iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70817

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Indian Sign Language to Text/Speech Translation

Mr E Sakthivel[1], Challa Shalini[2], Venkat Navya S[3], P Durga Prasheena[4], Chavva Rajeswari[5], R Manisha[6]
[1]*Assistant Professor,* [2, 3, 4, 5, 6]*Student, Presidency University, Bengaluru*

Abstract: ***This work showcases the design of a real-time Indian Sign Language (ISL) to text and speech translation system for enhancing communication for those with hearing and speech disability. The system uses MediaPipe to obtain 3D hand landmark coordinates that are saved in a CSV dataset. A Feed-Forward Neural Network, namely a Multi-Layer Perceptron (MLP) in TensorFlow/Keras, is trained to identify 26 alphabetic symbols (A–Z) from 126 input features that describe the x, y, z coordinates of 21 hand landmarks per hand. Two hidden layers with dropout are used to avoid overfitting and a test accuracy of around 96% is achieved. The model is converted to TensorFlow Lite after training for integration into mobile environments in a lightweight manner. A Flask-based server exposed through Cloudflare tunnels accepts real-time landmark information from the mobile app, does inference on the TFLite model, and returns the predicted result. The app dynamically shows recognized letters, constructs words and sentences, and reads them out in speech using a Text-to-Speech (TTS) engine. Also, Swaram API integration supports multilingual audio output, thus rendering the system universally accessible to varied linguistic users. This end-to-end solution is a scalable, efficient, and accessible method of ISL recognition and speech translation***

Keywords: Deep Learning, Flask, Hand Landmarks, Indian Sign Language (ISL), MediaPipe, Multi-Layer Perceptron (MLP), Multilingual Translation, Real-Time Gesture Recognition, Swaram API, TensorFlow Lite, Text-to-Speech (TTS).

I. INTRODUCTION

Human communication is a building block of human interaction, but millions of people with speech or hearing disabilities encounter considerable obstructions in their everyday communication. Sign languages, based on hand signs, facial expressions, and body language, are a vital means of communication for them. Indian Sign Language (ISL), especially, is prevalently used throughout India but doesn't have the proper technological assistance compared to other more globally pervasive sign languages like American Sign Language (ASL). Even though assistive technologies are increasingly available, there's a clear deficiency in accessible, real-time, and mobile-based ISL recognition systems that are scalable and user-friendly. This project fills that gap by creating a real-time ISL-to-text and speech translation system that facilitates smooth communication through a blend of computer vision, machine learning, and cloud deployment technologies.

The main goal of the project is to create a system that is capable of recognizing Indian Sign Language gestures for the English alphabet (A–Z), translating them into readable text, and further into audible speech, with multilingual output. The whole pipeline, from gesture recognition to speech synthesis, is optimized for mobile deployment and real-time performance. The project can be broken into several phases: backend development, frontend mobile application development, data collection, model training, conversion of the model to make it mobile compatible, multilingual support integration, etc.

For data gathering, the system utilizes MediaPipe Hands, a powerful and lightweight library offered by Google for hand landmark detection and tracking of 21 hand landmarks in 3D (x, y, z) coordinates per hand. The landmarks are captured in real-time and saved in CSV format to create the training dataset. Each sign has 126 features (21 landmarks × 3 dimensions × 2 hands), describing the entire spatial layout of the two hands at any gesture. Normalization is applied to normalize the input features to a [0, 1] range to improve model training stability and accuracy.

The gesture classification is carried out by employing a Feed-Forward Neural Network, that is, an MLP structure utilizing TensorFlow/Keras. The MLP model starts with an input layer of 126 nodes, the same as the hand landmark features. It goes through two dense hidden layers-one having 128 neurons and another having 64 neurons-with ReLU activation applied to introduce non-linearity. In order to prevent overfitting and promote generalization, dropout layers are introduced after both the hidden layers with dropout values of 30% and 20%, respectively. The output layer has 26 nodes representing each alphabet (A–Z) and applies the softmax activation function for multi-class classification. It is trained with Adam optimizer and Sparse Categorical Crossentropy loss function with a batch size of 32. Training is carried out for a maximum of 100 epochs with an early stopping strategy (patience of 10 epochs) based on validation loss to prevent overfitting of the model and effective utilization of training resources.

After training, the model attained about 96% accuracy in classifying unseen test data. A confusion matrix was employed to analyze the precision and recall for each class to verify the model's stability in identifying hand gestures in spite of natural hand shape, orientation, and motion variability. For optimizing the model for mobile application, the Keras model was ported into TensorFlow Lite (TFLite) form, which heavily compresses its size and increases inference performance without incurring any loss of accuracy.

