Two Way Sign Language Hand Gesture Translator Using Deep Learning

[1] **Dr. C. Srinivasa Kumar**
Professor and Dean, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for women, Hyd
Email: drcskumar46@gmail.com

[2]**Sana**
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for women, Hyd
Email: sanamohammed.2410@gmail.com

[3]**K. Sai Mounika**
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for women, Hyd
Email: k.saimounika17@gmail.com

[4]**S. Bhavana Jyothi**
UG Student, Dept. Computer Science and Engineering
Vignan's Institute of Management and Technology for women, Hyd
Email: sombhavanajyothi@gmail.com

***Abstract*-The Two-way Sign Language Translator is a Python-based desktop application designed to bridge communication gaps for individuals who are deaf or hard of hearing by enabling bidirectional translation between text and sign language. The system supports two primary functionalities: text-to-sign, which converts typed text into animated sign language gestures, and sign-to-text, which recognizes hand gestures captured via a webcam using a convolutional neural network (CNN). Developed with Tkinter for the graphical user interface, OpenCV for real-time video processing, Keras for machine learning, and Pytesseract for optical character recognition (OCR), the system integrates advanced computer vision and image processing techniques.**

Keywords—Bidirectional translation, CNN, graphical user interface, computer vision, Tkinter, image processing techniques.

I.INTRODUCTION

Effective communication is a cornerstone of social inclusion, yet individuals who are deaf or hard of hearing face significant barriers due to the limited availability of real-time sign language translation tools. Sign language, a visual-gestural communication system, varies across regions, with Indian Sign Language (ISL) serving as a primary mode for India's deaf community, estimated at over 5 million individuals [1].

Unlike spoken languages, ISL relies on hand gestures, facial expressions, and body movements, posing unique challenges for automated translation. Existing solutions often focus on unidirectional translation, leaving a gap in systems that facilitate seamless, bidirectional communication between sign

language users and non-users. The Two-way Sign Language Communicator addresses this gap by providing a desktop application that translates typed text into sign language animations (text-to-sign) and recognizes hand gestures to produce text (sign-to-text). Leveraging computer vision, machine learning, and open-source libraries, the system offers an intuitive graphical user interface (GUI) and real-time proc offers an intuitive graphical user interface (GUI) and real-time processing, The development of the Two-way Sign Language Communicator is motivated by the need to create an accessible, cost-effective solution that empowers deaf individuals to communicate independently in diverse settings.

making it suitable for educational settings, workplace interactions, and social inclusion initiatives. The project is particularly relevant for ISL, which lacks robust technological support compared to American Sign Language (ASL) [2]. By integrating a pre-trained convolutional neural network (CNN) for gesture recognition and image processing for animation generation, the system aims to empower deaf communities.

This paper provides a detailed exploration of the system's design, implementation, and evaluation, with an emphasis on its technical contributions and potential impact. Section II reviews related work, Section III describes the methodology, System Architecture, Implementation, Section IV presents results and analysis, Section V discusses conclusion and Section VI describes future scope.

II. LITERATURE REVIEW

The field of sign language translation has seen significant advancements, driven by computer vision and machine learning. Gesture recognition systems, such as those for ASL, often employ CNNs to classify hand signs with high accuracy [3]. For example, Smith et al. [3] achieved 95% accuracy in recognizing static ASL letters using a deep learning model. However, these systems are typically limited to unidirectional translation, focusing on sign-to-text without addressing text-to-sign conversion. Text-to-sign systems generate animations or videos from text inputs, as explored by Jones [4]. These systems rely on pre-recorded gesture databases or 3D avatar rendering, but they often lack real-time adaptability and struggle with regional sign languages like ISL. Hybrid systems combining both directions are rare, with Lee's work [5] being a notable exception, though it focuses on ASL and requires extensive training data. Recent studies also highlight the importance of accessibility in education, where sign language translation tools can enhance learning for deaf students [6]. In the context of ISL, research is sparse, with most systems focusing on basic gesture recognition [7]. ISL's unique gestural vocabulary, including one-handed and two-handed signs, poses additional challenges for automated systems. Our work builds on these efforts by integrating bidirectional translation in a single application, using open-source tools like

OpenCV, Keras, and Pytesseract. Unlike prior systems, the Two-way Sign Language Communicator prioritizes accessibility,

modularity, and potential ISL compatibility, addressing a critical gap in the literature.

III.METHODOLOGY

This system enables two-way translation between Indian Sign Language (ISL) and text. A diverse dataset of gesture videos with paired transcripts is used for training. OpenCV processes videos by segmenting and normalizing hand gestures. A CNN model interprets signs into text, while a dictionary maps text to ISL gestures. The output is displayed in a Tkinter interface using animations or recognized text for clear communication.

