Cureus Journal of Computer Science

Open Access Original Article

Word-Based Indian Sign Language Translator Using Seq2Seq Model

Fatima A. Ansari [1] , Vivek S. Chouhan [1] , Nishikant S. Raut [1] , Rehan F. Sayyed [1] , Rohit S. Deshmukh [1]

1. Computer Engineering, M.H. Saboo Siddik College of Engineering, Mumbai, IND

Corresponding author: Nishikant S. Raut, ns2019raut@gmail.com

Received 03/08/2025
Review began 03/18/2025
Review ended 08/30/2025
Published 09/05/2025

DOI:
https://doi.org/10.7759/s44389-025-03530-7

Abstract

The word-based Indian Sign Language (ISL) translator using sequence-to-sequence (Seq2Seq) model is a gated recurrent unit-based model, which is designed to establish communication for the deaf and hearing-impaired people in India with spoken language users. It utilizes OpenCV, MediaPipe for precise landmark recognition, and a Seq2Seq model built using TensorFlow to predict the actions. This combination allows the system to translate ISL gestures into text or speech in real time, overcoming the unique grammar and syntax challenges of ISL. Trained on an ISL dataset collected from students studying ISL, this model gives accurate and real-time translations of hand signs and gestures. This system demonstrates significant potential in bridging communication gaps and improving the lives of ISL users.

Categories: AI applications, Machine Learning (ML), Deep Learning
Keywords: deaf communication technologies and indian sign language (isl), gated recurrent unit (gru), seq2seq model, recurrent neural network (rnn), long short-term memory (lstm)

Introduction

The World Health Organization estimates that 63 million people in India suffer from hearing loss; Figure *1* shows the distribution of those affected. This underscores the critical need for efficient communication solutions for the deaf and hearing-impaired community. Indian Sign Language (ISL) is crucial for interaction within this group, known for its visual expression and unique grammar and syntax. Sign language is a type of communication done between differently abled individuals using facial expressions, gestures, and body language. There exists a communication gap between the general public and the differently abled. Currently, they rely on human interpreters and small diaries through which they communicate. American Sign Language (ASL), French Sign Language, Japanese Sign Language, and ISL are among the sign languages spoken around the world. In particular, Object Detection, Deep Convolutional Neural Networks, and other tools can be used to simply represent the majority of words in ASL, which are translated using static hand gestures. ISL combines actions, facial expressions, and body language, differing from other sign languages, which may use single-hand gestures; ISL usually uses both hands. This complexity presents challenges in developing an accurate machine learning model for ISL interpretation.

How to cite this article
Ansari F A, Chouhan V S, Raut N S, et al. (September 05, 2025) Word-Based Indian Sign Language Translator Using Seq2Seq Model. Cureus J Comput Sci 2 : es44389-025-03530-7. DOI https://doi.org/10.7759/s44389-025-03530-7

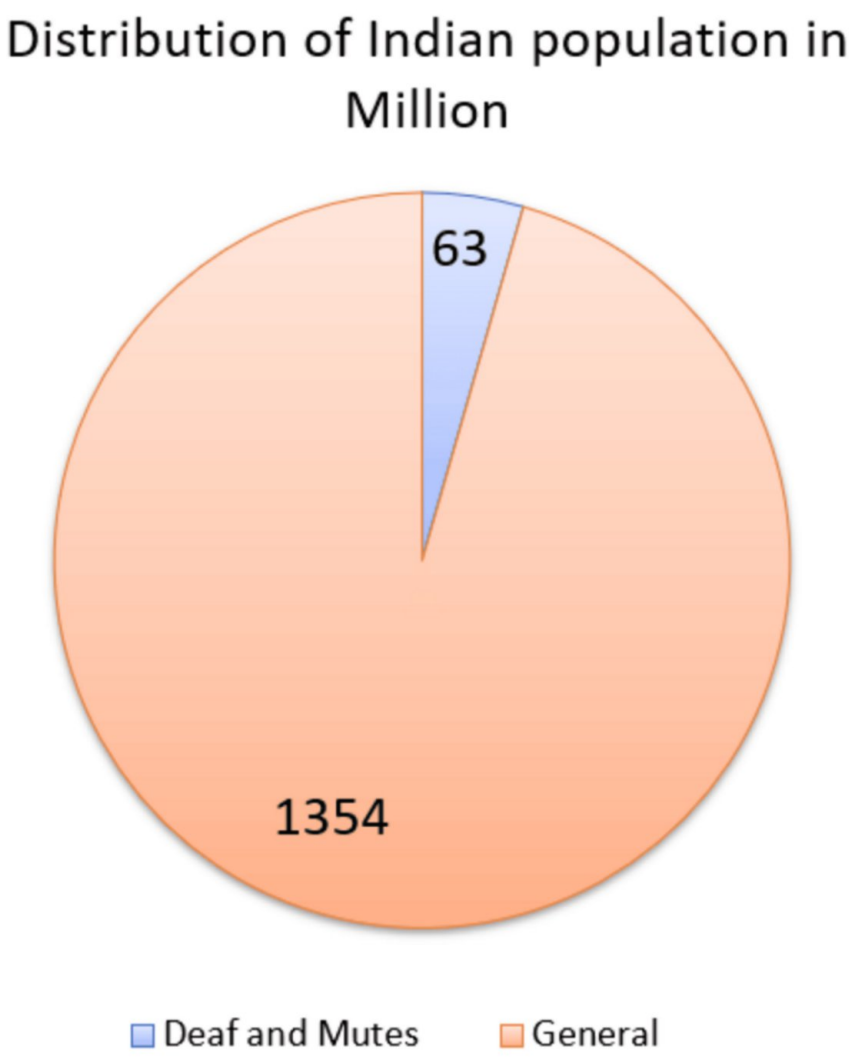

FIGURE 1: Deaf and mute population in India

Technological advancements for other sign languages like ASL, through systems like SLAIT AI and SignAll utilizing advanced computer vision and AI for real-time translation, have progressed significantly. However, eventually, as the dataset increased, their accuracy started decreasing. In contrast, progress in ISL translation tools has been slower. Efforts from platforms like DEF-ISL and institutions like the Indian Sign Language Research and Training Centre (ISLRTC) aim to advance ISL education and research but have yet to close this gap. There is no such legacy system still available, which will interpret ISL in real time and bridge this communication gap. To prevent isolation within the often-exclusive deaf community, we aim to bridge this communication gap using computer vision techniques.