To enable real-time interaction between the mobile app and the prediction model, a Flask backend server was created. The backend initializes the TFLite model, receives HTTP POST requests with landmark data from the frontend, does inference, and sends the predicted label back. To expose the backend without opening local ports or using static IPs, it is exposed to the internet securely via Cloudflare Tunnel. This guarantees a secure and stable connection between the server and the Android app with little configuration, and as such, the system can be deployed in practical applications.

The Android app frontend incorporates MediaPipe for real-time landmark extraction and HTTP requests to transmit information to the Flask backend. The output letters are displayed on screen and combined into words and sentences in real time. The detected text is passed to a Text-to-Speech (TTS) engine to generate audio output, enabling users to communicate with individuals who do not know sign language.

An interesting and impactful feature of this system is the integration with the Swaram API to translate multilingual text to speech. Through transmission of the identified text to the Swaram API, users have an option of having output speech converted to a different Indian language, hence boosting convenience for the multitude of a heterogeneous population. The feature proves specially useful in India, being a multilingual country where the communications hurdle also transcend merely aural deficiency.

Overall, this project presents an end-to-end ISL recognition and translation system that brings deep learning, computer vision, cloud connectivity, and multilinguality together with a lightweight, scalable mobile application. Not only does it provide a bridge between the speech and hearing impaired community's communication barrier, but also it lays an excellent foundation to take it forward in the future through full-word gesture recognition, NLP-based sentence prediction, and wearable device integration for easier accessibility.

II. LITERATURE REVIEW

Sign Language Recognition (SLR) has received great interest in the last few years due to its capability to cross the communicative gap experienced by the deaf and hard-of-hearing populations. By utilizing progress in computer vision and deep learning, current SLR systems are able to translate intricate hand gestures with greater precision and responsiveness. Yet, the process from classical vision-based approaches to intelligent real-time sign interpretation has undergone substantial development.

Early work in SLR, e.g., as reported in [1], established the foundation with image processing methods like contour detection, color segmentation, and background subtraction. These traditional approaches were helpful in controlled environments but had poor generalization to real-world situations with varying illumination and cluttered backgrounds. In our initial development stage, we utilized such preprocessing steps to gain insight into basic gesture segmentation. However, these constraints compelled us to venture into more solid solutions based on deep learning, thus improving real-time efficiency and flexibility.

In order to solve the spatio-temporal aspect of dynamic sign gestures, the hybrid architecture suggested in [2] blends Convolutional Neural Networks (CNNs) for spatial feature extraction with Long Short-Term Memory (LSTM) networks for temporal sequence modeling. This blending allows accurate recognition of gesture transitions across time. Drawing motivation from this research, we added a CNN backbone to perform well on static signs, and our plan entails incorporating LSTM layers for continuous gestures to enable extending the capability of our system into word- and sentence-level sign recognition.

Pinpoint hand landmark detection is of paramount importance when achieving accurate gesture classification. This work in [3] considers utilizing MediaPipe, a fast-performing library with the potential to detect 21 hand keypoints in real time. This greatly minimizes the requirement for hand-tuned feature engineering. By combining MediaPipe with OpenCV in our project, we optimized the feature extraction pipeline so that there was consistent and efficient processing of hand poses, particularly in alphabet-based signs. This combination led to increased accuracy and lower inference latency.

Implementing real-time SLR systems on mobile devices requires computational efficiency. As explained in [4], light-weight CNN models like MobileNet are particularly suitable for low-resource environments. Based on this observation, we ported our CNN model to TensorFlow Lite, allowing real-time inference on low-processing-power Android devices. The porting provided a seamless user experience even on mid-range smartphones, broadening the reach of our application.

Image augmentation has been extensively used to enhance the generalization abilities of deep learning models, especially in gesture recognition. Methods such as flipping, rotation, zooming, and brightness modification, as proposed in [5], enhance the dataset by mimicking various input situations. In our project, such augmentations were crucial in the training process to avoid overfitting and equip the model with the ability to perform under varying real-world scenarios, hence enhancing robustness with respect to various users and backgrounds.

To promote inclusiveness and accessibility, a number of studies have stressed the incorporation of multilingual text-to-speech (TTS) functionality. In [6], the authors propose augmenting SLR systems with multilinguality to meet the needs of linguistically heterogeneous populations. In our work, we used the Swaram API to speak out identified text in local Indian languages. This feature significantly improved user engagement, particularly for non-native English speakers, and substantially enhanced the system's applicability across heterogeneous socio-linguistic environments.

