A Mini-Project Report on

Conversion of Indian Sign Language(ISL) to Text and Speech

Submitted in partial fulfillment of the
degree of Bachelor of Technology in
Computer Engineering

Submitted by
Vidhi Agre(01), Kanchan Bhange(04), Shruti Ghume(17)

Under the guidance of
Ms. Toshi Jain

USHA MITTAL INSTITUTE OF TECHNOLOGY (SNDT Women's University)
3RPJ+H5Q, Juhu-Tara Road, Sir Vitthaldas Vidyavihar Santacruz (W),
Mumbai, Maharashtra 400049

2024-2025

Approval Sheet

This is to certify that Shruti Ghume,kanchan Bhange,Vidhi Agre has completed the mini project report on the topic "Conversion of Sign Language to Text and Speech using Convolutional Neural Network (CNN)"satisfactorily in partial fulfillment for the Bachelor's Degree in Computer Engineering under the guidance of Ms.Toshi Jain during the year 2024-2025, as prescribed by SNDT Women's University

Guide

Ms.Toshi Jain

Head of Department

Ms.Rachana Dhanawat

Principal
Mr.Yogesh Nerkar

Examiner 1

Examiner 2

Declaration

I declare that this written submission represents my own ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Vidhi Agre, Kanchan Bhange, Shruti Ghume

(17,04,01)

21/02/2025

Abstract

The communication gap between the Deaf community and the hearing population has long posed a challenge. This project presents a real-time Indian Sign Language (ISL) conversion system that translates dynamic sign language sentences into both text and speech. Using MediaPipe Holistic, the system extracts keypoints from the face, hands, and body to capture fine-grained gestures. These keypoints are fed into a Long Short-Term Memory (LSTM) model trained on sentence-level sign data. To ensure robustness against environmental variations, data augmentation techniques such as scaling, rotation, shifting, and additive Gaussian noise are applied. The predicted sentence is then converted into natural-sounding speech using a Text-to-Speech (TTS) system. The proposed system achieves accurate and fluid recognition, making communication more inclusive and accessible for the Deaf community.

Keywords: Indian Sign Language (ISL), Deep Learning, Long Short-Term Memory (LSTM), Real-time Sign Recognition, MediaPipe Holistic, Keypoint Extraction, Text-to-Speech (TTS), Sequence Classification, Gesture Recognition, Human-Computer Interaction, Data Augmentation, Speech Synthesis.

Contents

List of Tables

List of Figures

Nomenclature

ISL Indian Sign Language

TTS Text-to-Speech

LSTM Long Short-Term Memory

p Dropout probability

y True class label

$\hat{\boldsymbol{y}}^{\boldsymbol{i}}$ Predicted probability for class i

Epoch One complete pass through the entire dataset

Softmax Activation function for converting output scores into probabilities

Cross-Entropy Loss Loss function for classification tasks

MediaPipe Holistic Keypoint extraction tool from face, hands, and pose

Frame A single captured image in a video sequence

Sequence A series of frames (typically 30) used for LSTM input

Keypoints Landmark coordinates extracted per frame

Accuracy Percentage of correct predictions made by the model

Confidence Score Model's certainty of a predicted class

Real-time Inference Live processing from webcam to text/speech output

Normalization Preprocessing to scale values uniformly

Prediction Smoothing Stabilizing predictions across frames

Chapter 1

Introduction

1.1 Introduction

Indian Sign Language (ISL) is a visual-gestural language used primarily by the deaf and hard-of-hearing community in India. Like many other sign languages, ISL incorporates a combination of hand gestures, facial expressions, and body movements to communicate complex ideas. However, despite its significance, there is a noticeable gap in the availability of efficient communication tools that bridge the gap between sign language users and non-sign language users.

While there are various efforts in the field of sign language recognition, most systems focus on specific domains or use limited sign vocabulary, leading to usability issues in diverse real-world settings. This limitation further compounds the challenge faced by the deaf community, making it harder for them to engage with others in various environments.

