AI-Based Sign Language Translator

YOGESH BALAJI REDDY[1], PRATHAMESH NANASAHEB RAUT[2], ANIKET SNDIPAN TANDALE[3], PRAJESH VIKAS SURWASE[4], PROF. D. J. WAGHMARE[5]
[1, 2, 3, 4, 5]Department of Computer Science and Engineering, Shri. Tuljabhavani College of Engineering, Tuljapur, India

***Abstract-* Sign language is the primary communication method for deaf and hard-of-hearing (DHH) individuals, yet it remains unfamiliar to most non-DHH people, creating a significant communication gap. To address this, we propose an AI-powered Sign Language Translation system that utilizes computer vision and deep learning to interpret hand gestures in real time and convert them into readable text. Our system is based on a Convolutional Neural Network (CNN) model trained on American Sign Language (ASL) datasets and uses webcam input for gesture recognition. This paper outlines the design, methodology, and implementation of the system, discusses the challenges in sign language translation (SLT), and highlights possible improvements using transformer-based approaches.**

***Index Terms-* Sign Language Translation, Artificial Intelligence, Deep Learning, CNN, Computer Vision.**

I. INTRODUCTION

Communication is a fundamental aspect of human interaction. For deaf and hard-of-hearing (DHH) individuals, sign language serves as the primary means of communication. However, the lack of widespread understanding of sign language among non-DHH people creates a significant barrier, leading to social isolation and communication challenges for the DHH community. Sign Language Translation (SLT) aims to bridge this gap by converting visual gestures into text or speech, enabling smoother interaction between DHH and non-DHH individuals. This project presents an AI-powered sign language translator that recognizes hand gestures in real-time using a webcam and translates them into corresponding textual output. By employing a CNN model trained on an ASL dataset, the system identifies individual letters and outputs their meaning using Python, OpenCV, and TensorFlow.

II. LITERATURE REVIEW

Previous approaches to sign language translation ranged from expensive sensor gloves to vision-based systems using deep learning. Recent works have shown that Convolutional Neural Networks (CNNs) can effectively recognize static ASL alphabet gestures with high accuracy. Some studies also explore transformer models for sentence-level translation, but these are often computationally intensive. Our project focuses on a lightweight, real-time CNN-based system using webcam input to provide an affordable solution for daily communication.

III. METHODOLOGY

The system workflow includes the following components: image capture using a webcam, image preprocessing (grayscale conversion, normalization, and resizing), gesture classification using a pre-trained CNN model, and output display as text. Python and TensorFlow were used for model training, while OpenCV handled real-time video capture and preprocessing.

The CNN architecture includes convolutional, pooling, and fully connected layers with ReLU and Softmax activations for classification of ASL alphabets (A–Z).

IV. IMPLEMENTATION

The AI-powered Sign Language Translator was developed using Python with key libraries including OpenCV for image capture and preprocessing, and TensorFlow/Keras for building and training the CNN model. The dataset used was the publicly available American Sign Language (ASL) Alphabet dataset, split into training and validation sets. The model achieved approximately 92% accuracy on validation data, confirming its reliability for static gesture recognition.

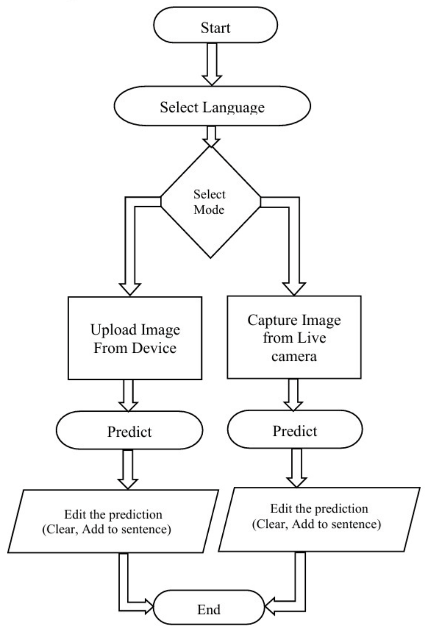
V. RESULTS

The trained CNN model achieved 92% accuracy on validation data. Real-time webcam testing demonstrated accurate recognition of most static gestures with minimal delay. However, lighting conditions and hand orientation sometimes caused misclassification. Future improvements may include transformer-based temporal models, dynamic gesture recognition, and expanded datasets for full-sentence translation.