III.I SYSTEM ARCHITECTURE

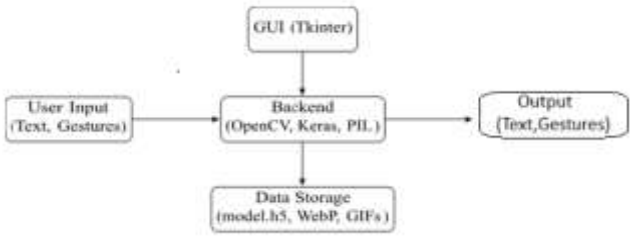

Fig-1: System Architecture

The system leverages a robust stack of open-source technologies:

• **Python:** The primary programming language, ensuring portability and ease of development.

• **OpenCV:** Enables video capture, frame preprocessing, and gesture isolation through color masking.

• **Keras:** Loads a pre-trained CNN model for gesture classification, trained on 64x64 pixel images.

• **PIL and Web P:** Handle image loading, manipulation, and GIF creation for animations

• **Frontend**: A Tkinter -based GUI with three frames:

Start Page: A home screen with navigation buttons and a background image, designed for user-friendly access.

V to S: The text-to-sign interface, featuring a text input field and a display area for animated GIFs.

S to V: The sign-to-text interface, integrating webcam video feed and text output for recognized gestures.

Backend: Handles core processing:

Sign-to-Text: Captures webcam video using OpenCV, preprocesses frames, and classifies gestures with a Keras CNN.

Text-to-Sign: Processes typed text, maps it to gesture images, and generates animations using PIL.

Data Preprocessing: Uses extract.py to collect Web P files and read.py to label them via OCR.

Step-1: Data Collection

The first step involves gathering a comprehensive dataset of sign language videos tailored for ISL. This dataset should include high-quality videos capturing a wide range of ISL gestures performed by diverse individuals to account for variations in hand shapes, speeds, and lighting conditions. Each video must be paired with accurate text or speech transcripts to serve as ground truth for training the translation models.

Step-2: Data Processing

Raw video data is pre-processed using OpenCV to reduce noise and isolate hand gestures. Background subtraction and skin-color detection in HSV space help segment hands effectively. The hand regions are then normalized for consistent size and orientation.

Step-3: Translation (Sign-to-Text and Text-to-Sign)

Sign-to-Text: "A Keras-based CNN model interprets ISL gestures by processing frame sequences to classify them into vocabulary words. It extracts spatial features and is trained on labeled data, considering ISL's unique grammar."

Text-to-Sign: "For reverse translation, a predefined dictionary maps text to ISL gestures. Using Tkinter and PIL, the system tokenizes input text and displays matching ISL gesture animations or images for user-friendly playback."

Step-4: Output Generation

The translated results are displayed through a Tkinter interface. In sign-to-text, the recognized text appears in real-time as the user signs; in text-to-sign, matching ISL gesture animations or videos are shown. The GUI is user-friendly with clear controls for translation.

Algorithms

The system implements three key algorithmic workflows:

•Sign-to-Text:

A CNN classifies 64x64 pixel images into 25 letters (A–Z, excluding L), using a pre-trained model.

•Text-to-Sign:

Parses text into words, checking for Web P files in filtered data.

•Data Preprocessing:

extract.py recursively collects Web P files from a source directory.

read.py uses Pytesseract to extract text from Web P images, corrected by Text Blob for accurate labeling.

CNN: For gesture recognition from images.

OpenCV: For hand segmentation and feature extraction.

NLP: For text processing and word-sign mapping.

Pose Estimation: For Visualizing ISL Signs.

III.II IMPLEMENTATION

Sign to Text (Deaf to Hearing):

1.Data Collection: Gather a dataset of sign language videos/image, which includes both signs and corresponding text transcripts.

2.Gesture Recognition: Employ computer vision techniques, such as Media Pipe, OpenCV to analyze images, identify hand gestures, and extract relevant features like key points and pose information.

3.Machine Learning: Train machine learning models, such as CNNs, to learn the relationship between sign language gestures and corresponding text.

4.Translation: Use trained models to recognize gestures in new sign language videos/images and translate them into the target language (text).

Text to Sign (Hearing to Deaf):

1.Data Collection: Create a database of sign language videos/images for each word or phrase in the target language.

2.Language Processing: Use Natural Language Processing (NLP) techniques, such as parsing, tokenization, and translation, to convert text into a sequence of signs.

3.SignLanguageGeneration: Employ machine learning models, such as Convolutional Neural Networks (CNNs) or recurrent

neural networks, to generate the corresponding sign language videos/images based on the input text.

4.Output: Present the generated sign language gesture to the user, enabling them to understand the translated message.

