 1 summarizes these works alongside the proposed system. The literature shows a clear progression from hand-crafted feature descriptors toward MediaPipe-based landmark extraction paired with recurrent architectures, together with an emerging, still-developing interest in continuous, sentence-level translation. Few works, however, integrate isolated-sign recognition, text-to-sign generation, and speech-to-sign generation within a single deployed, authenticated, real-time web application, which is the systems-integration gap the present project addresses.

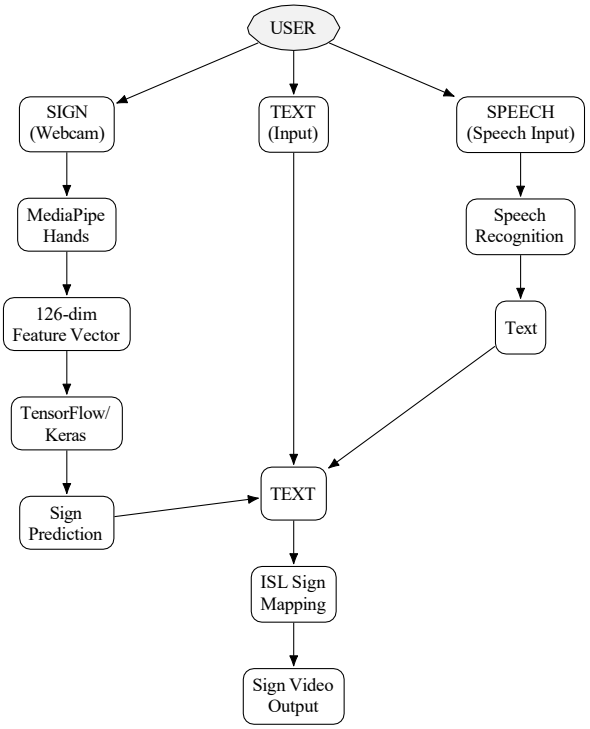


Figure 1: Overall system architecture. Sign, text, and speech inputs are each processed through their respective pipelines and converge on a shared textual representation, from which Sign-to-Text output is displayed directly and Text-to-Sign/Speech-to-Sign output is rendered as ISL sign video.

III. PROPOSED SYSTEM

A. System Overview

The proposed system is a web-based, real-time, bidirectional ISL translation platform composed of a React/Vite/TypeScript frontend, a FastAPI backend, a TensorFlow/Keras recognition model, a library of ISL sign videos, and a SQLite database accessed through SQLAlchemy. The platform exposes three translation modes – Sign-to-Text, Text-to-Sign, and Speech-to-Sign – behind a common, authenticated interface, alongside translation history and a usage dashboard.

Figure 1 illustrates the overall system architecture described above.

B. Sign-to-Text Translation

The Sign-to-Text module captures live webcam frames on the client and processes each frame using the MediaPipe Hand Landmarker, which detects up to two hands and extracts 21 landmark points (x, y, z) per detected hand. Each hand's landmarks are translated relative to the wrist landmark (index 0) and scale-normalized using the distance between the wrist and the middle-finger MCP landmark (index 9), with a small epsilon of 1e-6 added to prevent division by zero. The normalized landmarks for the left and right hand are each stored in a separate 63-value slot (21 landmarks × 3 coordinates); a slot corresponding to an undetected hand is zero-filled. The two 63-value slots are concatenated into a single 126-dimensional float32 feature vector, which is streamed to the backend over a WebSocket connection for classification by the trained TensorFlow/Keras model. The predicted sign is returned to the client and displayed as text, with optional text-to-speech output where