Institutes like the Bombay Institute of Deaf and Mutes (BIDM), founded in Bombay in 1885 and one of the oldest schools for the deaf and mute, and the Kamla Dharamshi Narsee Shruti School in Mumbai are dedicated to teaching and empowering India's deaf and hearing-impaired communities. YouTube channels such as ISL Shiksha, Deaf Enabled Foundation, and many more are actively promoting online courses and also news through sign language.

Due to the unavailability of standardized, general word-based ISL datasets, most researchers have either collected their own data or focused on alphabet-based ISL translation. Alphabet-based translation is relatively easier, as it does not involve hand movements. In contrast, word-based signs, such as "Hello" or "How are you?", are composed of a collection of actions. To advance our research, we visited the Bombay Institute of Deaf and Mutes in Mazgaon, Mumbai, where we had effective communication with teaching professionals and staff. We collected our own real-time dataset from deaf and mute students, which included six actions, each with 150 data points. As we delve deeper into the research, the vocabulary continues to expand. The primary goal of this system’s development is to create an intelligent solution capable of recognizing ISL in real time with minimal latency, translating signs into coherent sentences for

use in online meetings and rapid communication.

Materials And Methods

Related work

A research paper published by Mariappan and Gomathi [1] presents a system for real-time ISL recognition using hand gestures and facial expressions. OpenCV's skin segmentation identifies and tracks regions of interest. Training and prediction of gestures are performed using fuzzy c-means clustering. Applications include gesture-controlled robots, human-computer interaction, and sign language interpretation, aiding communication for hearing- and speech-impaired individuals.

In order to help the hearing- and speech-impaired individuals communicate, a real-time ISL identification system that uses grid-based features to record hand gestures is described in a paper by Shenoy et al. [2]. Instead of using separate hardware, the system uses the camera of a smartphone to capture signs. Methods such as skin color segmentation and facial detection are used. Using Hidden Markov Models, hand positions are categorized using k-Nearest Neighbors, with 99.7% accuracy for static poses and 97.23% for movements.

A research paper published by Nagendraswamy et al. [3] tackles sentence-level sign recognition for ISL. It proposes methods to detect sign boundaries and extract spatial features using fuzzy membership functions. The system processes video frames to capture face and hand components, using centroids for spatial features. Variations in signs are handled with interval-valued symbolic data. A nearest neighbor classifier is used to match unknown signs, demonstrating the system's effectiveness on a large corpus of Indian regional signs.

A study by Deora and Bajaj [4] discusses a framework for recognizing ISL gestures as a pattern recognition problem. The complexity of recognizing ISL is addressed by focusing on both hands and overlapping gestures. The system successfully recognizes alphabets and numbers using Principal Component Analysis (PCA). Proposals include using neural networks and incorporating fingertip distance from the hand's centroid for improved robustness.

A research conducted by Rajam and Balakrishnan [5] suggests a technique for creating a real-time ISL recognition system for a South Indian language. It identifies 32 signs according to the binary positions of the fingers and employs static images of the palm's surface. The system detects fingertip locations through image processing methods and attains 98.125% accuracy by training on 320 images and testing on 160 images.

A dynamic hand gesture detection system utilizing OpenCV and Hidden Markov Models is proposed in a paper by Shrivastava [6]. The system is separated into phases for recognition, feature extraction, and detection. It makes use of Hu invariant moments for feature extraction and Lαβ color space for hand detection. With a recognition rate of over 90% for solitary gestures, the Forward algorithm is used for recognition and the Baum-Welch algorithm is used for training.

By creating a hand gesture recognition system, a study by Agrawal et al. [7] tackles the communication difficulties faced by deaf and mute people. The system uses a webcam to record motions, converts video frames into pictures, and uses neural networks to identify them. The models, which have been tested on a variety of datasets, continue to be accurate regardless of background, assisting in bridging the communication gap between people with and without disabilities.

Existing systems

Long Short-Term Memory (LSTM) networks are specific types of Recurrent Neural Networks (RNNs) intended to efficiently learn long-term dependencies and handle sequential input. When it comes to jobs like translating sign language, they are especially well-suited since they comprehend the progression and significance of movements over time. Using memory cells and gating mechanisms (forget, input, and output gates) that keep or discard information selectively, LSTMs address the drawbacks of conventional RNNs. A more accurate understanding of the dynamic and context-dependent character of sign language is made possible by LSTMs' capacity to preserve significant context throughout lengthy sequences of movements.

The visual components of sign language need the processing and interpretation of convolutional neural networks (CNNs). Because these networks can identify and recognize patterns, forms, and textures, they are especially good at processing grid-like data, including photos and movies. CNNs use pooling layers to compress these characteristics into more significant forms and convolutional layers to build feature maps that emphasize major information in the input pictures. CNNs may be trained to recognize and categorize many hand gestures, body language and facial expressions that are fundamental to sign language. CNNs can extract spatial data from video frames when paired with LSTMs; LSTMs then analyze the spatial

features temporally to comprehend the context and sequence of gestures across time.

Flex sensors provide precise, real-time tracking of hand and finger motions, which is a crucial component of sign language translation. These tiny, flexible components provide accurate finger position data by varying their resistance in response to bending or flexing. Flex sensors are able to record the subtle motions of individual fingers when incorporated into wearables like gloves. These movements are essential for correctly deciphering the complex sign language gestures. With the help of microcontrollers like Arduino, these sensors may be combined to build wearable technology that can record gestures. The Arduino interprets the variable resistance of the flex sensors as various voltages using analog inputs. Custom software that has been developed on the Arduino may then be used to process and interpret this data. Using libraries and code, the Arduino can be programmed to map these voltage changes to specific gestures, providing real-time feedback on the movements.

Proposed system

As part of our research, we visited the Bombay Institute of Deaf and Mutes, a prominent school in Mumbai founded in 1885. During our visit, we interacted with both students and teachers, gaining valuable insights into ISL. Some key observations that emerged include: ISL uses both hands even for representing alphabets. Facial expressions play a crucial role in differentiating similar actions. A single word often requires a sequence of hand movements. Certain actions may begin similarly but differ in their endings. These complexities make it challenging to develop a model that can accurately interpret ISL gestures. Incorrect predictions can significantly distort the intended meaning conveyed by individuals with speech and hearing impairments. Moreover, the sequence in which data is fed into the system during training and inference is critical. The entire action must be processed sequentially to preserve its semantic meaning. To achieve this, a sequence-to-sequence model, such as RNNs, LSTMs, Bidirectional LSTMs, or Gated Recurrent Units (GRUs), is essential. In the following sections, we discuss our data collection process, model training, and the results from both training and testing phases. To further enhance inference accuracy, we also developed a depth detection system that tracks eye retina movement.