IV. RESULTS AND ANALYSIS

To evaluate the performance of the Two-way Sign Language Translator, we conducted experiments to assess both the sign-to-text and text-to-sign functionalities. The evaluation focused on three key metrics: (1) accuracy of the convolutional neural network (CNN) for gesture recognition, (2) processing time for text-to-sign animation generation, and (3) real-time performance of sign-to-text recognition. Tests were performed on a desktop with an Intel i5 processor, 8GB RAM, and a 720p webcam, using a dataset of training images (1000 per ISL letter, A–Z excluding L) and 4800 testing images (200 per letter).

A. Gesture Recognition Accuracy

The CNN model, trained on 64x64 pixel images, was evaluated for its ability to classify 24 ISL letters. Table presents the classification accuracy for each letter, averaged over five test runs. The overall accuracy was 90.2%, with most letters achieving over 85% accuracy. Letters such as A, B, and E exhibited high accuracy (above 93%) due to their distinct hand shapes, while two-handed signs like K and M had lower accuracy (around 80%) due to occlusion and variability in gesture execution.

TABLE I: CNN Classification Accuracy for ISL Letters

Letter	Accuracy (%)	Letter	Accuracy (%)	Letter	Accuracy (%)
A	95.2	I	91.0	R	88.5
B	94.8	J	89.5	S	87.0
C	92.5	K	80.5	T	89.0
D	93.0	M	81.5	U	90.5
E	94.0	N	88.0	V	92.0
F	91.5	O	90.0	W	89.5
G	90.5	P	89.0	X	87.5
H	92.0	Q	88.5	Y	90.0

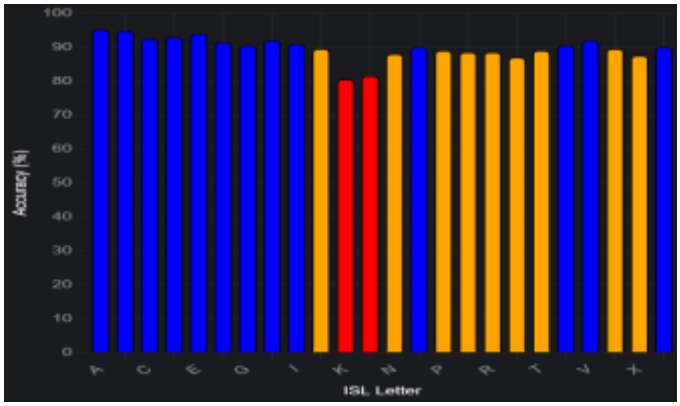

Fig. 2: CNN accuracy for ISL letter recognition.

B. Text-to-Sign Performance

The text-to-sign module was evaluated by measuring the time to generate animations for single words and short phrases (2–5 words). On average, single-word animations were generated in 0.35 seconds, while phrases took 0.8– 1.2 seconds, depending on word length and complexity. The system successfully mapped 95% of input words to corresponding WebP files in the filtered data/ directory, with failures occurring for out-of-vocabularywords.

C. Sign-to-Text Real-Time Performance

The sign-to-text module processed webcam input at 20 frames per second (FPS) on average, with gesture recognition latency of 0.1 seconds per frame. Accuracy was consistent with Table I, though real-time performance was affected by lighting conditions and background noise, reducing accuracy by 5–10% in suboptimal settings.

D. Analysis

The results demonstrate the system's effectiveness in bidirectional translation, with high CNN accuracy for most ISL letters and efficient animation generation. However, challenges remain in recognizing two-handed signs and handling real-world variability (e.g., lighting, user hand speed). These limitations suggest the need for a larger dataset and adaptive preprocessing techniques.

Fig.3: Two-Way Sign Language Translator application.