Table 1: Comparison of Related Indian Sign Language Research

Ref.	Year	Technique	Task	Dataset/Data	Reported Result	Scope / Limitation
[1]	2022	SURF + BOVW with SVM and CNN	Static alphabet/digit recognition	Custom dataset: 28,800 train / 7,236 test images	Not reported here (see source)	Hand-crafted features; static signs only
[2]	2021	CNN (from scratch) vs. pre-trained ResNet/Inception V3	Static ISL recognition	Custom illumination/angle-varied image dataset	CNN-from-scratch outperformed pre-trained models	Sensitive to optimizer/hyperparameter choice
[3]	2022	MediaPipe Holistic + MOPGRU (modified GRU)	Isolated sign recognition	MediaPipe-derived landmark sequences	Not reported here (see source)	Not designed for continuous/sentence translation
[4]	2023	MediaPipe + LSTM; speech-to-text for audio input	Sign-to-text and speech-to-sign	Custom gesture/word set	Not reported here (see source)	Limited vocabulary; not a full multi-user web app
[5]	2020	Pre-trained CNN + stacked LSTM	Spatio-temporal ISL recognition	ISL video dataset	Not reported here (see source)	Added compute cost from stacked recurrent layers
[6]	2024	MediaPipe + hybrid LSTM-GRU	Word-level ISL recognition	INCLUDE dataset	89.50% test accuracy	Incremental gain over prior recurrent models
[7]	2025	MediaPipe Holistic + sequential LSTM	Dynamic/static gesture recognition	Custom webcam dataset (fixed frame count/resolution)	Not reported here (see source)	Constrained recording conditions
[8]	2025	Survey (ML/DL/gesture-based methods)	Recognition and translation review	Literature 2011–2024	N/A (survey)	Identifies scarcity of sentence-level ISL datasets
[9]	2026	Transformer-based CSLT	Sentence-level ISL translation	ISH-NEWS: 4,222 videos, 6.5K+ words	+8.46 BLEU-4 with augmentation	Continuous translation quality remains low
Proposed System	Current	MediaPipe + TensorFlow/Keras + Web Application	Sign-to-Text, Text-to-Sign, Speech-to-Sign	132,906 landmark samples (108/110 classes) + 151 sign videos	92.42% validation accuracy (no independent test set)	Class imbalance; no independent test evaluation

implemented. This recognition pathway targets isolated ISL signs; it does not currently perform continuous, sentence-level recognition.

C. Text-to-Sign Translation

In the Text-to-Sign module, a user enters free-form text, which the system processes and maps to a sequence of corresponding ISL sign videos drawn from the 151-video sign library. The matched sign videos are played back sequentially to represent the requested text. Coverage is limited to words and phrases for which a corresponding sign resource exists in the current library; the system does not claim to represent every English word or the complete ISL vocabulary.

D. Speech-to-Sign Translation

The Speech-to-Sign module extends the Text-to-Sign pathway to spoken input: speech is first converted to text using a speech recognition component, after which the resulting text is mapped to ISL sign videos through the same pipeline used by the Text-to-Sign module. As with any speech recognition component, transcription is not guaranteed to be perfectly accurate, and downstream sign mapping quality depends on the accuracy of this transcription step.

E. Authentication and User Management

The platform implements user registration, login, and password recovery/reset functionality, with protected application routes accessible only to authenticated users. Authentication is implemented using JSON Web Tokens (JWT), and passwords are hashed using the pwdlib library prior to storage.

F. Translation History

Completed translations are stored and associated with the authenticated user who performed them, allowing a user to review past Sign-to-Text, Text-to-Sign, and Speech-to-Sign interactions through a dedicated history view.

G. Dashboard

A dashboard view presents translation-related statistics to the authenticated user, summarizing platform usage over time.

H. Real-Time Communication

Real-time interaction for the Sign-to-Text recognition pipeline is implemented using WebSocket communication between the React/Vite frontend and the FastAPI backend, allowing landmark data to be streamed continuously and predictions returned with lower round-trip latency than a conventional request/response HTTP pattern would provide.

IV. DATASET AND PREPROCESSING

The source dataset used to train the Sign-to-Text classifier contains 195,721 images organized into 110 target ISL sign classes. Each image is processed with the MediaPipe Hand Landmarker to extract hand landmarks; after this extraction step and subsequent validation, 132,906 valid landmark samples were retained. This retained landmark dataset represents 108 of the 110 target classes: two classes, "Stop" and "ok fine," produced no valid landmark samples during dataset construction and are therefore not represented in the trained classifier's effective training data, despite the output layer retaining 110 units to match the full target vocabulary (see Section IX). The exact number of source images that failed or were skipped during landmark extraction was not preserved by the dataset-building pipeline and is therefore not reported.

The project separately maintains a library of 151 ISL sign videos, used by the Text-to-Sign and Speech-to-Sign modules to render recognized or entered text; these videos are a distinct resource from the classifier's landmark training data and are not used as classifier training samples.