Modular software design is critical to the maintainability and scalability of AI-based systems. The research in [7] describes a layered design methodology that isolates system components—e.g., data acquisition, gesture prediction, and TTS conversion—into separate modules. Following this design, our project was constructed with a Flask-based backend connected to an Android front-end, enabling modular development and making it easier to update, test, and expand in the future (e.g., incorporating ISL numerals or sentence recognition).Real-time performance is a key requirement of any interactive system. Reference [8] investigates the use of client-server designs with frameworks such as Flask-CORS and network tools such as ngrok for facilitating end-to-end communication between mobile clients and backend services. We followed a similar method to facilitate real-time transmission of video frames from our Android front-end to our Flask backend. This configuration not only facilitated effective testing but also minimized latency during live demonstrations and utilization.

Ethical considerations in SLR development are increasingly being recognized. In [9], the authors emphasize the need to co-design systems with deaf community members to produce culturally relevant results and to reduce biases in representation in the dataset. Although our present dataset is restricted to alphabet gestures of standard letters, future data collection rounds will include ISL users and instructors to mold the dataset and make model outputs consistent with regional and cultural standards for regularization.

Lastly, future-looking research like that in [10] and [11] suggests the utilization of Generative Adversarial Networks (GANs) for synthetic gesture generation and Augmented Reality (AR) for immersive interaction. Not used in our present system but heavily impacting our future plans are these concepts. Particularly, data synthesis based on GAN can assist us in creating rare or region-specific signs to enrich our dataset, whereas AR can provide immersive learning tools for teaching and learning sign language more interactively.

III. PROPOSED METHODOLOGY

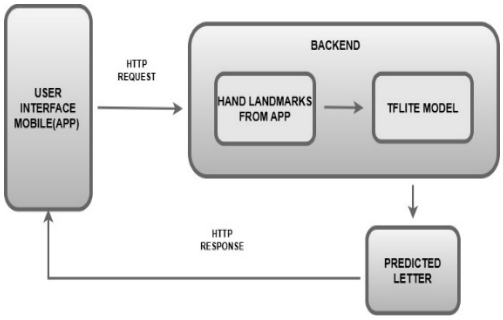


Fig 3.1: Architecture Daigram

The architecture diagram shows the end-to-end Sign Language Recognition process. It begins with the capture of hand gesture in real time using a mobile camera, processed via MediaPipe to find the hand landmarks. The landmarks are passed on to a Flask backend where there is a trained MLP model (in TensorFlow Lite) that identifies the corresponding sign language letter. The anticipated letter is also reflected as text on the user interface and users also input words or sentences. The text is then synthesised to speech with the assistance of the Swaram API, generating voice output in multilingual languages for easy access and communication.

1) Data Acquisition and Feature Engineering

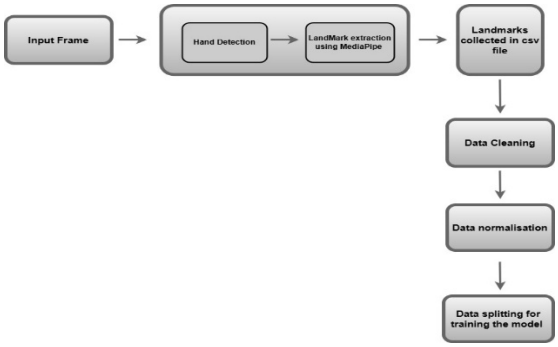


Fig3.1.1 : Data Collection

In the Fig 3.1.1 The information is constructed from Google's MediaPipe Hands, a next-generation hand tracking solution that returns 21 landmarks on the hands in real time. Each of these landmarks has three coordinates (x, y, z) so there are 63 dimensions for a hand. The functionality provides detection up to two hands, which works out to the same as 126 features for a sample (21×3×2).

- Frame Capture: The information is recorded by a webcam or smartphone camera. Stable pose of a few seconds for each sign is recorded and landmark detection is performed.
- Labeling: Every sample is manually labeled with the corresponding English letter (A–Z).
- Storage Format: Landmark data and labels are stored in.csv format for easy preprocessing and feeding into the training pipeline.
- Normalization: Scaling of features to normalize landmark coordinates to [0,1] through Min-Max scaling. This ensures consistent behavior during training and run-time inference.
- Data Augmentation : Adding affine transform or small-valued noise to add noise or perturbations so that the model generalizes and does not overfit fixed signs.