Data was collected from BIDM students under the guidance of their teachers. The dataset primarily consisted of six actions: "Hello," "How Are You," "Thank You," "Sorry," "Welcome," and "Blank (for no action)". The data for these words was collected because they are frequently used in daily life. A total of 900 data points were collected, for each class containing 150 points. Each data point had a shape of (30, 1,662) and was saved to the local disk in NumPy format with a “.npy” extension. (30, 1,662) represents 30 frames and each frame having data of 1,662 array. Figure *2* visually displays the datapoint tensor. MediaPipe by Google is used to extract the landmarks while the action is being performed using its holistic model. Holistic model provides total of 543 landmarks in total which includes 468 face landmarks, 42 hand landmarks and 33 pose landmarks. Each landmark is mapped to x, y, z and visibility hence we get total of 1,662 1-d array. The key consideration was to use the face landmarks to accurately capture and store facial expressions. The only preprocessing step required was to convert the BGR image to RGB before giving to the Holistic model as it expects in RGB format. Figure *3* demonstrates the entire data collection process.

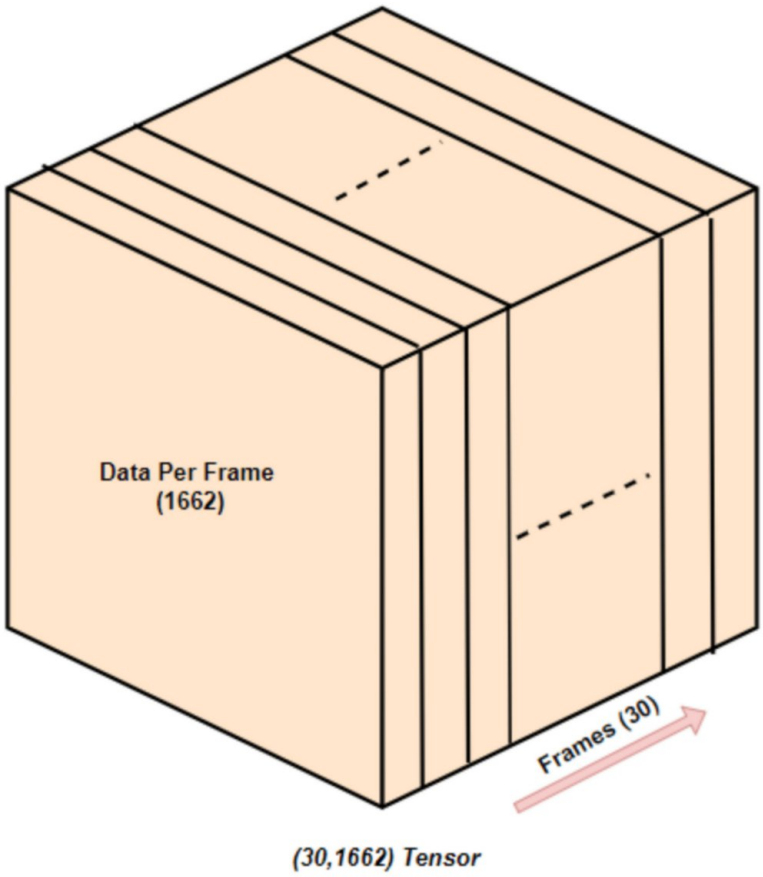

FIGURE 2: Tensor of size 30 frames × 1,662 landmarks

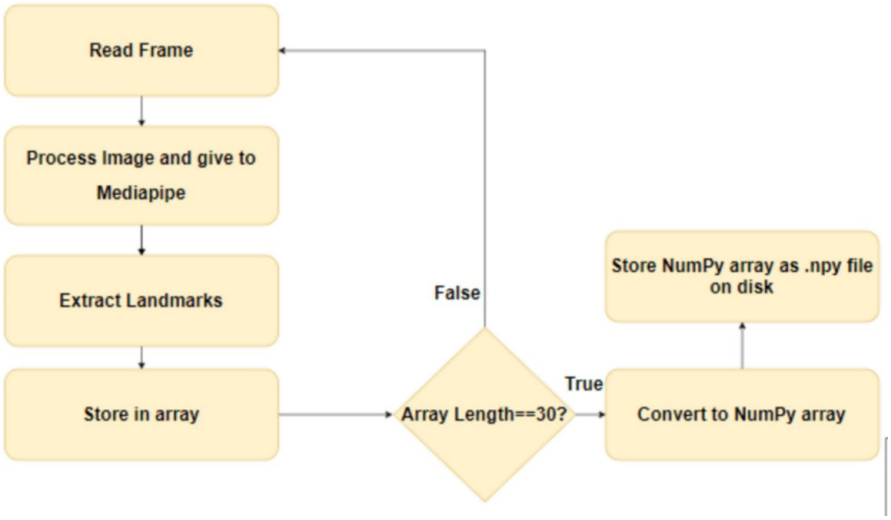

FIGURE 3: Data collection process

This system integrates depth detection algorithm, an addon to ensure subject is at the correct depth during inference. It also has ready model incorporated, which automatically checks if the user is ready and at correct distance for inference. These systems are developed using MediaPipe again. Using the Facemesh model, we can locate the left eye and right eye retina of the person in real time. Figure *4* presents the Medipipe's different models visually.