For each image, MediaPipe's Hand Landmarker detects up to two hands and extracts 21 landmarks per detected hand, each with (x, y, z) coordinates. Preprocessing proceeds as follows: (1) the input image is converted from BGR to RGB using OpenCV; (2) up to two hands are detected, each yielding 21 landmarks; (3) each hand's landmarks are translated so that the wrist landmark (index 0) becomes the coordinate origin; (4) the translated landmarks are scale-normalized by the distance between the wrist and the middle-finger MCP landmark (index 9), with an epsilon of 1e-6 added to the denominator to avoid division by zero; (5) the resulting 63 values (21 landmarks × 3 coordinates) for the left hand and the 63 values for the right hand are stored in two separate feature slots, with a slot corresponding to an undetected hand zero-filled; and (6) the two slots are concatenated into a single 126-dimensional float32 feature vector, which is the classifier's input representation. Hand detection uses MediaPipe confidence thresholds of 0.5 for minimum hand detection confidence, minimum hand presence confidence, and minimum tracking confidence. This landmark-based preprocessing does not involve image resizing, grayscale conversion, or histogram equalization.

The 132,906-sample landmark dataset is split into training and validation subsets using an 80/20 split (validation fraction of 0.2) via scikit-learn's `train_test_split` with a fixed random seed of 42, yielding 106,324 training samples and 26,582 validation samples. The current implementation does not use a separate, independent held-out test set distinct from this validation split (see Section IX).

The retained landmark dataset also exhibits substantial class imbalance. In particular, the classes "Theif" and "nothing" have only 1 valid landmark sample each; "I_m Good" and "There is Gun" have 2 valid samples each; and "Bridge" and "Victory" have 4 valid samples each. The original source-image folders for "Theif" and "nothing" contain substantially more images than this; the low counts reflect landmark-extraction and validation attrition rather than a lack of source images, and this uneven class representation is a limitation of the current dataset discussed further in Section IX.

Table 2: Dataset Summary

Parameter	Value
Source images	195,721
Target sign classes	110 (108 with valid landmark samples)
Number of sign videos	151 (separate resource)
Valid landmark samples	132,906
Train/validation split	80% / 20% (random seed 42)
Training samples	106,324
Validation samples	26,582
Independent test samples	None (no independent test dataset used in the current implementation)
Feature representation	126-dim vector (2 hands × 21 landmarks × x,y,z)
Additional preprocessing	Wrist-relative translation; scale normalization by wrist-to-middle-MCP distance (ε = 1e-6); no resizing, grayscale conversion, or histogram equalization

V. DEEP LEARNING MODEL

The Sign-to-Text classification model is implemented in TensorFlow/Keras and operates on the 126-dimensional normalized landmark feature vector described in Section IV, rather than on raw RGB image data. Representing a hand gesture through a small, fixed-size feature vector reduces the dimensionality of the model input relative to an equivalent image-based pipeline, which is a design rationale for this choice rather than a measured result, since no controlled comparison against an image-based baseline has been performed for this system. The model is a sequential feedforward neural network consisting of two hidden Dense layers (256 and 128 units, both with ReLU activation), each followed by batch normalization and a dropout layer with a rate of 0.3, and a final Dense output layer of 110 units with softmax activation corresponding to the 110 target sign classes. The model is compiled with the Adam optimizer at an initial learning rate of 0.001, using sparse categorical cross-entropy as the loss function, and is trained with a batch size of 32 for up to a configured maximum of 150 epochs with a fixed random seed of 42; training ran for the full 150 epochs. Three callbacks are used during training: `EarlyStopping`, monitoring validation accuracy with a patience of 15 epochs and `restore_best_weights` enabled; `ReduceLROnPlateau`, monitoring validation loss with a decay factor of 0.5, a patience of 5 epochs, and a minimum learning rate of 1e-6; and `ModelCheckpoint`, monitoring validation accuracy with `save_best_only` enabled.

Table 4 gives the exact per-layer parameter breakdown for the trained model.