VI. DIAGRAM

1.FlowChart
2.Data Flow Diagram
3. Use Case Diagram
4. ER Diagram
We represent the both type of diagram properly for the Ai Sign Language Translator.

1.FlowChart


Start
Select Language
Select Mode
Upload Image From Device
Capture Image from Live camera
Predict
Predict
Edit the prediction (Clear, Add to sentence)
Edit the prediction (Clear, Add to sentence)
End

Below is a clear and complete explanation of the Flowchart Diagram for your AI Sign Language Translator System.

every step and both branches (uploading an image or capturing live video).

Explanation of the Use Case / Flowchart Diagram
The diagram shows the user interaction flow in an AI Sign Language Translator application. It explains how the user selects the language, chooses how to input the sign, and receives the translation.

1. Start
The process begins when the user opens the AI Sign Language Translator application.

2. Select Language
The user is asked to choose a preferred output language—
Example:

- English
- Hindi
- Telugu
- Tamil
- Any supported language

This ensures that the predicted sign gesture will be translated into the selected language.

3. Select Mode (Decision Point)
A decision diamond indicates that the system now asks the user to pick one of two modes:

Mode Options:
1. Upload Image From Device
2. Capture Image From Live Camera
This decision splits the flow into two parallel paths.
4A. Upload Image From Device (Left Branch)
If the user chooses to upload an image:
Steps:

- User selects an image containing a hand sign from their phone or computer.
- The system prepares this image for prediction.

Then:
The system moves to the Predict step.

4B. Capture Image from Live Camera (Right Branch)
If the user chooses the live camera option:
Steps:

- The system activates the live camera.

- The user performs a sign gesture in front of the camera.
- A frame is captured for recognition.

Then:
The system moves to the Predict step.

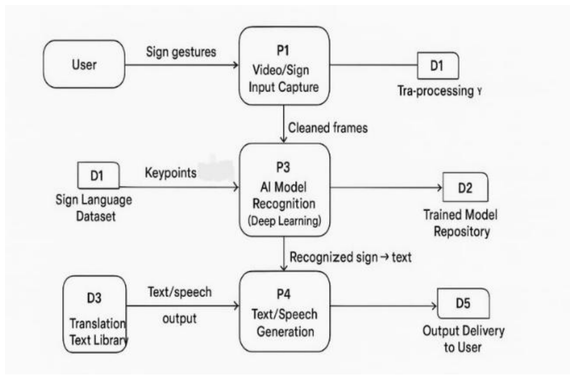
5. Predict
In both paths, the system now processes the image using the trained AI model.

What happens here:
- Hand keypoints are extracted.
- The model identifies the sign.
- The system predicts the corresponding text meaning.

Example:
✋ → "Hello"
🤟 → "I Love English"
👌 → "OK"

6. Edit the Prediction (Clear or Add to Sentence)
After prediction, the user has two options:
Edit Options:
1. Clear – Delete the prediction and try again.
2. Add to Sentence – Append the predicted word to build a full sentence.
Example sentence building:
"I" + "Love" + "English"
This step allows the user to create meaningful sentences from multiple signs.
Both branches (upload or live capture) reach this same editing step.
7. End
Once the user has completed editing or building a sentence, the process ends.

2. Data Flow Diagram


User
Sign gestures
P1 Video/Sign Input Capture
D1
Tra-processing
Cleaned frames
D1 Sign Language Dataset
Keypoints
P3 AI Model Recognition (Deep Learning)
D2 Trained Model Repository
Recognized sign → text
D3 Translation Text Library
Text/speech output
P4 Text/Speech Generation
D5 Output Delivery to User

Here is a clear, step-by-step explanation of the AI Sign Language Translator Data Flow Diagram. The Diagram describe every block (P1–P4, D1–D5) and how data moves through the system.
AI Sign Language Translator – Data Flow Explanation :→
The Diagram shows how an AI system captures sign gestures from a user, processes them with deep learning, and converts them into text or speech. The flow is divided into Processes (P) and Data Stores (D).

1. User → P1: Video/Sign Input Capture
- The process starts with the User performing sign gestures.
- These gestures are captured in real-time using:
- A camera
- A mobile device
- A webcam

P1: Video/Sign Input Capture
This module:
- Records video frames of hand and body movements.
- Cleans the frames (removes noise, adjusts lighting, etc.).
- Prepares the visual input for further processing.

Output of P1: Cleaned video frames.