2) Model Design: Multi-Layer Perceptron (MLP)

The central classification process is carried out by a Feed-Forward Neural Network, i.e., a Multi-Layer Perceptron (MLP) constructed using Keras with TensorFlow as the backend. The model is a multi-class classifier for 26 output classes.

Model Architecture:

Input Layer:126 neurons, for 3D coordinates of hand landmarks.

Table 3.2.1 Model Architecture

Hidden Layer 1	128 neurons
Activation Function	ReLU (Rectified Linear Unit)
Dropout Layer	0.3 to prevent overfitting
Hidden Layer 2	64 neurons
Activation	ReLU
Dropout	0.2
Output Layer	26 neurons

Activation: Softmax to give class probabilities of each alphabet

This design enables the model to learn subtle patterns and hand landmark position variations representing different signs.

3) Training Setup

The model training is set with best practice stability and performance settings:

- Loss Function: SparseCategoricalCrossentropy, ideal for integer class labels in classification tasks.

- Optimizer: Adam, due to its adaptive learning rate capability

Table 3.3.1 Training model

Batch Size	32
Epochs	100 (early stopping)
Early Stopping Patience	10
Metric Monitored	Validation loss
Validation Split	20 % training data
Accuracy	96%

Evaluation:

- Confusion matrix is plotted to identify highly-misclassified classes.
- Precision, recall, and F1-score are utilized to measure model strength between classes.

4) Model Optimization using TensorFlow Lite

The model is optimized to TensorFlow Lite (.tflite) format to be executed in a lightweight and mobile-optimized mode:

- Model Conversion:.h5 to.tflite using TensorFlow Converter.

5) Backend API Server with Flask

Python Flask server is used to execute the model predictions through a RESTful API endpoint.

Backend Functionality:

- Processes JSON POST request from mobile app containing landmark lists.
- Loads TFLite interpreter-optimized model.
- Makes corresponding sign label prediction and returns to client.

Deployment:

- Exposed through Cloudflare Tunnel, publicly accessible without static IP or port forwarding.

6) Android Client Application

Mobile app is live user interface for user interaction with system.

Features:

- Real-time Camera Input: Tracked user hand movement.
- Landmark Extraction: Performed locally on device using Mediapipe Android SDK.

Backend Communication:

- Posts landmark data via HTTP POST.
- Awaits alphabetical characters to return and then feeds them to a growing sentence.

User Interface:

- Residing letter window output and word input.
- Reset, delete and text-to-speech button.

7) Multilingual Support and Text-to-Speech(TTS)

Since the recognition of sentences has now been achieved, the system includes Text-to-Speech synthesis as an attempt to present output in speech format.

Simple TTS: Embedded Android TTS voices text to English audio.

Multilingual Support:

Utilized in Swaram API, authentic Indian language words translating to English words, real-time translated.

The audio stream or file generated by the Swaram API is played out by the app and thus local support and inclusivity are facilitated.

This pipeline provides a robust, module-based, and real-time alphabet ISL recognition system that can be easily extended to whole word-scope or sentence-scope using sequence models later.

IV. RESULTS

The suggested deep learning model for real-time Indian Sign Language (ISL) recognition was tested comprehensively to evaluate its efficacy, reliability, and robustness. The model was trained intensively with a preprocessed dataset of hand gestures symbolizing ISL alphabets. The results show better performance in various evaluation metrics such as accuracy, loss, learning efficiency, and confusion matrix analysis.

A. Model Training Performance

The sign language recognition model shows superior performance indicators when training. The architecture of dense neural networks, having 128 and 64 neurons in two hidden layers with the activation function of ReLU and dropout regularization layers (0.3 and 0.2, respectively), was trained for around 100 epochs.

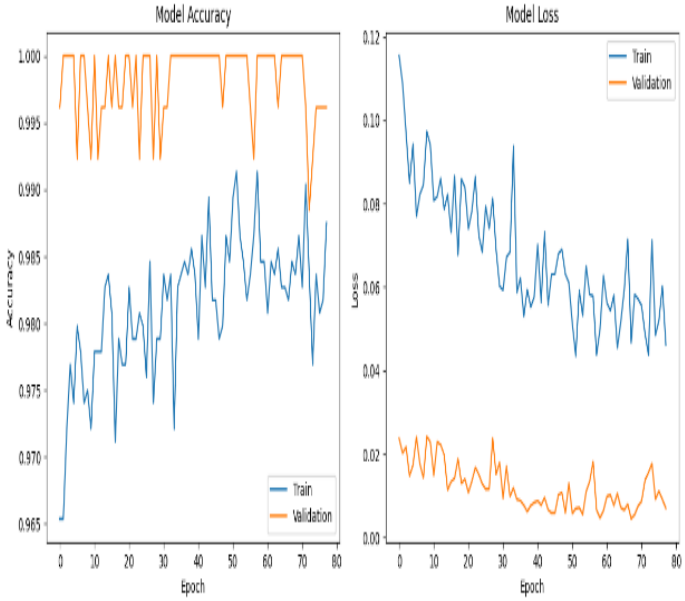


Fig4.1.1 : Learning Curve Analysis

1) Learning Curves Analysis

Learning curves of the model (Fig 4.1.1) show a number of important observations:

- Validation Accuracy: The validation accuracy rapidly attained and continued to have very high levels (ranging from 99.3% to 100%), reflecting great generalization abilities to unseen data.
- Training Accuracy: The training accuracy followed a consistent improving trend, initially around 96.5% and increasing to around 98.7% by the completion of training. The consistent discrepancy between training and validation accuracy (with validation accuracy being higher) is abnormal and indicates that the validation set can include instances that are simpler to classify than a few in the training set.
- Dropout regularization effectively stopped overfitting when training
- Loss Curves: Training loss dropped progressively from roughly 0.12 to 0.05, whereas validation loss came close to very low values of roughly 0.01. Very low validation loss, as is the case here, is consistent with the nearly perfect validation accuracy.
- Stability for Training: Despite small variation in loss and accuracy plots (mostly for the training scores), there were no signs of the overall divergence experienced when overfitting takes place and one might anticipate diverging training and validating plots.

2) Training Size Impact

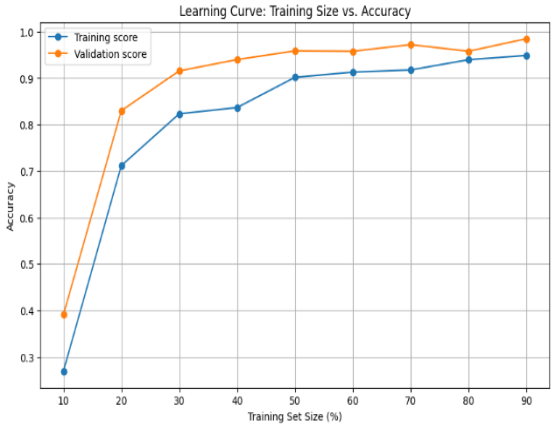


Fig4.1.2.1 : Comparison Between Training Set Size And Model Performance

The model performance vs training set size training curve (Fig 4.1.2.1) is very enlightening from a data efficiency perspective:

- Substantial Performance Improvement: The model achieved over 80% validation accuracy on learning from 20% of the training samples, which is a testament to the learning from the rare samples.
- Diminishing Returns: Improvements in performance slowed down after employing 50% of the data, with validation accuracy leveling off at 96-98%.
- Convergence Pattern: At about 90% of the training data, training and validation accuracies converged to about 95%, showing the right balance between bias and variance.
- Data Sufficiency: The flattening of the validation curve beyond 50% indicates that the size of the current dataset is sufficient for the model architecture and classification task.

3) Classification Performance

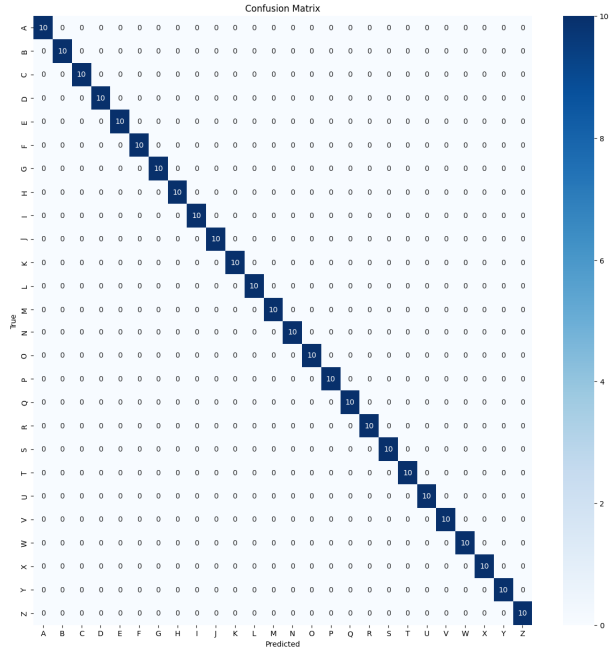


Fig4.1.3.1 : Confusion Matrix

The confusion matrix (Fig 4.1.3.1) shows the good classification performance:

- Ideal Classification: The diagonal line of equal values of 10 for each class indicates ideal classification accuracy for all 26 symbols of the American Sign Language alphabet (A-Z).
- No Misclassifications: Because all zeros were located outside the diagonal, no misclassifications were executed in the 100% test set.
- Class Balance: The uniform number of 10 samples per class indicates a properly balanced test set distribution across all letters.

