

# Hand2Help: A Bidirectional Sign Language Chatbot

Parth Wagholikar<sup>1</sup>, Riya Sutar<sup>2</sup>, Yogesh Salunke<sup>3</sup>, Aniket Nagdare<sup>4</sup>, Dr. Manish Sharma<sup>5</sup>,

Ms. Neeta Katariya<sup>6</sup>

<sup>1,2,3,4</sup> Students, Dept. of Artificial intelligence & Data Science, DYPCOE, Pune, India

<sup>5</sup> Professor, Dept. of Artificial intelligence & Data Science, DYPCOE, Pune, India

<sup>6</sup> Assistant Professor, Dept. of Artificial intelligence & Data Science, DYPCOE, Pune, India

\*\*\*

**Abstract**-Hand2Help is an AI-supported system of communicative assistance which will allow deaf and hard-of-hearing people to communicate with non-signers through intelligent conversion of textual to sign language format in real-time. The foundation of Hand2Help consists of a locally-deployed large language model (LLM). The LLM uses Ollama to create short, contextually aware sentences or responses to a user's request; those sentences/responses are then converted into sign language using a structured video-based mapping of each user's words against a database of sign language words. The mapping process includes both "word" mapping and "letter" mapping. Hand2Help is implemented entirely in Python, utilizing both Flask and JavaScript. As an offline application, it provides privacy for users, as there is no data sent to an outside server, and it does not require any external API calls to provide service. The outcomes from Hand2Help indicate that the generation of conversational responses and the accurate visualization of sign signal using sequential video presentations were achieved, providing a scalable starting point for the development of a future successful bi-directional communication system.

**Key Words:** Sign Language Translation, Large Language Model (LLM), Ollama, Chatbot, Assistive Technology, Human-Computer Interaction, Offline AI, Video-Based Sign Rendering

## 1. INTRODUCTION

Communication is a vital part of life for everyone; however, it is particularly difficult for those who have hearing loss to communicate with people who do not use sign language. There are traditional forms of communicating with others, including using human interpreters and using static translation tools; unfortunately, the availability and flexibility of these types of services have limitations. As such, there is a need for intelligent assistive systems that will allow for seamless and efficient communication. With much recent advancement in Artificial Intelligence (AI), specifically through the use of Large Language Models (LLMs), AI systems can now create context-sensitive, Relevant responses. Unfortunately, most current sign language implementations focus on one of two things: static translations or gesture recognition; therefore, they do not contain conversational

intelligence and real-time adaptive capabilities.

This paper introduces Hand2Help, an AI-powered assistive communication system that utilizes a conversational chatbot interface to facilitate text-to-sign communications. Using locally deployed LLMs through Ollama, the system allows for the generation of concise responses which are translated into sign language through structured video-based mappings with word-level matching and letter-level fallbacks. The entire system is written in Python, Flask, and JavaScript and is fully offline, thus protecting user data and providing reliability. The work described here establishes a pragmatic approach for developing scalable solutions for future bidirectional communication systems.

## 2. LITERATURE REVIEW

Recent improvements in assistive communication tools have resulted in many AI-based solutions for translating sign language. A virtual assistant that uses conversational AI along with sign language translation has been suggested to make communication better by combining natural language understanding with hand movements [1]. Although this method shows promise, it depends on cloud services and complex processes that handle different types of input.

There are a number of researchers researching different types of bi-directional sign language translation systems based on deep learning technologies. One example of a bi-directional sign language translation system utilized with deep learning is Indian Sign Language (ISL), which uses Convolutional Neural Networks and Transformer-based models for the conversion of sign language to text and vice versa in real time [2]. While these systems can provide good accuracy when converting between sign language and text, they tend to require large amounts of data and computing power to operate making them difficult to deploy on devices that have limited resources available.

Ishara-Verse improves the translation quality of sign language systems by using the rule-based processing and deep learning to integrate context and use multiple languages to support translation [3]. Even though Ishara-Verse can effectively handle many forms of inputs, it

requires an internet connection to function and has a relatively complicated setup.

Another bi-directional sign language translation system is Sign Chat AI, which uses technologies such as generative AI and natural language processing to provide communication by way of sign language can be used for conversational quality [4]. Although Sign Chat AI helps to improve the quality of conversation between two persons, it requires a high level of computing resources as well as relying on external services for function. Conversely, Hand2Help was created using a lightweight offline model to limit these dependencies while still allowing the users to maintain meaningful communication.

It works offline by using a local version of a large language model along with a structured way to convert text into sign language videos. This reduces the need for external services while still allowing for clear and context-aware communication.

### 3. METHODOLOGY

The Hand2Help system aims to create intelligent text-to-sign communication via the combination of a conversational-based Chatbot with a structured method of rendering Signed Languages through a multi-part method of processing input, generating a response using a locally installed Large Language Model (LLM), followed by converting the text response into signed video.

#### 3.1 System Overview

Users interact with a web-based interface that allows them to input text. This text is processed by a backend server implemented with Flask software, which communicates with the LLM and creates the proper response. Once the response is created, it is then sent to another module that converts the text into the appropriate sign language video sequence.

#### 3.2 Conversational Response Generation By LLM

To create contextualized and meaningful responses, the Large Language Model that we've built on Ollama is based on a local deployment. This enables the chatbot to interpret a user's input and create a response that is both concise and optimally formed to be translated into sign language. Different model parameters (e.g., Temperature) are set to ensure produced responses remain stable and consistent. Conversation history is recorded with the system throughout the session so that the chatbot may have context for all user interactions.

#### 3.3 Text Preprocessing and Tokenization

Generating text into signing first involves preprocessing the text. That is to say, the text needs to be cleaned and separated into single words to allow maximum efficiency when matching each word with the corresponding sign.

The length of the response is also controlled in order to create more seamless and compatible multi-dimensional signs.

#### 3.4 Sign Language Video Mapping

The process of taking text and converting it into sign language is done through an organized mapping process. The system will initially look up each word found in the source text by checking to see if there is a sign video available that matches that word. If no sign video is available for the word, the system will take the word and break it down into single letters and provide sign videos based upon those letters. This ensures that all input will be converted to a sign video.

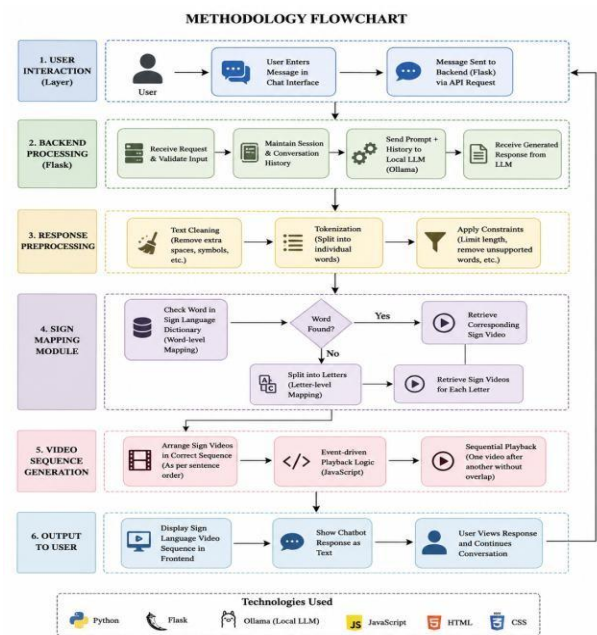


Fig no.1 Methodology Flowchart

#### 3.5 Sequential Video Playback

In order for the entire sentence to be presented accurately, the different sign language videos that were mapped out are shown one Right after another. In order for each video to begin Playing only after the video before It has finished, there is an event-driven, Javascript-Based mechanism for the Front end. Therefore, as the viewing of video progresses, visual continuity will not be interrupted because of multiple passings or overlapping between two adjacent sequential pictures will avoid disruption during playback from either picture or video to another.

#### 3.6 Offline System Design

The whole system has been built to run offline with an LLM deployed locally through Ollama. By doing so, it eliminates reliance on outside APIs, improves data privacy and reduces costs. The offline architecture also helps to improve the reliability of the system and is

therefore well suited for deployment in areas that require minimal resources.

#### 4. SYSTEM ARCHITECTURE

Hand2Help is a modular and scalable system architecture that includes a graphical user interface, backend processing unit, Large Language Models (LLM) for processing, and sign language rendering modules. The architecture enables data to move smoothly from the user's data entry through to rendering as sign language, and vice versa, while providing an efficient and dependable means of accomplishing this task.

##### 4.1 Overall Architecture

The software application uses a Client-server model in which the user interface (UI) is used to send queries from the front end to the back end server for processing. The front end is responsible for capturing user input and providing responses from the chatbot (along with sign language videos). The back end is implemented using Flask (Python-based web server framework) and acts as the action/traffic controller by routing requests appropriately, communicating with the LLM and processing responses.

##### 4.2 Frontend Module

We built an interactive chat interface with HTML, CSS, and JavaScript that allows users to send written messages to the chatbot from the text box and receive replies from the chatbot on the screen. The front-end has an event-driven way for users to play sign language video clips in a sequential manner. Additional features like saving chat history, deleting conversations, and updating the interactive front-end automatically, enhance user interaction.

##### 4.3 Backend Module

The Flask-based backend module is used to process requests made via APIs by users, and to communicate between various application components. The backend module accepts user input, tracks conversations during session activity, and interacts with the local LLM by making API requests for the generation of responses. In addition, the backend module pre-processes the generated responses prior to creating sign language sequences.

##### 4.4 LLM Integration Layer

The system integrates a locally deployed Large Language Model using Ollama. This layer is responsible for generating context-aware responses based on user input. By operating locally, it ensures data privacy and reduces latency. The model is configured with controlled parameters to produce short and deterministic responses suitable for sign language conversion.

##### 4.5 Sign Language Rendering Module

Using video to translate written words to sign language is accomplished by finding a pre-recorded sign video from the datasets that are linked to each word. If there is no recorded sign video that matches a written word directly, the system will search using an alternate method of converting the written word into its individual letters and locating a sign video associated with each letter. The completed response will be produced as a series of video files in the order they were produced.

##### 4.6 Data Flow

User inputs are received on the front end and transmitted through the backend server. The backend then relays the input to its LLM, receives the generated output from the LLM, and maps that output to signs (through the video sequence) before returning both the video sequence and the mapped signs to the front-end for sequential display to the user. This entire pipeline is designed for efficient, real-time communication between all components.

##### 4.7 System Advantages

The use of a modular architecture allows an easy path for scale and future enhancements. Integration of the offline LLM contributes to user privacy by providing secure methods for storing and processing personal information and minimizing reliance on third parties. The separation of front-end and back-end components has improved maintainability of the system as well as overall performance.

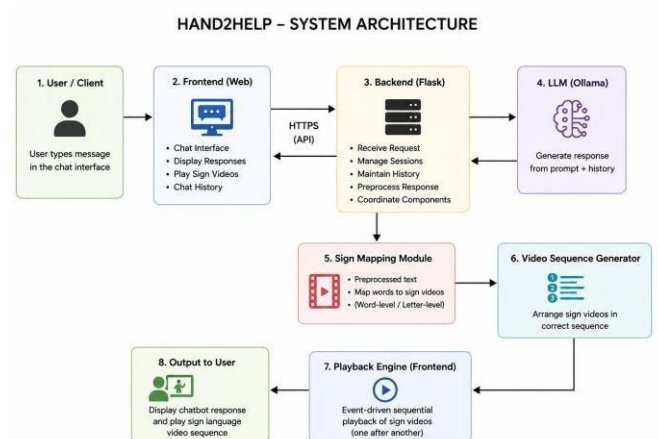


Fig no.2 System Architecture

#### 5. RESULTS AND DISCUSSION

The Hand2Help system has been developed and demonstrated as a functional prototype for communication between text and sign using a conversational chatbot. This prototype has effectively integrated a locally hosted Large

Language Model with a structured sign language rendering pipeline. A number of test cases were run to evaluate the accuracy of generated responses, sign mappings, and playback performance.

### 5.1 Chatbot Response Evaluation

Each type of user query was used to test the chatbot, including general questions, short requests, and conversational prompts. The LLM produced short but contextualised responses that could all be used to convert into sign language. By applying deterministic parameters, the output was consistent for each repeated input, which enabled the chatbot to provide reliable means to assist communication for people needing assistance.

### 5.2 Sign Language Mapping Accuracy

The evaluation of the text to sign conversion module was done according to whether the correct corresponding signs were produced for each message submitted. The system achieved very high coverage through a combination of matching at both the word level and a letter level fallback. The proportion of times a specific word would be available within the data and the fallback mechanism would also represent and communicate the total message using signs based on the individual letters of the words.

### 5.3 Sequential Video Playback Performance

The video player was tested and found to play smoothly with no skips or overlaps when playing back video of sign language sequences. In addition, an event-driven mechanism has been employed to eliminate any skips or overlap of videos with repeated letters (e.g., playing "a", then playing "a" again as one video signs) to allow for an uninterrupted flow between videos and a clear and understandable visual output.

### 5.4 System Performance and Efficiency

The offline operation of the system had no dependencies on external APIs. The response times were stable and experienced only small amounts of latency due to local LLM inference (i.e. very little). The lightweight architecture permitted the system to function efficiently on standard computing devices that did not need cutting-edge technology.

## 5.5 DISCUSSION

According to the findings from this study, there is a successful connection made between written communication and the representation of sign language. Through the use of AI for conversation, the quality of communication will be superior to what is provided by other traditional methods (e.g., Static Translation

Devices). At this time, however, the system is limited in its applications due to the lack of available sign video datasets as well as the inability to provide sign language-specific grammar restructuring for translated communications. In light of these limitations, the proposed system offers a good starting point for continual improvements (e.g., Gesture Recognition Systems & Linguistic Model Improvements).

## 5. RESULTS AND DISCUSSION

This article describes an AI-based assistive communication system called Hand2Help that was created in order to improve communication between Deaf/hard of hearing people who use sign language and people who don't use sign language through intelligent translations from text to sign language. By using a local deployment of a Large Language Model and adding a mechanism for structured sign language video mapping, the system generates responses that are both context aware and convert those responses into meaningful representations in sign language.

Through this combination of conversational AI and video-based sign rendering, the approach demonstrates how much more accessible and efficient communication can be when you have a reliable tool that operates offline, maintaining user privacy, reducing dependence on external services, and being able to function in resource constrained environments. The combination of word-level mapping and letter-level fallback mapping guarantees subject-matter coverage with reliable outputs across a variety of inputs.

Although the current version of the system is limited to text-to-sign communication, it establishes an excellent starting point for future development of fully bi-directional systems. Future work will include the incorporation of real-time gesture recognition capability, improvements in sign grammar, and increasing the volume of data for better accuracy and naturalness in their communication. Ultimately, Hand2Help is a practical and scalable step toward developing inclusive communication systems that employ AI.

## 6. REFERENCES

- [1] V. Vijay et al., "A Virtual Assistant with Real-Time Sign-to-Sign Language Translation for Users with Disabilities," in ICICNIS, 2024.
- [2] A. R. P. Adithyaraaj et al., "Indian Sign Language (ISL) Translator: AI-Powered Bidirectional Translation System," IJRASET, vol. 13, no. 3, 2025.
- [3] K. Kashish et al., "Ishara Verse: A Bidirectional, Context-Aware, Multilingual Sign Language System," in ISACC, 2025.
- [4] P. Venkadesh et al., "SignChatAI: Generative AI for Deaf-and-Mute Community using Indian Sign Language," in ICCIRT, 2024.

- [5] T. Starner, J. Weaver, and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 12, 1998.
- [6] O. Koller, H. Ney, and R. Bowden, "Deep Learning of Mouth Shapes for Sign Language," in ICCV Workshops, 2015.
- [7] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in IEEE ICASSP, 2013.
- [8] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in NAACL, 2019.
- [9] T. Brown et al., "Language Models are Few-Shot Learners," in NeurIPS, 2020.
- [10] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in OSDI, 2016