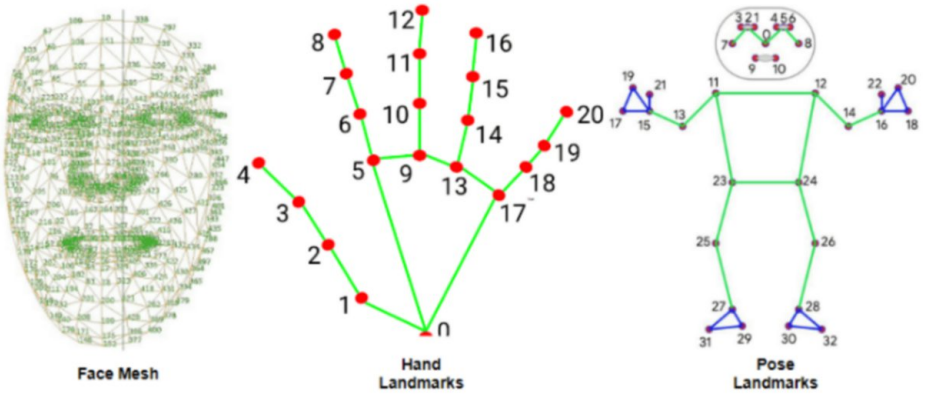

FIGURE 4: MediaPipe holistic landmarks

To obtain the landmarks for the left and right eyes, we use the 134th and 374th points, respectively. This method facilitates efficient retina tracking, as shown in Figure 5. The Euclidean distance between these points is then calculated. On average, the distance between the two retinas of a human eye is approximately 6.6 cm. This distance is converted to pixels based on an average focal length assumed to be 840 pixels using the formula as shown below.

$$d = \frac{(\text{average distance between two retinas of human eye} \times \text{focal length})}{\text{distance between two retinas in pixels}} \quad (1)$$

where d = Approximate Depth

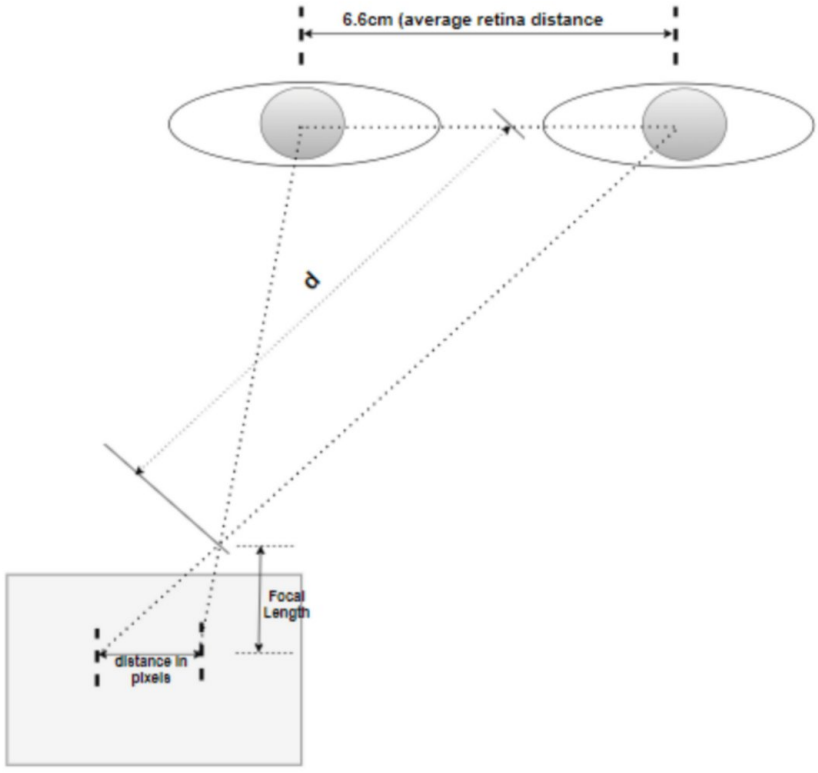

FIGURE 5: Depth detection using retina distance

Once the distance is obtained in pixels, it is compared with a prespecified range to control the depth or the distance from the digital screen to the subject/human. Important point to be noted is that the depth '*d*' is the approximate depth and not the actual one, but it helps greatly in positioning the user at the correct

location. Figure 6 depicts the states when the user is at the correct depth and when they are not, indicated by green and red bands, respectively. If the person is in the range, then the ready model system starts working. To automate the process and avoid human intervention, ready model is developed only using the hands model. Ready model checks if the user is ready to start inference and will show a thumbs up if is. Hand landmarks of being ready were collected and mapped to x, y, z values as a part of the dataset, where each datapoint had shape of (66,) and saved in a csv file. The ready model is trained using deep neural network using TensorFlow and accurately predicts the state with 99.4% accuracy. Once the person is ready, inference starts.

FIGURE 6: Depth detection when user stands at wrong and correct depths

Model building and training

First, we will review some key points about why the GRU is essential and necessary for modelling the ISL data collected. Then, we will provide information about the GRU and finally present the architecture of our GRU model. In ISL each action is performed by doing a set of movements and by using facial expressions. Each datapoint is a set of actions sequentially arranged in a tensor. Hence, they need to be processed sequentially to maintain the end-to-end semantics. Therefore, RNNs are ideal choice to model this data as output as time-step $t_{n\text{-}1}$ is also given as input when processing data at t_n. The entire movement can be captured and processed accurately.

These RNNs drive numerous well-known applications such as voice search and Google Translate. Like convolutional neural networks (CNNs) and feedforward, RNNs acquire knowledge from training data. What distinguishes them is their "memory," enabling them to utilize information from earlier inputs to affect current inputs and outputs [8]. In contrast to standard deep neural networks, which view inputs and outputs as separate entities, RNNs take into account the preceding elements of the sequence when producing outputs. While possessing knowledge of future events could aid in making predictions, unidirectional RNNs are limited to utilizing only past data to shape their outputs.

$$h_t = f(W_h \cdot h_{t-1} + W_x \cdot x_t + b_h) \qquad (2)$$

where h_t = Hidden state, W_h = Weight matrix for the hidden state, h_{t-1} = Hidden state from the previous time step, W_x = Weight matrix for the input, x_t = Input at the current time step, b_h = Bias vector for the hidden state, and f = Activation function (e.g., Tanh, ReLU).

$$y_t = g(W_y \cdot h_t + b_y) \qquad (3)$$

where y_t = Output, W_y = Weight matrix for the output, h_t = Current hidden state, b_y = Bias vector for the output, and g = Activation function (e.g., softmax for classification tasks).