It is noted that the Keras model summary additionally reports 160,734 optimizer-state parameters associated with the Adam optimizer's internal moment estimates, for a combined displayed total of 241,868 parameters; this optimizer state is not part of the model's learned weights and is excluded from the 81,134-parameter count above, which corresponds to the value

Table 3: Model Configuration

Parameter	Value
Architecture	Dense(256, ReLU) → BatchNorm → Dropout(0.3) → Dense(128, ReLU) → BatchNorm → Dropout(0.3) → Dense(110, Softmax)
Input representation	126-dim feature vector
Classes (output units)	110 (108 with valid training data)
Optimizer	Adam
Loss function	Sparse categorical cross-entropy
Learning rate	0.001 initial, with ReduceLROnPlateau (factor 0.5, patience 5, min 1e-6)
Batch size	32
Random seed	42
Max epochs (configured)	150 (actual epochs run: 150)
Early stopping	Monitors val_accuracy; patience 15; restores best weights
Model checkpointing	Monitors val_accuracy; saves best model only
Trainable parameters	80,366
Non-trainable parameters	768 (batch-normalization statistics)
Total model parameters	81,134
Activation functions	ReLU (hidden layers), Softmax (output layer)

Table 4: Per-Layer Parameter Breakdown

Layer	Parameters
Dense(256)	32,512
BatchNormalization (256)	1,024
Dropout(0.3)	0
Dense(128)	32,896
BatchNormalization (128)	512
Dropout(0.3)	0
Dense(110, Softmax)	14,190
Total model parameters	**81,134**
Optimizer (Adam) state parameters	160,734
Combined Keras-reported total	**241,868**

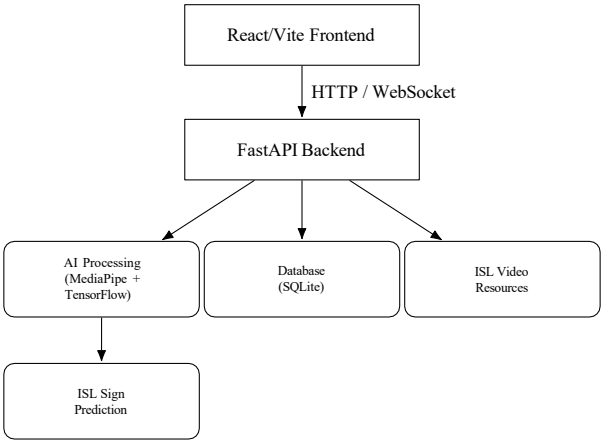


Figure 2: Web application architecture showing the frontend, backend, and the AI processing, database, and video-resource components it accesses.

returned by model.count_params().

VI. SYSTEM IMPLEMENTATION

The frontend is implemented using React with the Vite build tool and TypeScript, styled with Tailwind CSS, and uses Framer Motion for interface animation, React Router for client-side routing, and React Icons for iconography. The backend is implemented in Python using the FastAPI framework, with SQLAlchemy as the object-relational mapping layer over a SQLite database. User authentication uses JWT tokens with pwdlib-based password hashing. Computer vision processing relies on MediaPipe Hands together with OpenCV for frame handling, and the Sign-to-Text classification model is implemented in TensorFlow/Keras. The ISL sign video resources used by the Text-to-Sign and Speech-to-Sign modules are served by the backend and stored alongside the application's model files and database.

The frontend communicates with the backend over standard HTTP requests for authentication, translation history, dashboard data, and Text-to-Sign/Speech-to-Sign requests, and over a WebSocket connection for the real-time, frame-by-frame Sign-to-Text pipeline described in Section III-B. Completed translations are written to the database and associated with the authenticated user's account, which is what enables the translation history feature described in Section III-F.

Figure 2 illustrates this layered web application architecture.

Figure 3 details the Sign-to-Text pipeline referenced in Section III-B.

Figure 4 depicts this bidirectional workflow at a conceptual, communication-flow level, complementing the technical, module-level view given in Figure 1.

VII. EXPERIMENTAL METHODOLOGY

The Sign-to-Text model is intended to be evaluated on an inde-pendent test set disjoint from the data used for training and val-idation. The following standard classification metrics are used to characterize model performance:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \quad (1)$$

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

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively, computed per class and aggregated as appropriate for a multi-class classification setting.