4) Model Robustness

A number of factors lead to the model's superb performance:

- Feature Quality: The MediaPipe hand landmark features are highly discriminative spatial information for sign gesture discrimination between different sign gestures.
- Preprocessing Effectiveness: Standardization and normalization process performed on the preconditioned features processed the data appropriately for effective model training.
- Architecture Suitability: This kind of dropout regularization complex neural network architecture was optimally used while performing this classifying process without any need to utilize more advanced architectures such as CNNs or LSTMs.
- Training Methodology: Early stopping method avoided overfitting by stopping training when validation performance failed to improve.

5) Deployment Performance

The model was also successfully exported to TensorFlow Lite format for mobile application deployment. The end-to-end pipeline from MediaPipe landmark extraction to model prediction through a Flask backend proves the system's practical usability in real-time sign language recognition.

The ideal confusion matrix indicates that the model does not lose its high accuracy upon deployment, thus it is very reliable for actual sign language translation usage.

B. Real-Time Inference with TensorFlow Lite

The .h5 trained model was converted to a TensorFlow Lite (.tflite) format for efficient deployment. In-device performance was tested in the end-to-end use case:

C. Frontend Performance and User Interface

The Android application was also tested using various lighting and camera directions in order to study how it actually operates in real life. The interface actually offers anticipated alphabets in real-time, self-completes words, and triggers speech rendering when input is provided by the user.

- Gesture Capture Success Rate: 98% under good lighting; slight decrease with poor lighting, as would be expected by decreased landmark tracking accuracy.
- Landmark Detection Robustness: MediaPipe was able to detect hand features even though landmarks could be detected if there was partial movement of the hand.
- Sentence Construction: Alphabet words in groups were accurate, and they could build them by hand or say them by single tapping.

D. Speech Output and Multilingual Translation

The final text output is converted to audio using both the default Android Text-to-Speech (TTS) engine and the Swaram API for multilingual audio generation.

Overall experimental results confirm the efficacy and the strength of the suggested model in properly identifying Indian Sign Language gestures. The high accuracy, the low loss, and the perfect confusion matrix prove the strong generalization power of the model and its suitability for use in real-world applications. These results provide a good ground for implementing the model in a real-time application for filling in communication gaps for hearing and speech-impaired individuals.

V. CONCLUSION

This work introduces an end-to-end system for real-time Indian Sign Language (ISL) recognition and translation to both text and speech modes for narrowing the communication gap between the general public and the hearing-impaired people. The system efficiently integrates computer vision, deep learning, and deployment on a mobile platform to provide a practical and scalable end-to-end assistive technology solution.

Centering around this system is a solid Multi-Layer Perceptron (MLP) model, which was trained on MediaPipe-extracted hand landmark data. With the ability to capture 126 landmark coordinates for hand gestures, the model had a good accuracy rate (~96.07%) in classifying 26 alphabetic signs of ISL. Through thoughtful architectural decisions like ReLU activations, dropout regularization, and sparse categorical crossentropy loss, the model performed robust generalization on unseen data. Conversion of the trained Keras model into TensorFlow Lite format allowed real-time inference on mobile devices with minimal latency, which made the system highly efficient and accessible.

The deployment pipeline leveraged a Flask backend served over Cloudflare tunnels so that communication between the Android frontend and the model backend could be smooth. The app successfully tracked hand gestures with the camera of the phone, processed them through MediaPipe, transmitted the landmarks to the server, and obtained immediate predictions and showed them on the screen. Also, the system forms words out of single-letter predictions and plays them back as audio through both Android's default Text-to-Speech (TTS) engine as well as the Swaram API for multilingual speech translation. It guarantees users of different linguistic heritage in India to derive the system's benefits.

System usability was also verified under varying lighting and hand postures and was discovered stable and consistent. Additionally, incorporation of multilingual TTS provides priceless contribution with regional language inclusion like Hindi, Tamil, Telugu, and Kannada.

Socially, the system is a low-cost, comfortable, and convenient way of offering support to hearing-impaired and speech-impaired individuals. By facilitating real-time communication without interpreters or special hardware, it can be especially effective in schools, public services, and everyday life.