An improvement over the conventional RNN is the GRU. GRUs have similarity with LSTM. GRU also uses gates to control flow of information like LSTM. They are relatively new as compared to RNN. This is the reason they offer some improvement over RNN and have simpler architecture. There are various limitations of RNN such as vanishing gradients, exploding gradients, limited memory, and training time. Due to these limitations, it becomes difficult to train RNN's on complex data, which has long sequences. Each data point in the collected dataset contains 1,662 features per time step. Standard RNNs struggle to effectively preserve the sequential context, resulting in lower accuracy. GRU helps us solve this problem using special gates like Update gate and Reset gate to control the flow of information within the network. These gates serve as filters, selecting which historical data to update, discard, or preserve. They are also

the best choice for sign language recognition as they tend to remember longer sequences.

$$R_T = \sigma\left(W_R \cdot [H_{T-1}, X_T]\right) \quad (4)$$

where R_T = Reset gate, W_R = Weight matrix for the reset gate, learned during training, H_{T-1} = Hidden state from the previous time step, X_T = Input at the current time step, and σ = Sigmoid activation function, squashing values between 0 and 1.

$$Z_T = \sigma\left(W_Z \cdot [H_{T-1}, X_T]\right) \quad (5)$$

where Z_T = Update gate, W_Z = Weight matrix for the update gate, learned during training, H_{T-1} = Hidden state from the previous time step, X_T = Input at the current time step, and σ = Sigmoid activation function, squashing values between 0 and 1.

$$\tilde{h}_t = \tanh\left(W_h \cdot [r_t * h_{t-1}, x_t]\right) \quad (6)$$

where $h_{t\sim}$ = Candidate Activation vector, W_h = Weight matrix for the candidate activation, learned during training, r_t = Reset gate (RT), h_{t-1} = The previous hidden state (H_{T-1}), and tanh = Hyperbolic tangent activation function, scaling values between -1 and 1.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (7)$$

where h_t = New hidden state, z_t = Update gate output, controlling the blend of old and new information, h_{t-1} = The previous hidden state, and $h_{t\sim}$ = Candidate Activation vector.

The new hidden state, or h_t, is calculated by combining the candidate activation vector with the previous hidden state [8], which is weighted by the update gate.

GRU architecture

In the following, we will describe the GRU architecture that we have developed and has been found permissible for ISL translation. Firstly, the input layer, which accepts 30 frames as input, each having 1,662 features, is shown in Figure 7. Multiple features can lead to the curse of dimensionality, but GRU handles this data very perfectly. The input layer is one of seven layers in our architecture. Following the input layer, the next three layers are of GRU, followed by two fully connected dense layers and lastly the output layer. The amount of GRU layers and GRU cells that each layer comprises should be considered when constructing the architecture. This is achieved through trial and error. After trying several architectures, this architecture proved better than other ones as we achieved higher testing accuracy. The first GRU layer contains 64 cells, the second contains 128 cells, and the third contains 32 cells. The first dense layers contains 64 perceptrons and the second contains 32 perceptrons with ReLU activation function. This allows the weights to converge properly while training. Finally the output layer consists of six perceptrons with softmax as the activation function as we worked on six actions, which are used most frequently. The loss function used is categorical cross-entropy as the labels were given to the model in one hot-encoded form, and the optimizer used was Adam.

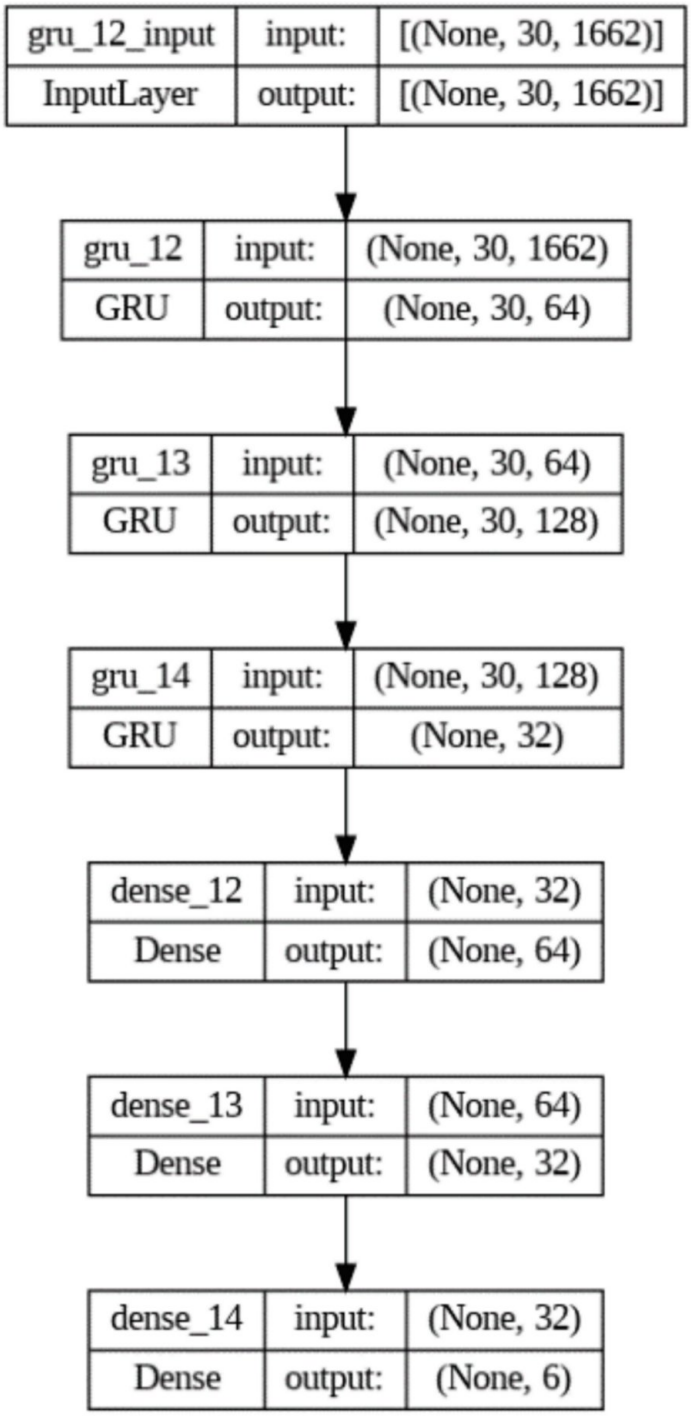

FIGURE 7: GRU architecture

GRU: Gated Recurrent Unit

Algorithm

Figure *8* illustrates the algorithm's flow and the pseudocode used to create the smart system.