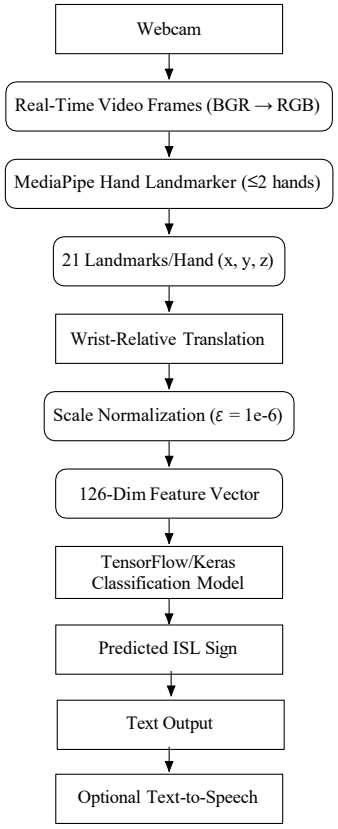


Figure 3: Sign-to-Text processing pipeline, from webcam capture through MediaPipe-based landmark extraction, wrist-relative translation and scale normalization, 126-dimensional feature-vector construction, classification, and text/speech output.

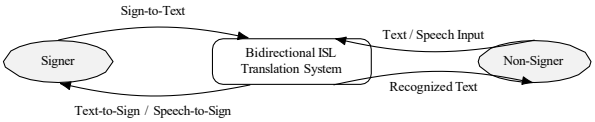


Figure 4: Bidirectional translation workflow. The system converts a signer's gestures to text for a non-signer (Sign-to-Text) and converts a non-signer's typed or spoken input into ISL sign video for a signer (Text-to-Sign, Speech-to-Sign).

In addition to these classification metrics, a full confusion matrix would be used to examine class-wise misclassification patterns, and system-level performance would be characterized using average inference latency (the time from receipt of a frame's landmark data to production of a prediction) and recognition frames-per-second (FPS) under real-time operating conditions; neither has been formally benchmarked for the current implementation. It is noted that the application's dashboard interface may display illustrative or placeholder text such as "94% accuracy"; any such interface text is not an experimental result and must not be interpreted as a measured accuracy value for the trained model.

VIII. RESULTS AND DISCUSSION

The trained Sign-to-Text model attains a training accuracy of 92.97% (training loss 0.255964) and, on the 20% validation split held out during training using scikit-learn's `train_test_split` (random seed 42) as described in Section IV, a validation accuracy of 92.42% (validation loss 0.293553),

Table 5: Experimental Results

Metric	Result
Training Accuracy	92.97%
Training Loss	0.255964
Validation Accuracy	92.42% (0.9242)
Validation Loss	0.293553
Validation Precision	93.56%
Validation Recall	92.42%
Validation F1-score	92.79%
Test Accuracy	Not evaluated; no independent test dataset was used in the current implementation
Average Inference Latency	Not formally benchmarked in the current implementation
Recognition FPS	Not formally benchmarked in the current implementation



Figure 5: Confusion matrix generation run for the Sign-to-Text classifier, confirming computation over the full 26,582-sample validation set across all 110 classes and export of `confusion_matrix_validation.png`.

with a validation precision of 93.56%, a validation recall of 92.42%, and a validation F1-score of 92.79%. These validation-split metrics are computed on a held-out subset drawn from the same underlying dataset distribution as the training data, and the project's own training records explicitly note that this figure is not a measurement of real-world webcam accuracy. Accordingly, the 92.42% validation accuracy should not be interpreted as, or reported as, a test accuracy or a general-purpose accuracy figure for the deployed system; it reflects performance on held-out samples from the training distribution only. An independent test set, collected separately from the training and validation data (e.g., under varied signers, backgrounds, or lighting), has not yet been constructed for this project. Consequently, test accuracy is not evaluated in the current implementation, and average inference latency and recognition FPS have not been formally benchmarked; all three remain reported as such in Table 5 rather than as estimated numerical values.

The confusion matrix below was generated from the trained classifier's predictions on the full 26,582-sample validation split (not an independent test set), confirming the reported validation-level performance across all 110 output classes. Training and validation accuracy/loss curves have not yet been generated for this implementation and are therefore not included as figures.

The generated confusion matrix



Figure 6: Sign-to-Text interface, showing the live camera workspace, MediaPipe-based hand tracking status, and the recognition confidence panel prior to a detected gesture.