This project aims to address this gap by developing an automated system that can convert Indian Sign Language gestures into both text and speech. The primary goal is to facilitate seamless communication between people who use ISL and those who do not, by enabling real-time translation of sign language to written and spoken language. The project will focus on improving the accuracy and performance of the recognition system, using advanced machine learning techniques such as Long Short-Term Memory (LSTM) networks for sequential data processing, along with computer vision tools like MediaPipe for real-time landmark detection.

The system will use video input from a camera to recognize gestures, which will then be translated into corresponding text and spoken words. By leveraging deep learning models and state-of-the-art frameworks, the project seeks to create an end-to-end solution that can perform real-time translation effectively under varying environmental conditions (e.g., lighting, background, and hand size variation).

1.2 Objectives of the Study

The objectives of this study are centered around developing an accurate, real-time sign language recognition system capable of converting Indian Sign Language gestures into text and speech. The detailed objectives include:

1. **Data Collection and Preprocessing:** To collect a dataset consisting of videos and keypoints representing various ISL signs and preprocess the data for use in training deep learning models. This includes performing normalization, augmentation, and feature extraction to ensure the model can generalize well across different users, environments, and lighting conditions.

2. **Model Design:** To design a deep learning model based on Long Short-Term Memory (LSTM) networks to process sequential gesture data. The LSTM network will allow the system to recognize and interpret temporal dependencies between consecutive frames, improving the accuracy of recognizing gestures in real-time.

3. **Real-Time Gesture Recognition:** To develop a system that can recognize hand gestures in real-time using video input. The system will utilize MediaPipe for real-time landmark detection of both hand and body movements, providing the necessary features for gesture classification.

4. **Text-to-Speech Conversion:** To integrate a Text-to-Speech (TTS) system that will convert the recognized ISL signs into spoken language, ensuring that non-sign language users can understand the communication in real-time. The TTS system will use pyttsx3 or similar libraries to generate audio feedback.

5. **Model Evaluation and Optimization:** To evaluate the performance of the gesture recognition model using various metrics, including accuracy, precision, recall, and F1-score. In addition to evaluating the model's performance on a validation set, the system will also undergo rigorous testing in real-world scenarios to ensure robustness in different conditions (e.g., lighting, background noise, hand orientation).

6. **User Interface Development:** To create an intuitive, user-friendly interface that displays the translated text in real-time and provides voice feedback. This will enable individuals without prior knowledge of ISL to interact seamlessly with sign language users.

7. **Real-World Application Testing:** To conduct testing of the developed system in real-world environments with varied lighting conditions, backgrounds, and diverse signers. This testing phase will aim to identify and

mitigate challenges like overfitting to specific users and environmental factors that may affect performance.

8. **Improving Accessibility:** To improve the accessibility of communication for the deaf and hard-of-hearing community by providing a tool that supports both text and voice feedback. The system aims to enable greater inclusion in everyday interactions such as in educational settings, workplaces, and social gatherings.

9. **Exploring Future Improvements:** To identify potential areas for further improvement in the recognition system, such as expanding the vocabulary of recognized signs, improving real-time performance, and enhancing the system's ability to handle dynamic backgrounds and various hand gestures.

These objectives collectively aim to create a system that can bridge the communication gap between the hearing and non-hearing communities in India by enabling easy and real-time translation of ISL into text and speech. The outcomes of this study have the potential to greatly enhance communication for the deaf and hard-of-hearing individuals, making their interactions with the broader community smoother and more inclusive.

1.3 Organization of report

he structure of this paper is as follows:

1. Chapter 2: discusses the literature review, covering existing sign language recognition methods and related research.

2. Chapter 3: provides details on the proposed methodology, including dataset preparation, model architecture, and training procedures.