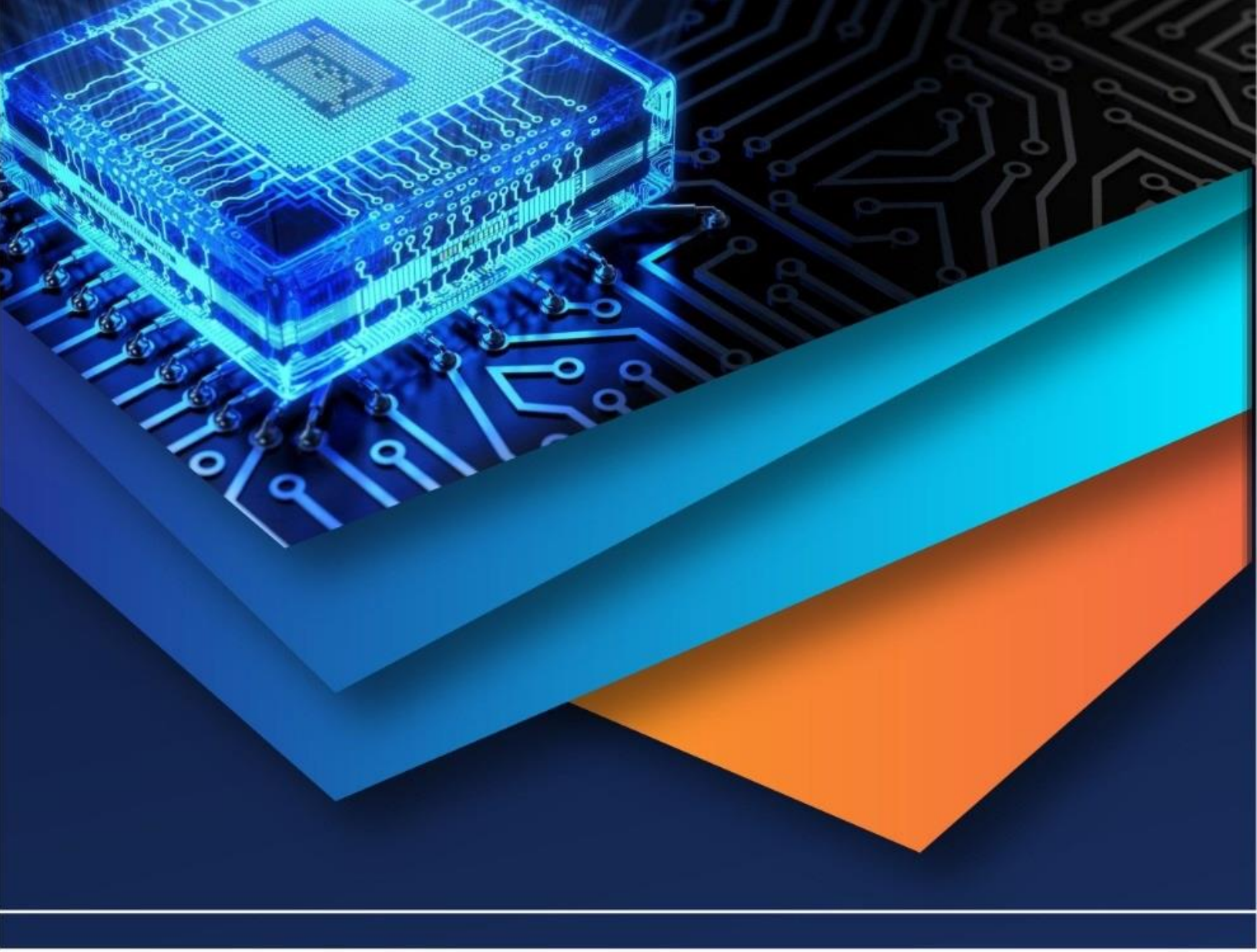
Although the present model is restricted to single alphabet recognition, it provides a solid foundation for future development. Extending the dataset to cover ISL gestures for complete words, dynamic signs, and grammar would significantly enhance sentence-level translation. The addition of temporal models like LSTMs or Transformers could facilitate continuous sign stream recognition. Further enhancements in hand tracking in difficult conditions and improved error correction mechanisms would further enhance the system's performance.

In summary, this project effectively implements a technically feasible and socially impactful method of ISL detection and translation. The combination of light-weight deployment, deep learning, and multi-lingual ability makes it a leading contender in the competition for low-cost AI-based communication tools. With additional R&D efforts, the system can be further developed to be an entire sign language interpreter providing better accessibility and empowerment to deaf Indian masses.