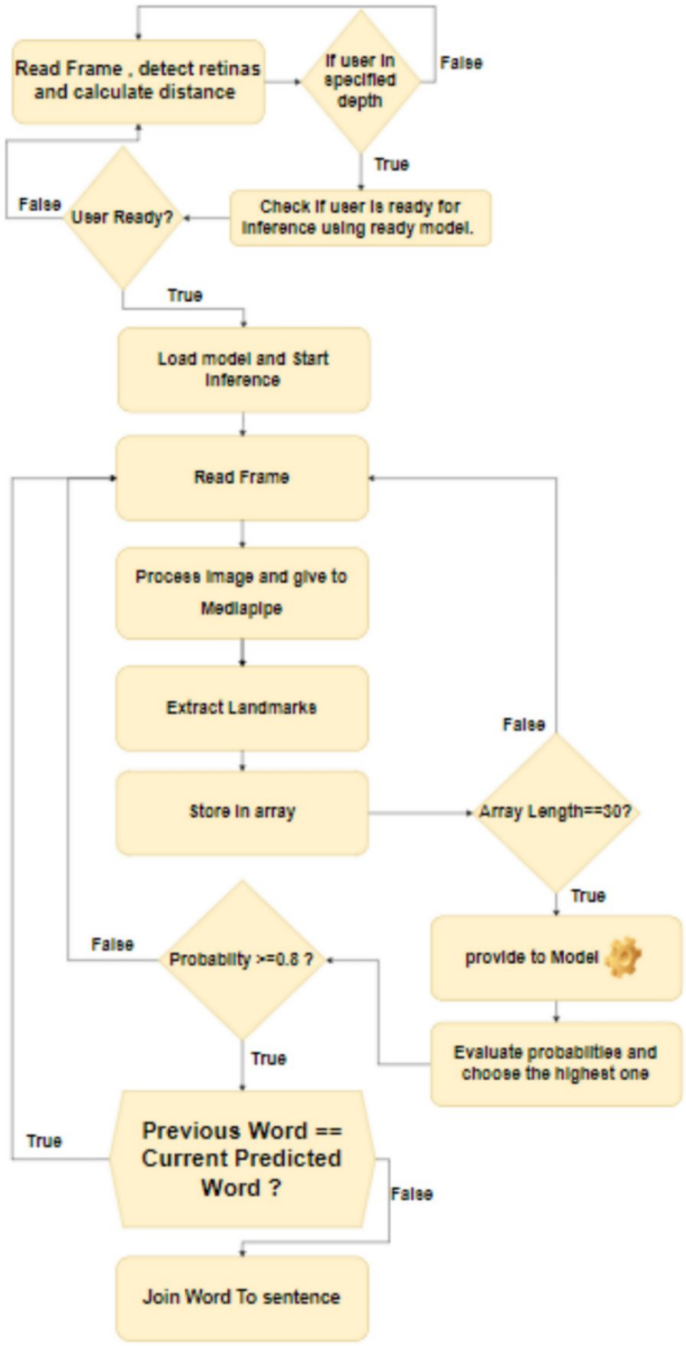

FIGURE 8: The translation system's algorithm

Results

The model was trained using TensorFlow for 24 epochs on CPU, which took around three to four minutes. It can be observed from Table *1* that GRU achieved better accuracy than LSTM and Simple RNN. Also, it took comparatively less epochs than the other two. LSTM yielded a very low testing accuracy because it could not interpret actions with similar movements.

Type of model	Training accuracy	Testing accuracy	No. of epochs
Simple Recurrent Neural Network	89.8%	84.7%	30
Long Short-Term Memory	50.89%	17.34%	40
Gated Recurrent Unit	99.8%	98.98%	24

TABLE 1: Training summary

Various experiments with different deep learning architectures based on recurrent neural networks RNNs were done to ensure a generalized and robust model is trained, as shown in Table *2*. The goal was to determine which model and optimization technique would yield the best performance in terms of accuracy and training efficiency. Testing and trials were done on three types of models - Simple RNN, LSTM, and GRU - with three widely used optimizers: Adam, RMSprop, and Adagrad. The results of these experiments are summarized in Table *2*. The Simple RNN model performed reasonably well across all three optimizers, but its best performance was observed with the Adam optimizer, achieving 89.8% accuracy in 30 epochs. In comparison, the model achieved 82.4% accuracy with RMSprop and 83.67% with Adagrad, though both required 60 epochs for convergence. Based on this observation, it can be said that while Simple RNN can effectively learn patterns based on the given ISL dataset, it requires careful tuning of hyperparameters to achieve optimal results. The LSTM model did not perform as well as expected. Despite being designed to handle long-range dependencies better than a Simple RNN, it struggled to achieve high accuracy in this particular task. The highest accuracy recorded for LSTM was 52.54% with RMSprop, followed closely by 50.89% with Adam and 49.6% with Adagrad. Each optimizer required 40 epochs, suggesting that the LSTM model faced challenges in learning meaningful representations from the dataset.

Model	Optimizer	Accuracy %	Epochs
Simple Recurrent Neural Network	Adam	89.8	30
	RMSprop	82.4	60
	Adagrad	83.67	60
Long Short-Term Memory	Adam	50.89	40
	RMSprop	52.54	40
	Adagrad	49.6	40
Gated Recurrent Unit	Adam	99.8	24
	RMSprop	97.9	29
	Adagrad	97.3	30

TABLE 2: Analysis of optimizers with each model

One possible reason for this underperformance could be the nature of the dataset and the complexity of ISL gestures, which might not align well with LSTM's learning mechanism. Additionally, vanishing gradients could have affected the training process, limiting the model's ability to retain long-term dependencies effectively as each data point is of 30 rows and 1,662 columns. The GRU model outperformed both Simple RNN and LSTM by a significant margin, making it the most promising model for ISL translation using keypoints. With the Adam optimizer, GRU achieved an impressive accuracy of 99.8% in just 24 epochs. The performance remained strong with other optimizers as well - 97.9% with RMSprop (29 epochs) and 97.3% with Adagrad (30 epochs). These results indicate that GRU not only learns effectively but also converges faster than the other two architectures. GRU's ability to retain long-term dependencies while avoiding the computational overhead of LSTM likely contributed to its superior performance.