2. P1 → P3 (Processing continues)
After capture, the cleaned frames are sent to the AI model for recognition.

3. D1 → P3: Sign Language Dataset
D1: Sign Language Dataset
This contains:
- Labeled sign samples
- Keypoints (hand skeleton points)

- Training data used for recognizing gestures

P3 uses this dataset to compare input signs with known patterns.

4. P3: AI Model Recognition (Deep Learning)
This is the heart of the system.
P3 Responsibilities:

- Takes cleaned frames.
- Extracts keypoints (hand, face, body joints).
- Uses Deep Learning (CNN, RNN, LSTM, or Transformers) to:
- Recognize each sign
- Map it to a corresponding meaning

Output of P3: Recognized sign → converted to text meaning.
Also Connected:
D2: Trained Model Repository

- Stores trained deep learning model.
- P3 loads this model to interpret signs.

5. P3 → P4: Recognized text generation
After recognizing the sign, the system sends the interpreted text to the next process.

6. D3 → P4: Translation Text Library
D3: Translation Library
This includes:

- Natural language phrases
- Grammar rules
- Speech synthesis resources

Helps convert recognized signs into:

- Proper readable text
- Natural speech output

7. P4: Text/Speech Generation
This module generates the final communication output.

P4 Responsibilities:

- Converts recognized text into readable sentences.
- Optionally converts text to speech using TTS (Text-to-Speech).
- Ensures the final output is user-friendly.

Output of P4: Text or speech message.

8. P4 → D5: Output Delivery

D5: Output Delivery to User
This includes:

- Mobile device screen
- Speaker output
- Application display
- Notifications

Final message is delivered to the user as:

- Text (written message)
- Audio (spoken sentence)

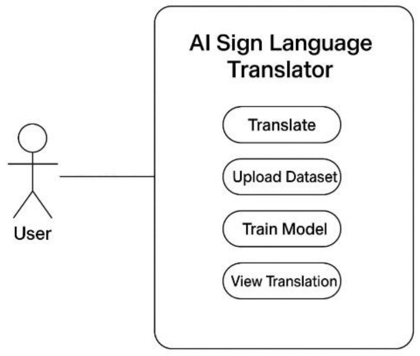
Summary (Simple View) :→

1. User performs sign.
2. P1 captures and cleans video.
3. P3 analyzes gestures with deep learning.
4. P4 converts recognized gestures into text/speech.
5. D5 delivers final output to user.

Data Stores support:

- D1: Sign dataset
- D2: Trained deep learning model
- D3: Language and translation library
- D5: Output handling

3.Use Case Diagram



Explanation of the Use Case Diagram for the AI Sign Language Translator
The Use Case Diagram illustrates the major interactions between the User and the AI Sign Language Translator System. It identifies the core functionalities provided by the system and shows how an external user interacts with each feature. This diagram is essential for understanding the functional requirements of the application.

1. Actor: User
The User represents anyone who operates the AI Sign Language Translator. This may include:

- People communicating using sign language,

- Hearing individuals who want to understand signs,
- Researchers or administrators managing datasets and model training.

The actor interacts with the system through all four main use cases.

2. Use Case: Translate
This use case allows the user to convert real-time sign language gestures (captured via camera or uploaded video) into readable text or speech.
Functional steps may include:
- Capturing the gesture input
- Preprocessing the video frames
- Running the AI model for prediction
- Displaying the recognized text

This is the primary feature of the system.

3. Use Case: Upload Dataset
This functionality is typically used by researchers, developers, or system administrators.
It enables the user to upload a new dataset consisting of sign language images or video samples for model improvement.

The purpose of this use case is:
- Expanding or updating the dataset
- Improving model accuracy
- Supporting diverse sign patterns

4. Use Case: Train Model
After uploading or updating the dataset, the user can trigger the Train Model use case.
This involves:
- Data preprocessing
- Model training with machine learning or deep learning techniques
- Generating a trained model for real-time translation

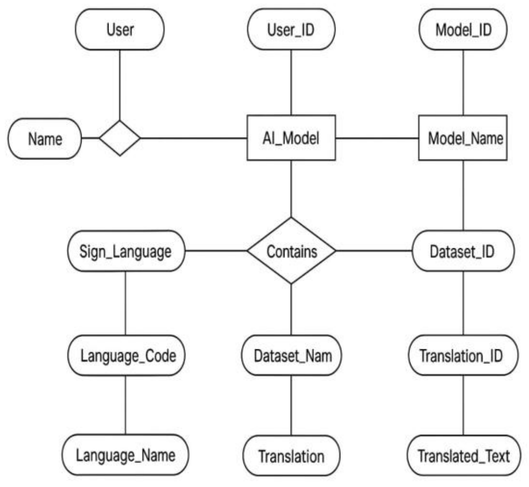
This use case ensures the system stays accurate and up-to-date.

5. Use Case: View Translation
This use case allows the user to view the output of the translation process.
The system displays the translated text, which may also be converted into speech through a text-to-speech module.
Users can:
- Review the recognized signs
- Verify the accuracy
- Use the output for communication

Overall Functionality
The diagram clearly shows that the User is the central actor who interacts with all system features. The system provides four major capabilities:
1. Translate sign language
2. Upload sign datasets
3. Train or retrain AI models
4. View translation results

These use cases collectively define the functional architecture of the AI Sign Language Translator and ensure efficient gesture recognition, model improvement, and user interaction.

4.ER Diagram



Explanation of the ER Diagram for AI Sign Language Translator
The Entity–Relationship (ER) Diagram represents the database structure of the AI Sign Language Translator system. It shows the key entities, their attributes, and how they are logically related. This diagram ensures that the system has an efficient and well-organized data model to support translation, dataset management, and AI model training.

1. Entity: User
This entity stores information about every person who interacts with the system.

Attributes:

- User_ID – A unique identifier for each user.
- Name – Stores the user's name.

Purpose:

The User entity helps track:

- Who uploads datasets
- Who trains or uses models
- System-level personalization

The User is connected to the AI_Model entity, meaning each user manages or utilizes one or more AI models.

2. Entity: AI_Model
This entity represents the AI models used for sign language translation.
Attributes:

- Model_ID – Unique ID assigned to each AI model.
- Model_Name – Name or type of the model (e.g., CNN-Model, MobileNet-Model).

Purpose:
The AI_Model stores details of the machine-learning or deep-learning models created, trained, or used. It connects:

- With User (user who uses/trains the model)
- With Dataset (the dataset used for training)

3. Relationship: Contains
This is the central relationship connecting AI_Model, Sign_Language, and Dataset.
Meaning:

The *Contains* relationship indicates that:

- Every AI Model contains or uses
- A Dataset, which is
- Related to a specific Sign Language

Thus, an AI Model is trained on a Dataset that belongs to a particular Sign Language.
This helps ensure that models and datasets are organized language-wise (example: ISL, ASL, BSL).

4. Entity: Sign_Language
This entity represents the different sign languages supported by the system.
Attributes:

- Language_Code – A unique language identifier.
- Language_Name – Example: "American Sign Language (ASL)", "Indian Sign Language (ISL)".

Purpose:
To categorize datasets and models based on the language they represent.

5. Entity: Dataset
This entity stores information about the datasets used for training AI models.
Attributes:

- Dataset_ID – A unique identifier for each dataset.
- Dataset_Name – Name of the dataset (e.g., "ASL_HandGestures_V1").

Purpose:
Each dataset belongs to a specific sign language and is used to train AI models.
Organized dataset storage ensures proper training and updating of the system.

6. Entity: Translation
This entity stores the translations generated by the AI model.
Attributes:

- Translation_ID – Unique identifier for each translation instance.
- Translated_Text – Output text generated after interpreting sign gestures.

Purpose:
To store and retrieve past translation results, which can be useful for:

- Displaying results to the user
- Evaluation and accuracy testing
- Learning analytics

This entity is directly linked to the *Contains* relationship through the datasets used for translation.

Overall ER Structure Summary
This ER model clearly defines how the system manages essential elements:

- Users create or use AI Models
- AI Models are trained using Datasets
- Datasets belong to a particular Sign Language
- Translations store the recognized text output

The diagram provides a strong and normalized database structure for efficient handling of sign language data, training workflows, and translation outputs.

VII. CONCLUSION

This paper presented an AI-based Sign Language Translator using a CNN model to recognize ASL alphabet gestures. The proposed system provides a low-cost, real-time, and accessible tool for bridging communication between deaf and hearing individuals. Future work will explore transformer architectures, dynamic sign recognition, and enhanced user interfaces for everyday usability.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” arXiv:1312.6114, 2013.

[3] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987.

[4] TensorFlow Documentation, https://www.tensorflow.org/.

[5] OpenCV Documentation, https://opencv.org/.