Figure 7: Text-to-Sign interface, showing the text composer and the corresponding ISL sign animation played back for the entered text.

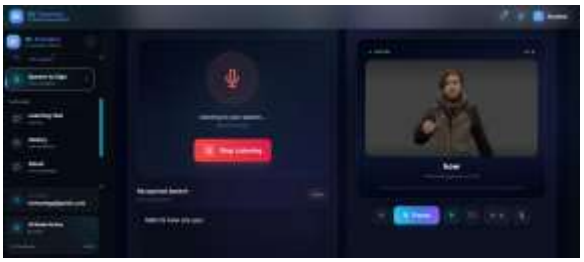


Figure 8: Speech-to-Sign interface, showing live speech transcription alongside the resulting ISL sign video playback.

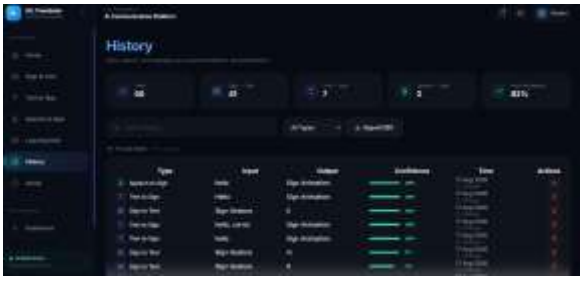


Figure 9: Translation History interface, showing per-mode translation counts, average recognition confidence, and a searchable, exportable log of past Sign-to-Text, Text-to-Sign, and Speech-to-Sign translations for the authenticated user.

(confusion _matrix _validation.png) is available for detailed class-wise inspection; a full discussion of class-wise behavior (including confusion between visually similar signs and the effect of the class-imbalance limitation described in Section IX) is left for future work once an independent test-set evaluation is also available, alongside the effect of real-time operating conditions on recognition quality, sensitivity to environmental variation such as lighting and background, and general system usability under that independent evaluation. No test-set numerical claim about model performance is made anywhere in this paper; the only performance figures reported are the training- and validation-split metrics explicitly given above, and test accuracy, inference latency, and recognition FPS remain reported as not evaluated or not benchmarked, respectively, until an independent evaluation is carried out.

IX. LIMITATIONS

The dataset contains 195,721 source images organized into 110 sign classes. After MediaPipe-based hand landmark extraction and validation, 132,906 valid 126-dimensional landmark samples were retained, representing 108 of the 110 target classes. The classes “Stop” and “ok fine” produced no valid landmark samples in the generated training dataset; consequently, although the model’s output layer retains 110 units to match the full target vocabulary, it has not been trained on any positive examples for these two classes and cannot be expected to recognize them reliably. In addition, several classes had very few valid samples, including “Theif” and “nothing” with one sample each, “I_m Good” and “There is Gun” with two samples each, and “Bridge” and “Victory” with four samples each; the exact number of source images that failed or were skipped during the original landmark-extraction run was not preserved and is therefore not reported. This uneven class representation is a limitation of the current dataset and should be considered when interpreting class-specific and overall validation performance.

Beyond this dataset imbalance, the current implementation has several further limitations relevant to interpreting its scope. The Sign-to-Text module performs isolated-sign recognition; continuous, sentence-level recognition is not currently the primary recognition task and is discussed as future work in Section X. No independent test set, separate from the training/validation split, has yet been constructed; the 92.42% validation accuracy reported in Section VIII is measured on a held-out split drawn from the same dataset distribution as the training data, and, as explicitly noted in the project’s own training records, is not a measurement of real-world webcam accuracy under novel signers, backgrounds, or lighting conditions. Average inference latency and recognition FPS have likewise not been formally benchmarked for the current implementation. Visually similar signs, signer variation, lighting variation, camera positioning, background variation, and partial hand occlusion can all be expected to affect recognition accuracy, though the precise extent of these effects has not yet been formally measured for this implementation. The Text-to-Sign and Speech-to-Sign modules depend on the coverage of the current sign-video library, so words or phrases without a corresponding sign resource cannot currently be represented. The Speech-to-Sign pathway inherits any transcription errors made by its underlying speech-recognition component. Finally, the password-reset token display in the current implementation is appropriate for a college demonstration setting but would require secure email-based delivery, rather than direct display, in a production deployment.