3. Chapter 4: presents the experimental results and performance evaluation of the ISL recognition system.

4. Chapter 5: concludes the paper with key findings, limitations, and future research directions.

Chapter 2

Review of Literature

Study	Methodology	Key Findings	Limitations
Miah et al. [1]	Multi-Culture Sign Language (McSL) recognition system using Graph Convolutional Networks (GCN) and Transformers. Incorporates multi-head self-attention (MHSA).	High accuracy across multiple datasets (ASL, BSL, LSA64, etc.). Addresses generalizability across different sign languages.	Computational complexity is higher due to deep learning architecture.
Natarajan et al. [2]	Uses MediaPipe and a hybrid CNN + Bi-LSTM model for sign language recognition, translation, and video generation. Incorporates Neural Machine Translation (NMT) and Generative Adversarial Networks (GAN).	Achieves above 95% classification accuracy, 38.06 BLEU score, and 0.921 SSIM. Demonstrates high accuracy and video quality for multiple sign languages.	Challenges remain in continuous sign language recognition and real-time implementation.
Shanableh [3]	Two-stage deep learning model for Arabic Sign Language using word count prediction and motion images. Utilizes CNN transfer learning and Bi-LSTM for recognition.	97.3% word recognition accuracy and 92.6% sentence recognition accuracy. Improves over existing approaches with context-aware word segmentation.	Limited to Arabic Sign Language; may require fine-tuning for other languages.

Table 2.1: Literature Table

2.1 Methods

research studies have focused on sign language recognition using different methods.

2.1.1 r-Based Approaches:

Traditional sign language recognition relied on wearable sensors such as data gloves and motion sensors. These systems captured hand movements and finger positions to classify gestures. While effective in structured environments, these methods were costly, required constant calibration, and were inconvenient for users due to the need for specialized hardware.

2.1.2 2.1.2 Computer Vision-Based Approaches:

Vision-based systems, which use cameras to capture images or videos of gestures, have gained popularity due to their non-intrusive nature. Feature Engineering Methods: Earlier techniques involved manually extracting features such as edge detection, shape recognition, and motion tracking. However, these methods struggled with variations in lighting, hand size, and background noise. Machine Learning Models: SVMs and HMMs were used to classify hand gestures, but their accuracy was limited when handling dynamic and complex sign sequences. Deep Learning Approaches: With the advent of CNNs, automated feature extraction from image data became possible, significantly improving recognition accuracy. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been explored to process continuous gesture sequences, enhancing real-time performance.

2.1.3 Multimodel Approaches:

Some studies have integrated multiple sensory inputs, such as combining visual, audio, and contextual cues, to improve recognition accuracy. Multimodal integration enables a deeper understanding of sign language nuances, allowing better predictions. Real-time feedback mechanisms have also been introduced to help signers refine their gestures, leading to improved communication accuracy.

2.1.4 Challenges

Despite these advancements, several challenges persist:

1. **Thresholding Issues:** In vision-based systems, maintaining accurate thresholding in different lighting conditions is challenging. Poor lighting can cause distorted grayscale images, affecting recognition accuracy.

2. **Variability in Skin Tone and Hand Shape:** Differences in skin complexion among users can lead to misclassification. Some studies have suggested using gloves to standardize recognition, but this limits usability.

3. **Gesture Proficiency:** The accuracy of recognition systems heavily depends on the user's ability to perform gestures correctly. Incorrect or unclear gestures can lead to misinterpretation.

4. **Language Expansion:** Most existing systems focus on ASL, while regional sign languages such as ISL remain underexplored. Expanding datasets and training models for native sign languages is necessary for broader adoption.

5. **Contextual Signing:** Current systems primarily recognize finger-spelling but struggle with contextual gestures representing words or phrases. Natural Language Processing (NLP) techniques are required to enhance sign language translation beyond isolated gestures.

2.2 Existing Methods

2.2.1 Sensor-Based Methods

Traditional sign language recognition relied on wearable sensors such as data gloves and motion sensors. These systems captured hand movements and finger positions to classify gestures. While effective in structured environments, these methods were costly, required constant calibration, and were inconvenient for users due to the need for specialized hardware.

2.2.2 Computer Vision-Based Approaches

Vision-based systems, which use cameras to capture images or videos of gestures, have gained popularity due to their non-intrusive nature. Earlier techniques involved manually extracting features such as edge detection, shape recognition, and motion tracking. However, these methods struggled with variations in lighting, hand size, and background noise. With the advent of CNNs, automated feature extraction from image data became possible, significantly improving recognition accuracy.

Chapter 3

Methodology

The methodology for the real-time Indian Sign Language (ISL) recognition system involves three key stages: data collection, model development, and real-time prediction. The system uses deep learning models, specifically LSTMs, to classify ISL signs from keypoints extracted from face, body, and hand landmarks via MediaPipe.
Data preprocessing standardizes and normalizes keypoints for better model performance. During training, techniques like data augmentation and regularization are used to improve generalization. Once trained, the model is deployed for real-time gesture recognition, where predictions are displayed visually, and feedback is provided through text-to-speech synthesis. Users can confirm or reject predictions and manually add punctuation to enhance sentence fluency.
This methodology ensures accurate and interactive ISL recognition in varying real-world conditions.

3.1 Data Collection and Preparation

In this project, data collection was carried out using a webcam-based interface powered by MediaPipe Holistic. Each video sample represented a sign from the Indian Sign Language (ISL) vocabulary. The holistic model simultaneously captured facial landmarks, pose, and both hand landmarks in each frame. For each sign label, 100 video sequences were recorded, each comprising 30 frames. A clear and consistent background, proper lighting, and centered camera positioning were maintained to ensure clean data capture. All frames were stored in a hierarchical directory structure organized by sign label and sequence number.

3.2 Data Preprocessing

Once the data was collected, each frame underwent keypoint extraction and normalization. Keypoints included 33 pose landmarks (4D), 468 facial landmarks (3D), and 21

landmarks for each hand (3D). To make the model invariant to the subject's position and size, all landmarks were first translated relative to a reference point (typically the nose or hips) and then scaled based on shoulder distance. This ensured uniformity in gesture representation across users. Additionally, temporal smoothing was applied using a Savitzky-Golay filter to reduce jitter. Optional augmentation techniques (noise, scaling, shifting) were introduced to increase data diversity and robustness

3.3 Model Training and Evaluation

This step involves training a deep learning model to classify Indian Sign Language (ISL) sentences using sequences of keypoints extracted from pose, face, and hand landmarks. The process includes data loading, optional augmentation, model creation, training, and evaluation using performance metrics such as accuracy and confusion matrix.

3.3.1 Data Loading and Preprocessing

The dataset is loaded, where each sample consists of a sequence of 30 frames, with each frame containing keypoints representing the pose, face, and hand landmarks. The keypoints are normalized, and the sequences are stored in a 3D NumPy array. The corresponding labels are one-hot encoded to prepare the data for multi-class classification. A stratified split ensures that the class distribution is preserved between training and validation sets.

3.3.2 Data Augmentation (Optional)

To improve model generalization and prevent overfitting, data augmentation is applied to a portion of the training data. Augmentation techniques include adding Gaussian noise, applying scaling, and shifting the keypoints. This introduces variations that make the model more robust to real-world scenarios.

3.3.3 Model Architecture

A sequential deep learning model is created with two LSTM layers to capture the temporal dependencies in the input sequences. The first LSTM layer has 128 units, followed by a second LSTM layer with 64 units. Both LSTM layers are followed by dropout layers for regularization and batch normalization layers for stabilization. Finally, a dense layer with 64 units and a softmax output layer corresponding to the number of sign language classes is added.

3.3.4 Model Training

The model is compiled using the Adam optimizer and categorical cross-entropy loss function, suitable for multi-class classification. Early stopping, learning rate reduction, and

model checkpoint callbacks are used to avoid overfitting and improve training efficiency. The model is trained for 100 epochs with a batch size of 16, and its performance is monitored on the validation dataset.

3.3.5 Evaluation and Metrics

After training, the model's performance is evaluated using accuracy and loss metrics. A confusion matrix is generated to visualize the model's performance across different classes, indicating where it struggles with misclassifications. Training and validation accuracy and loss over the epochs are plotted to monitor the learning progress.

This model evaluation phase ensures that the model has been trained properly and provides insight into its ability to generalize to new, unseen data.

3.4 Real time inference and deployement

The real-time prediction system utilizes a deep learning model trained on Indian Sign Language (ISL) data to recognize and convert gestures into text and speech. The process involves capturing video input, extracting keypoints for hand, pose, and face landmarks using MediaPipe, normalizing the keypoints, and feeding them into the trained model for classification.

Key Components:

- **MediaPipe Holistic:** The MediaPipe Holistic solution is used for detecting and tracking pose, face, and hand landmarks in real-time. This provides a comprehensive understanding of the user's body posture and gestures.
- **Model Prediction:** The pre-trained model, loaded from a '.keras' file, predicts the sign language gesture in each frame. A sequence of 30 frames is processed, and predictions are made based on the extracted features.
- **Prediction Smoothing:** To avoid jittery predictions, a smoothing mechanism is applied, where predictions over the last 10 frames are considered to choose the most common prediction.
- **Speech Output:** The pyttsx3 library is used to convert the recognized signs into speech, providing an auditory output for the user.
- **User Confirmation:** Predictions are held for 3 seconds, allowing the user to confirm or reject the prediction. This is achieved by pressing the "y" or "n" keys on the keyboard, respectively.
- **Manual Insertions:** Users can manually insert words such as "am" and "are" by pressing specific keys ('a' for "am" and 'r' for "are").

- **Punctuation Insertion:** Users can insert punctuation marks (period or question mark) during the sign recognition by pressing the '.' or '?' keys.

Process Flow:

The following steps outline the process of recognizing and confirming a sign:

1. **Video Capture:** A live video stream is captured using OpenCV, and the frames are processed to detect keypoints using MediaPipe.
2. **Keypoint Extraction and Normalization:** Keypoints are extracted from the pose, face, and hand landmarks, and normalized to ensure consistency regardless of the user's position or size.
3. **Prediction:** The sequence of keypoints is passed to the trained model for classification. The model outputs a prediction, which is processed to extract the most likely sign.
4. **Confirmation:** If the confidence level of the prediction exceeds a threshold (0.7), the system waits for user confirmation for 3 seconds. The user can press "y" to confirm or "n" to reject the prediction.
5. **Speech Output:** Upon confirmation, the recognized sign is spoken aloud using pyttsx3.
6. **Sentence Construction:** A sentence is constructed by appending the confirmed signs in sequence. Punctuation can be added manually, and the full sentence can be spoken using the "s" key.
7. **Clearing Sentence:** Users can clear the entire sentence by pressing the "c" key.

The system provides real-time feedback and can process continuous gestures, allowing for efficient communication using Indian Sign Language. Additionally, the ability to handle punctuation and manually add words provides flexibility in real-world applications.

Control Interface:

The following keys are used to control the real-time prediction system:

- **y:** Confirm the predicted sign.
- **n:** Reject the predicted sign.
- **a:** Insert "am" into the sentence.
- **r:** Insert "are" into the sentence.
- **. :** Insert a period (full stop).

- **? :** Insert a question mark.
- **s:** Speak the current sentence aloud.
- **c:** Clear the current sentence.
- **q:** Quit the program.

Chapter 4

Realisation/Implementation of Indian Sign Language conversion to Text and Speech

4.1 Introduction

The Indian Sign Language (ISL) recognition system is designed to interpret hand gestures into textual and spoken output using deep learning and computer vision. This project aims to recognize dynamic gestures in real-time through a webcam, convert them to text, and provide speech feedback using a Text-to-Speech engine. The system combines MediaPipe for extracting body and hand landmarks, and LSTM (Long Short-Term Memory) networks to learn the temporal patterns in gestures.

4.2 System Architecture

The system architecture for real-time Indian Sign Language recognition consists of a modular pipeline that processes live video input from a webcam to recognize hand gestures. Initially, video frames are captured and passed through MediaPipe's Holistic model, which detects and tracks key landmarks of the hands, pose, and optionally the face. These landmarks are extracted and converted into numerical feature vectors representing the spatial position of body joints and hand movements. A sequence of 30 such frames is collected to represent one complete gesture, and each sequence is flattened into a structured array of features, forming the input for the gesture recognition model.
For gesture recognition, a Long Short-Term Memory (LSTM) neural network is used, which is well-suited for processing sequential time-series data like gesture sequences. The LSTM model is trained on multiple labeled gesture sequences, allowing it to learn the temporal patterns in hand movements. During real-time prediction, the system continuously processes live video, extracts sequences, and classifies the gesture using the

trained LSTM model. The output is displayed as text on the screen and can optionally be converted into speech using a text-to-speech engine, enabling seamless and interactive communication for users using sign language.