On the other hand, Simple RNN had a better performance. It can be inferred from the training stats Figure *9* that till the 11th epoch, the loss exponentially decreased and the accuracy of exponentially increased. There was an increase in the loss during the 11th and fluctuated a little till the 22nd epoch, but it

gradually decreased till 0.01% at the last one. The accuracy gradually kept on increasing till the 21st epoch after which it started to plateau. Due to scarcity of data, we kept out testing data as a part of validation to monitor how well the model performs. The validation graph in Figure *10* also follows the same pattern as that of the training stats but showed a comparatively less fluctuation during training. GRU has the capability to remember previous sequences better and to learn spatial-temporal features from a raw video stream. The majority of the signs are accurately identified, according to the classification report shown in Figure *11*, which shows the accuracy, recall, and f1-score.

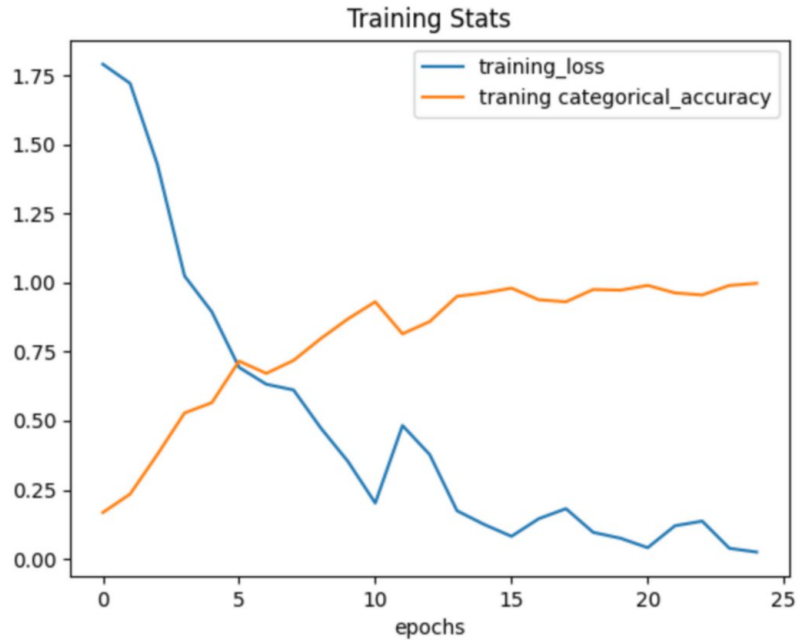

FIGURE 9: Training stats

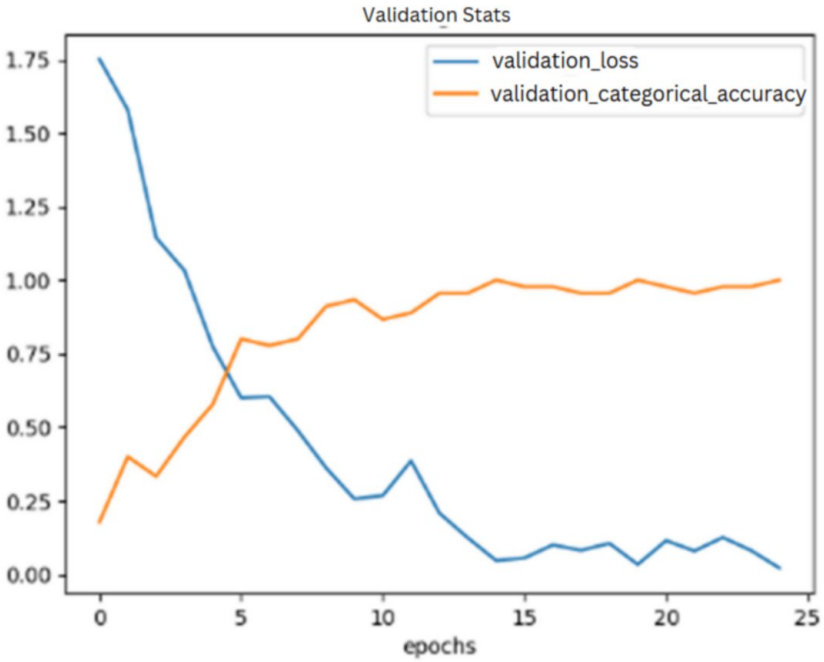

FIGURE 10: Validation stats

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6
1	1.00	1.00	1.00	7
2	1.00	1.00	1.00	7
3	1.00	1.00	1.00	7
4	1.00	1.00	1.00	7
5	1.00	1.00	1.00	11
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

FIGURE 11: Classification report

Once the model was trained successfully, an automated system was developed, which combined all the technologies developed specially for inference. This included the depth estimation and the readiness model. It used python and libraries such as OpenCV, MediaPipe, etc. OpenCV is used for capturing the video stream through the local machine. The user initially must go through the depth detection and readiness phases to reach the inference state. This helps positioning the user correctly, and inference can be done accurately by the model. This increases the user satisfaction. You can see sentence formation in Figure *12*.

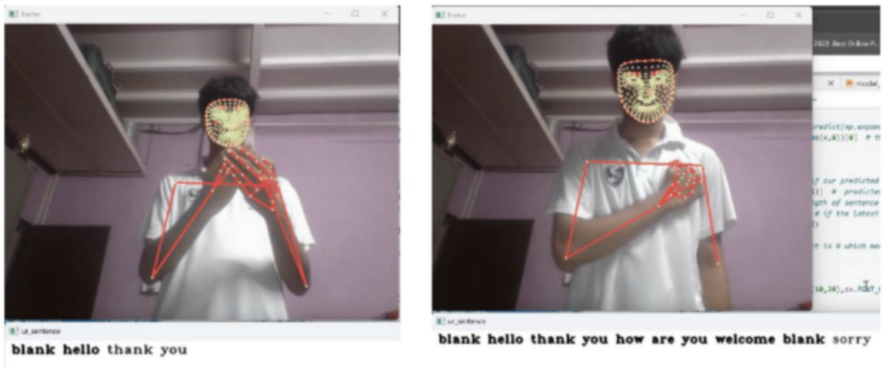

FIGURE 12: Sentence formation while performing actions during inference

Discussion

This study aimed to identify the most effective deep learning model and optimizer for translating ISL using keypoints. Three RNN-based architecture Simple RNN, LSTM, and GRU were evaluated with three optimizers: Adam, RMSprop, and Adagrad. The results demonstrated that GRU outperformed the other models, achieving a remarkable 99.8% accuracy with Adam in just 24 epochs. GRU's superior performance can be attributed to its ability to efficiently capture spatial-temporal dependencies while avoiding the computational complexity of LSTM. Furthermore, GRU maintained strong performance across other optimizers, achieving 97.9% accuracy with RMSprop in 29 epochs and 97.3% with Adagrad in 30 epochs. This consistency highlights GRU's robustness and adaptability, making it the optimal choice for ISL translation tasks, particularly in real-world applications where computational efficiency and accuracy are critical.

In contrast, Simple RNN achieved a maximum accuracy of 89.8% with Adam in 30 epochs, but its performance declined with RMSprop (82.4%) and Adagrad (83.67%), both requiring 60 epochs for

convergence. While Simple RNN demonstrated the ability to learn patterns from the ISL dataset, its limitations in handling long-term dependencies were evident, as reflected in the fluctuations in loss and accuracy during training. LSTM, despite its theoretical advantages in modeling long-range dependencies, underperformed significantly, achieving only 52.54% accuracy with RMSprop, 50.89% with Adam, and 49.6% with Adagrad, all within 40 epochs. This subpar performance may be attributed to the complexity of ISL gestures, which involve subtle spatial-temporal variations, as well as potential issues such as vanishing gradients. The dataset's structure, comprising 30 rows and 1662 columns per data point, may have further exacerbated these challenges, limiting LSTM's ability to learn meaningful representations.

The choice of optimizer played a pivotal role in model performance. Adam consistently delivered the best results across all architectures, enabling GRU to achieve near-perfect accuracy with minimal training time. RMSprop and Adagrad also performed well but required additional epochs to achieve comparable results, indicating that Adam's adaptive learning rate and momentum-based updates are particularly well-suited for this task. For Simple RNN, Adam proved to be the most effective optimizer, while LSTM showed marginal improvements across optimizers, underscoring its inherent limitations in this application. These findings emphasize the importance of selecting an appropriate optimizer-architecture pairing to maximize performance. Additionally, the development of an automated system integrating depth estimation and readiness models demonstrated the practical feasibility of deploying GRU-based models for real-time ISL translation. This system ensures accurate user positioning and enhances inference reliability, highlighting its potential for real-world deployment.

Despite these promising results, the study has limitations, particularly the limited dataset size, which necessitated the use of testing data for validation. Expanding the dataset to include a wider variety of gestures and users would improve the model's generalizability and robustness. Future work could explore advanced techniques such as hybrid architectures combining CNNs and RNNs, attention mechanisms, or transfer learning to further enhance performance. Additionally, investigating the impact of data augmentation and hyperparameter optimization could address some of the challenges posed by the current dataset. Overall, this study establishes GRU as a highly effective model for ISL translation and provides a strong foundation for future research in this domain. By addressing these limitations and building on these findings, more efficient and accessible sign language translation systems can be developed.

Conclusions

The paper presents an ISL Translator using a Seq2Seq model built with TensorFlow, OpenCV, and MediaPipe for real-time gesture translation. It uses a dataset of common ISL actions and employs GRUs to handle sequential data, achieving high accuracy. Depth detection and readiness checks ensure proper user positioning for accurate inference. This system significantly enhances communication for the deaf and hearing impaired community in India.

Appendices

ISL dataset

The dataset containing six signs "hello", "how are you", "sorry", "thank you", "welcome", and "blank" is provided in the Google Drive. Each folder contains 75 other folders (whole sequence of gesture in one folder) in which 30 .npy files are stored, which are landmarks for each frame: https://drive.google.com/drive/folders/1xggJVVCi8AYd1BmmqVctOrPopRBFG0g5?usp=sharing

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Nishikant S. Raut, Rehan F. Sayyed, Vivek S. Chouhan, Fatima A. Ansari, Rohit S. Deshmukh

Acquisition, analysis, or interpretation of data: Nishikant S. Raut, Rehan F. Sayyed, Vivek S. Chouhan, Fatima A. Ansari, Rohit S. Deshmukh

Drafting of the manuscript: Nishikant S. Raut, Rehan F. Sayyed, Vivek S. Chouhan, Fatima A. Ansari, Rohit S. Deshmukh

Critical review of the manuscript for important intellectual content: Nishikant S. Raut, Rehan F. Sayyed, Vivek S. Chouhan, Fatima A. Ansari, Rohit S. Deshmukh

Supervision: Nishikant S. Raut, Rehan F. Sayyed, Vivek S. Chouhan, Fatima A. Ansari, Rohit S. Deshmukh

Disclosures

Human subjects: Consent was obtained or waived by all participants in this study. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. Mariappan MH, Gomathi V: Real-time recognition of Indian sign language. 2019 International Conference on Computational Intelligence in Data Science (ICCIDS). 2019, 1-6. 10.1109/ICCIDS.2019.8862125
2. Shenoy K, Dastane T, Rao V, Vyavaharkar D: Real-time Indian sign language (ISL) recognition. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). 2018, 1-9. 10.1109/ICCCNT.2018.8493808
3. Nagendraswamy HS, Chethana BM, Chinmayi RL: Indian sign language recognition: An approach based on fuzzy-symbolic data. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2016, 1006-1013. 10.1109/ICACCI.2016.7732176
4. Deora D, Bajaj N: Indian sign language recognition. 2012 1st International Conference on Emerging Technology Trends in Electronics, Communication & Networking. 2012, 1-5. 10.1109/ET2ECN.2012.6470093
5. Rajam PS, Balakrishnan G: Real time Indian sign language recognition system to aid deaf-dumb people. 2011 IEEE 13th International Conference on Communication Technology. 2011, 737-742. 10.1109/ICCT.2011.6157974
6. Shrivastava R: A hidden Markov model based dynamic hand gesture recognition system using OpenCV. 2013 3rd IEEE International Advance Computing Conference (IACC). 2013, 947-950. 10.1109/IAdCC.2013.6514354
7. Agrawal M, Ainapure R, Agrawal S, Bhosale S, Desai S: Models for hand gesture recognition using deep learning. 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA). 2020, 589-594. 10.1109/ICCCA49541.2020.9250846
8. Recurrent Neural Networks (RNN). What are Recurrent Neural Networks. (2023). Accessed: April 20, 2024: https://medium.com/@malihaider240/what-are-recurrent-neural-networks-rnn-764b992f62f0.