X. CONCLUSION AND FUTURE WORK

This paper presented the design and implementation of a real-time, web-based, bidirectional Indian Sign Language translation system integrating Sign-to-Text, Text-to-Sign, and Speech-to-Sign communication modes within a single au-

Table 6: Functional Capabilities of the Proposed System

Mode	Input	Processing	Output
Sign-to-Text	Webcam gesture	MediaPipe + TensorFlow/Keras	Text
Text-to-Sign	Text	ISL sign mapping	Sign video
Speech-to-Sign	Speech	Speech recognition + sign mapping	Sign video

thenticated application. The implemented system combines MediaPipe Hands for hand landmark extraction, a TensorFlow/Keras model for sign classification, a FastAPI backend with SQLAlchemy/SQLite persistence, WebSocket-based real-time communication, and a React/Vite/TypeScript frontend, together with user authentication, translation history, and a usage dashboard.

The following directions have been identified for future work and are explicitly distinguished from the functionality currently implemented: expanding the recognition vocabulary and the sign-video library toward broader ISL lexicon coverage; extending recognition from isolated signs to continuous, sentence-level recognition; incorporating two-hand gesture modeling; training on larger and more diverse datasets collected under varied recording conditions; exploring improved sequence models, including Transformer-based approaches, for both recognition and translation, building on the direction demonstrated by continuous-translation work such as [9]; developing a mobile deployment of the platform; improving the robustness of the speech-processing component; exploring personalized, signer-adaptive recognition; and improving robustness under lighting- and background-variation conditions. Continuous sentence-level Transformer-based translation, in particular, is identified here as future work and is not part of the current implementation.

REFERENCES

[1] S. Katoch, V. Singh, and U. S. Tiwary, "Indian Sign Language recognition system using SURF with SVM and CNN," *Array*, vol. 14, Art. no. 100141, 2022, doi: 10.1016/j.array.2022.100141.

[2] P. Sharma and R. S. Anand, "A comprehensive evaluation of deep models and optimizers for Indian sign language recognition," *Graphics and Visual Computing*, vol. 5, Art. no. 200032, 2021, doi: 10.1016/j.gvc.2021.200032.

[3] B. Subramanian, B. Olimov, S. M. Naik, S. Kim, K.-H. Park, and J. Kim, "An integrated MediaPipe-optimized GRU model for Indian sign language recognition," *Scientific Reports*, vol. 12, Art. no. 11964, 2022, doi: 10.1038/s41598-022-15998-7.

[4] T. Nehra, D. Saisanthiya, and A. Modi, "Indian Sign Language (ISL) Recognition And Translation using Mediapipe and LSTM," in *2023 World Conference on Communication & Computing (WCONF)*, Raipur, India, 2023, pp. 1–5, doi: 10.1109/WCONF58270.2023.10235113.

[5] C. Aparna and M. Geetha, "CNN and Stacked LSTM Model for Indian Sign Language Recognition," in *Machine Learning and Metaheuristics Algorithms, and Applications (SoMMA 2019)*, Communications in Computer and Information Science, vol. 1203, Springer, Singapore, 2020, doi: 10.1007/978-981-15-4301-2_10.

[6] K. Navendu and V. Sahula, "Word Level Sign Language Recognition Using MediaPipe and LSTM-GRU Network," in *2024 IEEE International Symposium on Smart Electronic Systems (iSES)*, New Delhi, India, 2024, pp. 13–18, doi: 10.1109/iSES63344.2024.00014.

[7] P. Rawat, P. Kumar, V. K. Tamta, and A. Kumar, "A Comprehensive Approach to Indian Sign Language Recognition: Leveraging LSTM and MediaPipe Holistic for Dynamic and Static Hand Gesture Recognition," *EAI Endorsed Transactions on AI and Robotics*, vol. 4, 2025, doi: 10.4108/airo.8693.

[8] R. Damdoo and P. Kumar, "An integrative survey on Indian sign language recognition and translation," *IET Image Processing*, vol. 19, 2025, doi: 10.1049/ipr2.70000.

[9] R. Damdoo, P. Kumar, and R. Gogoi, "End-to-end sentence-level Indian sign language translation with ISH-NEWS dataset and transformer model," *Scientific Reports*, 2026, doi: 10.1038/s41598-026-60893-0.