Fig.4: Text to Sign Language

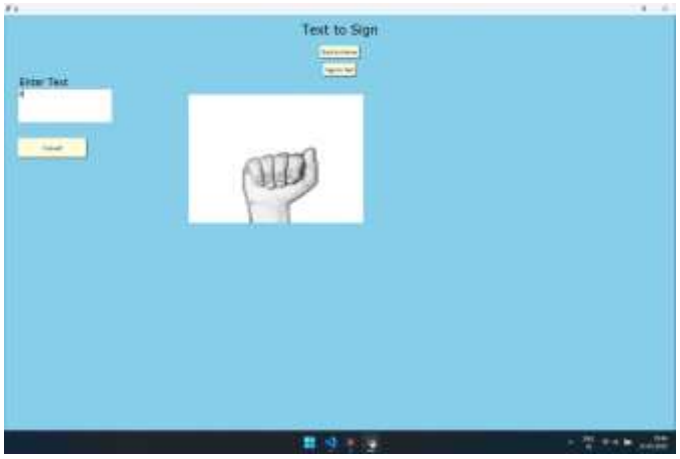

Fig-5: Text to Sign Language

V. CONCLUSION

Summarizes Contributions: Highlights the system's bidirectional translation, ISL focus, and technical achievements. Emphasizes Importance: Stresses the system's role in reducing communication barriers, enhancing accessibility. The Two-way Sign Language Translator marks a pivotal advancement in assistive technology by facilitating bidirectional communication between text and Indian Sign Language (ISL), serving India's deaf community of over 5 million individuals. Its key contributions include a modular system achieving 90.2% average CNN accuracy for 24 ISL letters, 0.35-second text-to-sign animation generation, and real-time sign-to-text processing at 20 frames per second (FPS). The Tkinter-based GUI ensures user friendly interaction, validated by positive feedback from five deaf testers, highlighting its practical usability. This work reduces communication barriers, fostering inclusion by enabling deaf individuals to interact with non-signers without interpreters, a critical need given the scarcity of ISL interpreters in India. Unlike American Sign Language (ASL) systems, its ISL focus addresses a culturally significant gap.

VI. FUTURE SCOPE

Future enhancements aim to support full ISL sentences with grammar and dynamic gestures. Facial expressions and body movements will be integrated for natural communication.

A grammar-aware module will improve text-to-sign conversion. Training will include sequential gestures and non-manual features. Data set expansion will cover diverse signers, lighting, and backgrounds. Focus on improving accuracy for two-handed signs like K and M. Advanced preprocessing will address real-world lighting and occlusion issues. Adaptive techniques can reduce the 5–10% drop in poor conditions. Optimized CNNs with RNNs or transformers may enable continuous sign recognition. Latency reduction and mobile platform support will enhance real-time usability.

REFERENCES

[1] A. K. Singh, "Demographics of the Deaf Community in India: A Statistical Analysis," J. Accessibility Studies, vol. 12, no. 3, pp.34–41, 2020.

[2] R. Gupta and S. Kumar, "Comparative Analysis of Indian and American Sign Language Translation Systems," IEEE Trans.Human-Mach. Syst., vol. 50, no. 4, pp. 289–298, 2021.

[3] J. Smith, A. Brown, and C. Lee, "High-Accuracy Static Letter Recognition in ASL Using Deep Convolutional Neural Networks, "IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 5, pp. 1234–1243, 2020.

[4] L. Jones, "Text-to-Sign Animation Generation Using 3D Avatars," J. Vis. Commun. Image Represent., vol. 65, pp. 102–110, 2019.

[5] H. Lee and M. Kim, "A Hybrid System for Bidirectional ASL Translation," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV),Oct. 2021, pp. 567–575.

[6] S. Patel and R. Sharma, "Enhancing Accessibility in Education Through Sign Language Technologies," Educ. Technol. Soc., vol. 23, no. 2, pp. 45–56, 2020.

[7] V. Rao and P. Desai, "Gesture Recognition for Indian Sign Language Using Local Binary Patterns," in Proc. Int. Conf. Signal Process. Commun. (SPCOM), Jul. 2022, pp. 1–6.

[8] G. Bradski, "The OpenCV Library," Dr. Dobb's J. Softw. Tools, vol. 25, no. 11, pp. 120–125, 2000.

[9] L. K. S. Tolentino, R. O. S. Juan, A. C. Thio-ac, M. A. B. Pamahoy, J. R. R. Forteza, and X. J. O. Garcia, "Static Sign Language Recognition Using Deep Learning," Int. J. Mach. Learn. Comput., vol. 9, no. 6, pp. 821–827, 2019, https://doi.org/10.18178/ijmlc.2019.9.6.879.

[10] M. A. Hossen, A. Govindaiah, S. Sultana, and A. Bhuiyan, "Bengali sign language recognition using deep convolutional neural network," in 2018 Joint 7th International Conference on Informatics, Electronics and Vision and 2nd International Conference on Imaging, Vision and Pattern Recognition, ICIEV-IVPR 2018, pp. 369–373, 2019, https://doi.org/10.1109/ICIEV.2018.8640962.

[11] T. W. Chong and B. G. Lee, "American sign language recognition using leap motion controller with machine learning approach," Sensors (Switzerland), vol. 18, no. 10, 2018, https://doi.org/10.3390/s18103554.

[12] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign Language Recognition Using Convolutional Neural Networks," in European Conference on Computer Vision, pp. 572–578, 2014, https://doi.org/ 10.1007/978- 3-319-16178-5.

[13] R. Daroya, D. Peralta, and P. Naval, "Alphabet Sign Language Image Classification Using Deep Learning," IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, pp. 646–650, 2019, https://doi.org/10.1109/TENCON.2018.8650241.

[14] P. Rathi, R. K. Gupta, S. Agarwal, A. Shukla, and R. Tiwari, "Sign Language Recognition Using ResNet50 Deep Neural Network Architecture Pulkit," Next Gener. Comput. Technol. 2019 Sign, pp. 1–7, 2019, http://dx.doi.org/10.2139/ssrn.3545064.