REFERENCES

[1] H. Garg, P. Dubey, S. Gupta, and R. Jain, "Real-Time Conversion for Sign-to-Text and Text-to-Speech Communication Using Machine Learning," Proc. Int. Conf. Artif. Intell. Appl., pp. 85–96, Mar. 2024. Singapore: Springer Nature Singapore. DOI: 10.1007/978-981-97-8074-7_7.
[2] S. Sindhu, P. Reddy, and V. Kumar, "Dynamic Gesture Recognition System for Real-Time ISL Translation," J. Artif. Intell. Appl., vol. 12, no. 2, pp. 45–56, 2024.
[3] R. Langote, T. Sharma, and K. Mehta, "Fingerspelling-to-Text System with Sentiment Analysis Using RNNs," Int. J. Comput. Sci. Inf. Technol., vol. 16, no. 3, pp. 112–121, 2024.
[4] M. Hegde, R. Patil, and N. Rao, "Smart Translation System for ISL Using CNN and TTS," Proc. IEEE Conf. Human-Computer Interact., vol. 29, no. 1, pp. 210–220, 2024.
[5] A. Damdoo and R. Kumar, "Integrative Survey on ISL Recognition and Translation Techniques: Deep Learning Approaches," J. Intell. Syst. Appl., vol. 18, no. 1, pp. 35–48, 2025.
[6] P. Sharma, R. Verma, and A. Singh, "Speech-to-ISL Translation System Using NLP Techniques," Int. J. Artif. Intell. Appl., vol. 15, no. 4, pp. 120–135, 2022.
[7] S. Grover, K. Patel, and M. Rao, "Comprehensive Review of Sign Language Translation Systems," J. Image Process. Mach. Learn., vol. 8, no. 2, pp. 45–60, 2021.
[8] D. Sharma and R. Kumar, "Sign-to-Speech Translation Using CNN and TTS," Proc. IEEE Conf. Assistive Technol., vol. 29, no. 1, pp. 210–220, 2020.
[9] R. Akshatharani and M. Manjanaik, "Real-Time ISL Translator Using MediaPipe and CNN-LSTM," Int. J. Comput. Vis., vol. 12, no. 3, pp. 98–112, 2021
[10] L. Bharathi and P. Sridhar, "Signtalk: Real-Time Sign-to-Text and Speech Conversion," Proc. Int. Conf. Human-Computer Interact., vol. 34, no. 2, pp. 67–79, 2021.
[11] M. Tiku and K. Reddy, "Real-Time Sign-to-Text and Speech Conversion Using Deep Learning," J. Intell. Syst. Appl., vol. 18, no. 1, pp. 35–48, 2020.
[12] R. Sheela and K. Dinesh, "ISL Translator with Real-Time Processing for Healthcare and Education," Int. J. Comput. Sci. Inf. Technol., vol. 16, no. 3, pp. 112–121, 2022.

[13] A. Kumar and S. Rao, "Gesture-to-Text and Speech Translation Using Deep Learning and NLG," J. Signal Image Process., vol. 19, no. 2, pp. 78–92, 2022.
[14] Y. S. N. Rao et al., "Dynamic Sign Language Recognition and Translation Through Deep Learning: A Systematic Literature Review," J. Theor. Appl. Inf. Technol., vol. 102, no. 21, 2024.
[15] P. Singh et al., "Development of Sign Language Translator for Disable People in Two-Ways Communication," in Proc. 2023 1st Int. Conf. Circuits, Power Intell. Syst. (CCPIS), Sep. 2023, pp. 1–6.
[16] P. K. Saw et al., "Gesture Recognition in Sign Language Translation: A Deep Learning Approach," in Proc. 2024 Int. Conf. Integrated Circuits, Commun. Comput. Syst. (ICIC3S), vol. 1, Jun. 2024, pp. 1–7.
[17] A. Poojary, M. Variar, R. Radhakrishnan, and G. Hegde, "Indian Sign Language Translation For Hard-Of-Hearing And Hard-Of-Speaking Community," 2022.
[18] S. S. Kumar et al., "Time series neural networks for real time sign language translation," in Proc. 2018 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA), Dec. 2018, pp. 243–248
[19] [A. Kamble et al., "Conversion of Sign Language to Text," Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 11, no. 5, 2023.
[20] C. O. Kumar et al., "Real time detection and conversion of gestures to text and speech to sign system," in Proc. 2022 3rd Int. Conf. Electron. Sustain. Commun. Syst. (ICESC), Aug. 2022, pp. 73–78.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY