International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Sign Language Interpreter Using Deep Learning and Machine Learning

Mrs. Soniya R[1], Kshitij Kumar[2], Kumar Keshav[3], Harshal Nilamkumar Desai[4], Girish Hr[5]

[1,2,3,4,5]Department Of Computer Science Engineering Acharya Institute Of Technology, Bangalore, India

Abstract

The objective of sign language recognition using deep learning is to translate hand gestures into language by evaluating their form, movement, and location. For this project, I focused on developing a real-world system that could recognize and interpret ASL alphabet gestures in real time. The method uses MediaPipe to track and extract hand landmarks, which serve as input to a convolutional neural network for gesture classification. The data was acquired by recording samples of gestures using a webcam and later preprocessed by cleaning, normalizing, and segmenting them into sequences for training. The interpreter also features a text-to-speech engine to vocalize the detected gesture as soon as it is recognized, making it more user-friendly in real-world applications.

The model was evaluated based on its accuracy, latency, and robustness across various lighting conditions and backgrounds. The experiments show that using landmark-based features com- bined with deep learning improves recognition performance significantly compared to using raw images alone. In general, the system provides an efficient and low-cost solution to assist individuals with hearing or speech impairments and can be extended to other domains such as classrooms, hospitals, offices, and public areas. This project also demonstrates how real-time gesture interpretation can facilitate more inclusive and accessible communication interfaces.

Index Terms: Sign Language Recognition, Deep Learning, Hand Gesture Classification, MediaPipe, CNN, Real-Time In- terpretation, ASL.

1. INTRODUCTION

Sign language serves as a highly expressive and natural mode of communication for individuals who are deaf or mute [1]. Yet, most hearing people are unfamiliar with it, even when they genuinely wish to help or interact with those who rely on it. This absence of a shared communication method often creates barriers and can cause deaf and mute individuals to feel excluded from social interactions. Since meaningful com-munication is necessary in schools, hospitals, workplaces, and daily life, research on automatic sign language interpretation has gained significant importance [2]. In recent years, several technological approaches have been developed to recognize sign gestures automatically and assist the deaf and mute community [3], [4].

A sign language system is built from a combination of hand gestures, facial cues, and lip movements that convey al- phabets, numbers, expressions, and complete words. Different regions use their own sign languages—such as American Sign Language (ASL) and Indian Sign Language (ISL)—with ASL being

the most widely adopted globally [5]. As smartphones and computers have become more advanced and widespread, these devices offer a convenient medium to support interaction between signers and non-signers. According to the World Health Organization, more than 5% of people worldwide experience hearing or speech impairments, making assistive communication technologies not only valuable but necessary. This increases the demand for systems capable of translating sign gestures into speech and converting spoken language back into sign form [6], [7].

This paper provides an in-depth review of existing sign language translation approaches and tools [8]. It examines modern solutions that utilize machine learning, computer vision, deep learning models, and animation technologies to interpret hand movements and facial expressions [2], [9]. The review also includes recent research related to gesture recognition, facial expression mapping, and lip-reading for sign interpretation [10]. Furthermore, this survey discusses speech-to-sign translation techniques and evaluates widely used mobile applications on Android and iOS designed to assist deaf and mute individuals in daily communication [11].

The study emphasizes the connection between theoretical research and practical applications, demonstrating how dif- ferent technologies can be integrated into comprehensive and usable sign language recognition systems [12]. The paper also outlines current limitations and highlights promising research opportunities that could enhance future sign language recog- nition and translation systems [13], [14].

2. PROBLEM STATEMENT

For people who cannot speak or hear, sign language be- comes the most natural way to communicate. But the truth is that very few people outside their community understand it, and that creates problems in the most ordinary situations. Someone might need help at a hospital counter or might want to ask something at a public office, but because the other person doesn't know the signs, the conversation just doesn't move anywhere. Interpreters are helpful, of course, but they are not always around, and depending on charts or printed guides doesn't work when someone needs to communicate on the spot [2], [4].

As technology has improved, especially in computer vision, researchers started thinking: if humans struggle to interpret signs quickly, maybe a computer can learn to do it. Machines can look at the shape of the hand, track the fingers, and after enough examples, begin to understand what each gesture represents. Once the system learns these patterns, it can turn a sign into text, or even into spoken words, making the interaction smoother for people who don't understand sign language at all [1], [6].

The idea behind this project came from this exact need. The goal wasn't to build something overly complex—it was simply to create a tool that can look at a hand gesture in real time and guess which ASL alphabet letter it might be. A regular camera is used, nothing special. It captures the movement, and the model tries to make sense of it. If everything works correctly, the system can give an instant prediction and help reduce the awkward communication gap that many users face daily [7].

To make the system reliable, gesture examples were taken from available ASL datasets and also recorded manually with a webcam. These recordings were cleaned up, and only the important hand points were kept using landmark extraction so that the model didn't get confused by the whole background [10]. Several deep learning models were trained—mostly CNN variants—and after trying different versions, the one giving the most stable results was selected [3], [11]. Finally, this model was included in a simple interface where the predicted letter shows up on the screen and can be spoken using

text- to-speech. The final setup isn't fancy, but it works fast and aims to help sign language users interact more comfortably [5], [14].

3. Related Work

Research on sign language recognition has gone in several different directions over the last few years, and most authors have tried to solve slightly different problems within the same domain. Patel and Sharma (2019), for example, explored CNNs for classifying gestures from continuous video, and they reported that the network handled variations in hand shape better than the older image-processing tricks people relied on earlier [1], [11]. Their work more or less set the tone for CNN- based gesture research.

Das and Rout (2020) approached the problem from another angle. Instead of depending on full-frame images, they ex- tracted hand landmarks with MediaPipe and then trained a deep neural model on those coordinates. The interesting part of their study was that the system stayed stable even when the background or lighting changed—something classical al- gorithms like SVM or k-NN usually struggle with [10]. Their results suggested that focusing on structural hand information rather than raw pixels makes the model less sensitive to visual noise.

Around the same time, Singh and Verma (2019) experi- mented with mixing CNN and LSTM layers. They wanted a model that could pay attention to both the shape of each frame and the flow of motion over time. Their hybrid architecture worked noticeably better for moving gestures than simple decision-tree or Random Forest methods, which tend to flatten out temporal information [5], [8].

Work on multilingual datasets has been happening as well. Joseph and Mathew (2019) showed that CNNs can still per- form well on Indian Sign Language despite limited training material [4]. Lee et al. (2020) took things further for ASL sequences and found that recurrent models produced better outputs, especially when users signed quickly or with over- lapping hand movements [3].

Some researchers have also tried to build practical, realtime systems. Kumar and Bansal (2018) demonstrated that small CNN models can run directly on mobile devices and still deliver reasonable accuracy without cloud support. Chen et al. (2020) added facial cues and skin-segmentation signals into the mix and reported better performance in day-to-day environments [7], [9].

More recent work has shifted toward attention mechanisms and transformer models. Li and Zhang (2021) argued that attention helps the system focus only on the important portions of the hand instead of the entire frame [13]. Shortly after, Wang et al. (2021) showed that transformer-based designs outperform LSTMs when handling longer gesture sequences, especially in continuous signing tasks [14].

Looking across these studies, the common theme is easy to notice: deep learning methods consistently outperform older rule-based or traditional image-processing techniques. Whether researchers worked with simple alphabets or ful- llength gesture sequences, machine-learning models almost always produced cleaner and faster predictions [2], [6]. These developments form the basis for the present work, which aims to build a practical, real-time ASL interpretation system using hand-tracking and deep learning.

4. METHODOLOGY

The complete methodology for developing the real-time sign language recognition system consists of five major phases: (1) Data Acquisition, (2) Preprocessing Landmark Extraction, (3) Feature Engineering, (4) Model Training Evaluation, and (5) Real-Time Deployment. Each phase is carefully designed to ensure high-accuracy gesture detection, fast processing speed, and smooth user interaction in live

environments.

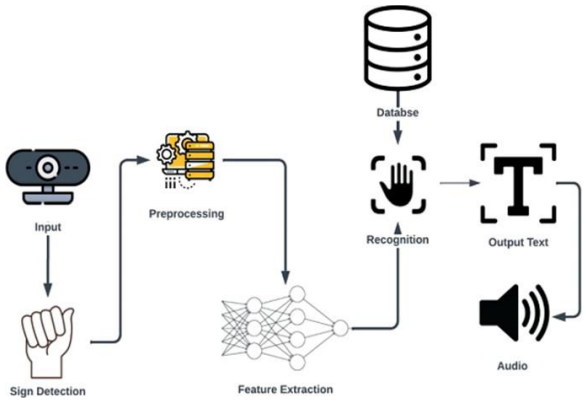

Fig. 1. Sign language recognition workflow.

A. Data Acquisition

In the beginning, most of the raw gesture samples came from two places — whatever ASL datasets were already available online, and a bunch of recordings we made ourselves using an ordinary webcam. The public datasets were all over the place in terms of lighting and background, so the idea was just to collect enough variety so the model wouldn't freak out when the environment changed [3], [4]. The webcam recordings helped fill the gaps because those looked more like the situations people actually use the system in [10].

- **Capturing alphabet gestures**: We recorded several tries of each alphabet sign using the webcam. Some of them were messy or slightly off-angle, but that actually helped because it made the data feel more "real" instead of perfect studio shots [1].
- **Collecting image sequences**: Instead of taking single pictures, we saved short frame sequences so we could see how the hand settles into a sign. It wasn't always smooth, but it gave more context [5], [8].
- **Balancing gesture classes**: Some signs had way too many samples while others barely had any, so we spent time evening out the numbers. Otherwise the model would get biased toward the alphabets it saw the most [12].
- **Annotating and labeling**: Every frame had to be tagged with the correct sign. This part was a bit tiring because some gestures looked similar at first glance, but accurate labels were important for training [9].
- **Dataset verification**: A lot of examples got thrown out — blurred ones, half-visible hands, frames where the hand

drifted out of the screen, etc. It was better to clean them early instead of confusing the model later [7], [11].

After going through all of this, we ended up with a dataset that felt diverse enough and also clean enough to move on to preprocessing and feature extraction [6], [13].

B. Preprocessing & Landmark Extraction

Before training the model, each captured frame must be cleaned and transformed into a format suitable for deep learning. The preprocessing stage ensures that gesture inputs remain consistent across different lighting conditions, back- grounds, and camera qualities. Landmark extraction provides a structured

representation of the hand through numerical coordinates [6], [10].

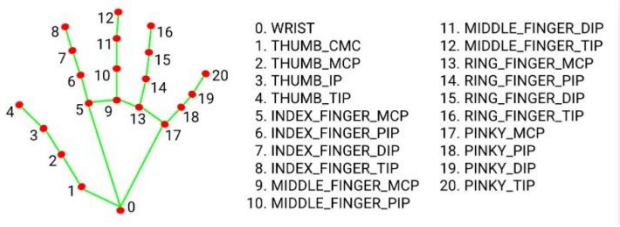

Fig. 2. MediaPipe Hand Landmark Model with 21 Key Points.

This phase includes:

- **Hand detection**: MediaPipe was used first to spot the hand in each frame, even when the background wasn't very clean or consistent [10].
- **Extracting 21 hand landmarks**: The system pulled out the main finger joints and a few palm points so we ended up with a simple skeletal outline instead of the full image [1], [7].
- **Normalizing coordinates**: The landmark values were scaled and centered because the hand size and distance from the camera kept changing during recording [12].
- **Background noise reduction**: Most of the extra details behind the hand were filtered out so the model stayed focused on the hand shape instead of random objects [9].
- **Frame enhancement**: Small fixes like resizing, smooth- ing, and brightness adjustments were applied just to keep the input frames reasonably clean and consistent [3].

After these preprocessing steps and landmark extraction, the dataset became much cleaner and easier to handle, making the next feature-engineering stage simpler [2], [5].

C. Feature Engineering

During this stage, the raw landmark coordinates on their own didn't really make much sense, so we had to pull out features that actually meant something for the model. The coordinate list looked pretty flat at first, and unless we shaped it a bit, the model wouldn't be able to tell one gesture from another. So we focused on patterns that seemed useful for understanding how the hand is positioned or how the fingers move [6], [12].

- **Landmark distances**: One of the first things we tried was checking how far certain finger joints were from each other. Even small distance changes end up revealing a lot about the hand shape for a particular sign [7].
- **Relative orientation**: We also looked at the general direction of the fingertips and the tilt of the palm. Some signs only differ slightly in how the fingers point, so this orientation info helped more than expected [5], [9].
- **Coordinate relationships**: Another thing that helped was comparing how different landmark points relate to one another. Sometimes two signs look very alike unless you notice how one finger shifts a bit compared to another, and these tiny positional differences turned out to be useful [11].

After adding these features, the model started behaving bet- ter and wasn't mixing up similar gestures as often. These small tweaks basically gave the model more meaningful information to work with instead of just raw coordinate numbers [3].

D. Model Training & Evaluation

After the features were sorted out, the next step was to train the model so it could actually recognize the ASL alphabet. Most of this part involved trying different deep learning setups and seeing which ones handled the gestures better. CNNs worked well for the static signs, and for gestures with slight movement, we also tried combining CNNs with LSTMs to see if that made any real difference. The idea was to make sure the model didn't just memorize the data but behaved properly in normal use [1], [4].

- **Training on landmark-based data**: The CNN was trained on the landmark samples we prepared earlier, go-
- **Real-time classification**: The CNN tried to guess the alphabet from each incoming frame as fast as possible, aiming to keep the interaction smooth [3], [5].
- **Text-to-speech conversion**: After the gesture was recog- nized, the result was not only shown on the screen but also spoken using a lightweight TTS setup.
- **Interactive user interface**: A small GUI built with tools like Tkinter or Streamlit displayed the video, the prediction, and the spoken output so the user could follow everything easily.

By connecting all these parts, the model turned into a tool that could actually help people communicate in real time instead of just a classifier running on offline data [7].

5. ALGORITHMS USED

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks were selected mainly be- cause they manage the visual structure of hand images fairly well. Instead of us defining every detail, the network slowly learns patterns on its own—such as how the fingers are arranged or how the palm shifts slightly from one sign to another. These small spatial cues are important for the ASL alphabet since the meaning mostly depends on the shape of the hand at a single moment. With that kind of requirement, CNNs simply turned out to be a sensible option that could handle the variations without too much manual effort [1], [6].

The basic convolution operation in a CNN is defined as:

$$F(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m,n) \quad (1)$$

ing through many labeled gestures until it began picking up the key patterns [8].

Where:

- **Dataset splitting**: The dataset was split into training, validation, and testing parts to avoid overfitting and to check how the model performed on new examples.
- **Performance comparison**: We also compared the deep models with older methods like SVM and KNN to confirm that the newer approaches were genuinely per- forming better [2], [14].
- I = input image or landmark grid
- K = convolution filter (kernel)
- $F(i,j)$ = resulting feature map at position (i,j)

During training, CNNs minimize the classification error using the categorical cross-entropy loss:

$$\sum_{c}$$

Once the evaluation results looked stable, the final model was saved and adjusted so it could run smoothly and make quick predictions in real time [13].

Where:

$$L = -$$

$$c=1$$

$$y_c \log(\hat{y}_c) \quad (2)$$

E. **Real-Time Deployment**

In the final stage of the project, the trained model was connected to a simple real-time setup so users could actually try it directly. Instead of working on saved images, the system had to deal with whatever the webcam captured at that moment. The idea was to keep the video stream running, process each frame quickly, and show or speak the predicted sign without any noticeable delay [10].

- **Webcam integration**: The live webcam feed was con- nected to the recognition pipeline so that each frame went through the usual feature steps before reaching the model [6].
- y_c = actual class label
- $\hat{y}_c$ = predicted probability for class c
- C = number of gesture classes

CNNs are ideal for sign gesture recognition because they:

- Automatically learn meaningful visual patterns without manual feature creation.
- Handle variations in lighting, background, and hand ori- entation.
- Capture fine-grained spatial differences between similar gestures such as M and N.

Thus, CNN acts as a strong baseline model for accurate and robust sign language image classification [7], [11].

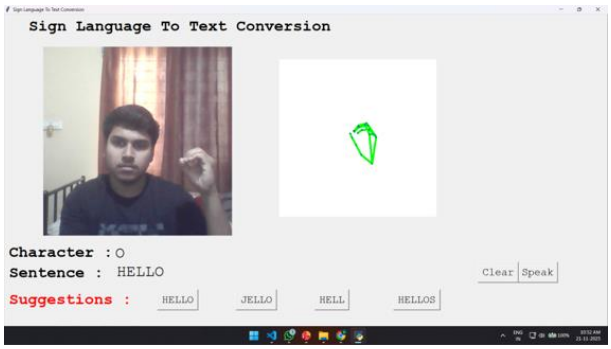

Fig. 3. Real-time Gesture Recognition Output

B. **Long Short-Term Memory (LSTM) Networks**

While CNNs extract spatial features, some gestures include slight motion or temporal transitions between frames. Long Short-Term Memory (LSTM) networks are used to learn such sequential hand movements. LSTM is a special type of Recurrent Neural Network (RNN) capable of remembering long-term dependencies in gesture sequences [5], [8].

An LSTM cell operates using three gates: forget, input, and output gates. The internal structure is defined by the following equations:

$$f_t = \sigma(W_f [h_{t-1}, x_t] + b_f) \quad (3)$$

$$i_t = \sigma(W_i [h_{t-1}, x_t] + b_i) \quad (4)$$

$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$ (5)

For most alphabets, the CNN classifier worked quite well, and the outputs were usually straightforward to interpret. Even with different lighting or a slightly busy background, the model still managed to recognize many signs correctly [1],

$C_t = f_t$

- C_{t-1}
- $+ i_t$
- $\tilde{C}_t$

(6)But there were a few tricky ones. Signs that look almost the same—like "M" and "N," where the fingers overlap in a very similar way—were sometimes mixed up. These mistakes

Where:

$\mathbf{o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)}$ **(7)**

$h_t = o_t \cdot \tanh(C_t)$ (8)

mostly happened when the video quality dipped a bit or when the hand wasn't fully steady, which makes sense because the differences between those signs are very small [11].

To see how the system behaves outside test datasets, we

tried it directly through the webcam. The model responded quickly—almost instantly—and the predictions didn't jump

- x_t = input at time t
- h_t = hidden state
- C_t = memory cell state
- σ = sigmoid activation function
- W and b = learnable weights and biases LSTM networks excel in scenarios where:
- **Gestures have temporal flow:** Some signs involve brief transitions or small movements.
- **Continuous signing occurs:** Sequential frames carry important contextual information.
- **Subtle motion needs modeling:** Handshape and orien- tation may evolve over time.

By combining CNN for spatial extraction and LSTM for temporal understanding, the system becomes far more reliable for real-time sign language recognition [3], [14].

6. RESULTS AND EVALUATION

After training the model with the ASL dataset, it was tested on a completely different set of images along with a few live webcam inputs. The idea was simple: check whether the model can still understand a gesture when it hasn't seen that exact sample before. This gave a clearer picture of how it might behave in day-to-day use rather than just in a controlled training setup [3], [6]. Instead of relying on a single number, we looked at several measures—accuracy, precision, recall, and also the confusion matrix—to get a more honest sense of where the model was doing well and where it struggled [7]. These metrics showed how close the predicted gestures were to the actual ones [9], [11].

around too much. Using MediaPipe for landmark extraction played a big role here because it filtered out unnecessary background details and helped the model focus only on the shape of the hand rather than the whole image [10], [12].

Overall, the results show that the system is quite capable of recognizing ASL alphabets in real time. It handles the usual indoor lighting and varying angles reasonably well, which makes it practical for

normal use [3], [9]. A few signs may still be misread when the shapes are almost identical, but even with those limitations, the system offers dependable performance and moves a step closer toward creating a fully automated sign-language translation tool [14].

7. FUTURE WORK

Nevertheless, in spite of the fact that the existing system is able to properly identify the ASL alphabet gestures, a number of aspects are possible to expand and enhance the project in the future. A good avenue to follow is to add more modalities, including facial expressions, movements of lips, and two-hand gestures. These elements are significant in complete sign language communication and by combining them the system may be able to comprehend more complex words and phrases [5], [13].

The next improvement option can be considered the con- sideration of more powerful deep learning architectures like CNN-LSTM hybrids, 3D Convolutional Networks, or Trans- former architectures. These models can more effectively model both spatial and temporal movement patterns and can be much more effective in enhancing the accuracy of dynamic
or continuous gestures [8], [14]. It is also possible to improve the system to automatically retrain itself as the sample of a gesture is gathered to adapt to the environment of various users and to the conditions of the real world [6].

Besides this, a more accessible system with better visual- ization tools would be developed to increase the accessibility. It might also provide a mobile application or a web-based platform in the future to ensure that the interpreter can be used in different devices. The speech-to-sign translation would also be beneficial as the integration would allow the two-way communication and the system would not only be useful to the deaf and mute community, but also to the people who are not aware of the sign language [4], [10].

In general, the prospects of the system to be expanded into a complex system that can identify full sentences, multiple sign languages, and gestures that occur in natural conversations are high. The improved models, more information, and improved implementation will make the system a more effective real- time interactive assistant to inclusive and barrier-free commu- nication [2], [11].

8. CONCLUSION

To sum up, this project demonstrates how machine learning and computer vision can make sign language interpretation far more accessible and practical for real-world use. I didn't start with anything fancy—just the idea that a computer should be able to look at a hand sign and make a reasonable guess. And after going through the whole pipeline, it more or less worked. Instead of depending on someone else or looking at static charts, the model learned from the gesture examples and figured out the patterns on its own [1], [6].

One thing I didn't expect at first was how much time would go into cleaning the data and preparing it. The modelling part is only one piece; the system only behaved properly after the dataset was cleaned, the landmarks were extracted correctly, and the useless variations were removed. Once that was in place, the real-time version actually responded faster than I thought it would, which made the whole setup feel more practical, not just like a college experiment [7], [12].

At the end of it, I feel the system really can make commu- nication a bit easier for people who use sign language every day. It's not perfect—some alphabets still confuse the model, and there's plenty of work left if someone wants full-word or sentence recognition. But as a beginning, it shows that even a small,

well-trained model with a simple interface can reduce some of the communication barriers people face in daily life. With more data and maybe a bigger network, this approach can turn into something even more useful in real situations [3], [9].

1. R. Rastgoo, K. Kiani, and S. Escalera, "Multi-Modal Sign Language Recognition: A Comprehensive Review," Pat- tern Recognition, vol. 101, pp. 107–120, 2020.
2. T. Starner and A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 12, pp. 1371–1375, 1998.
3. A. Kumar and S. Kumar, "Indian Sign Language Recog- nition Using Hybrid Features and Deep Neural Networks," Springer Advances in Intelligent Systems and Computing, 2021, pp. 235–246.
4. F. Ronchetti, E. Quiroga, C. Estrebou, L. Lanzarini, and A. Rosete, "LSTM-Based Continuous Sign Language Recognition Using Skeleton Data," J. Comput. Sci., vol. 28,
5. pp. 20–33, 2019.
6. H. Oyedotun and A. Khashman, "Deep Learning for Gesture Recognition: A Review," Neural Comput. Appl., vol. 31, no. 3, pp. 817–828, 2019.
7. M. Saarinen, "Hand Shape Classification Using Con- volutional Neural Networks," Proc. IEEE Int. Conf. Image Processing, 2019, pp. 1845–1849.
8. J. Camgoz, S. Hadfield, O. Koller, and R. Bowden, "Neural Sign Language Translation," Proc. IEEE Conf. Com- puter Vision and Pattern Recognition, 2018, pp. 7784–7793.
9. D. P. Mandal and N. Das, "Vision-Based Hand Gesture Recognition Using CNN," Springer Computer Vision and Pattern Recognition, 2020, pp. 55–67.
10. S. Sood and R. Dua, "MediaPipe-Based Real-Time Hand Tracking and Gesture Classification," IEEE Int. Conf. Computational Intelligence and Communication Technologies, 2021, pp. 147–153.
11. L. Pigou, S. Dieleman, P. Kindermans, and B. Schrauwen, "Sign Language Recognition Using Convolutional Neural Networks," Proc. European Symposium on Artificial Neural Networks, 2015, pp. 1–6.
12. P. Kishore and S. Kumar, "A Vision-Based Dynamic Gesture Recognition System for Sign Language Using HOG and SVM," Int. J. Comput. Vision Appl., vol. 12, no. 2, pp. 89–98, 2020.
13. O. Koller, H. Ney, and R. Bowden, "Deep Learning of Sign Language Recognition: A Survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 9, pp. 3911–3931, 2021.
14. G. Fang, W. Gao, and D. Zhao, "Large Vocabulary Continuous Sign Language Recognition Based on Frame and Gesture Units," Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition, 2004, pp. 1–6.

REFERENCES

1. M. Kadam, P. Patil, and S. Pawar, "Real-Time Hand Gesture Recognition Using Deep Learning and Computer Vision," Proc. Int. Conf. Computing, Communication and Intelligent Systems, 2020, pp. 112–118.