4.3 Media pipe Holistic method

MediaPipe is a cross-platform library developed by Google that provides efficient and accurate pose, face, and hand landmark detection. In this project, the Holistic model is used to detect:

- 33 pose landmarks
- 33 pose landmarks
- 468 face landmarks
- 21 left-hand landmarks
- 21 right-hand landmarks

Each detected landmark returns X, Y, Z coordinates, and visibility (for pose). These keypoints are combined into a single feature vector for model training.

4.4 Data collection:

Data collection is a fundamental part of building a gesture recognition system. In this project, we aim to collect data for recognizing Indian Sign Language (ISL) gestures using a real-time video feed. The collected data consists of sequences of keypoint landmarks representing different parts of the body, including hands, face, and pose. The following section provides a detailed explanation of the methodology used for data collection, including the setup, process, and organization of the collected data.

4.4.1 Data Collection Setup

The data collection is carried out using a webcam, and we employ the **MediaPipe Holistic** model for extracting pose, face, and hand landmarks from the video. The goal is to capture 12 different ISL signs, each performed by a subject, in sequences of 30 frames. These signs are:

- hello
- sign
- language
- indian

- I
- again
- bye-bye
- you
- free
- today
- hiding
- something

Each sign is recorded for **100 sequences**, making the total number of videos collected for training 1200 (12 signs $\times$ 100 sequences). The video data is organized by sign labels, and each sequence has its own folder containing the frames as keypoint data files.

4.4.2 Keypoint Extraction

The core of the data collection process involves extracting keypoints from the video frames. The MediaPipe Holistic model is used to detect and extract the following keypoints:

- **Pose Landmarks:** These landmarks represent the positions of various body joints, including shoulders, elbows, wrists, hips, knees, and ankles.
- **Face Landmarks:** These represent the positions of key facial features, such as the eyes, nose, and mouth.
- **Left Hand Landmarks:** These are the positions of landmarks on the left hand, including the fingers, palms, and wrists.
- **Right Hand Landmarks:** These are the positions of landmarks on the right hand.

Each of these landmarks is represented as **(x, y, z)** coordinates, where x and y represent the position in the 2D plane (image coordinates), and z represents the depth of the landmark (relative to the camera).
For each frame, we extract these keypoints and flatten them into a 1D array. This flattened array serves as a feature vector that represents the position of all the landmarks in the frame.

4.4.3 Data Normalization

To ensure that the model generalizes well across different subjects, the extracted keypoints are normalized. The normalization process compensates for variations in the subject's size, distance from the camera, and other factors.
Normalization is done by using the distance between the two shoulder landmarks as a reference. Specifically:

- The positions of the pose landmarks, face landmarks, and hand landmarks are shifted so that the origin (0, 0, 0) is set at the reference point (the midpoint between the two shoulders).
- The keypoints are then scaled by the distance between the shoulders, ensuring that the scale of the data remains consistent across different individuals and hand movements.

This normalization helps the model to focus on relative movements and spatial relationships between keypoints rather than absolute positions or sizes.

4.4.4 Data Augmentation (Optional)

To make the model more robust to variations in the environment and the signer's movements, data augmentation can be applied during training. Although it is not used during data collection, it can be added when training the model. The augmentation techniques that can be applied include:

- **Adding Gaussian noise:** This simulates random variations in the keypoints, mimicking slight changes in hand positioning.
- **Scaling:** The keypoints can be slightly scaled to simulate different distances between the signer and the camera.
- **Shifting:** The keypoints can be randomly shifted to account for small deviations in the signer's position.

These augmentations are applied to the data during the training process to help the model generalize better across a wider range of inputs.

4.4.5 Data Saving Structure

The keypoints for each frame are saved as individual **.npy** files, which store the extracted data in a binary format. The files are organized in a folder structure where:

- Each sign has its own directory, named after the sign (e.g., `hello`, `sign`, `language`, etc.).
- Within each sign's directory, there are subdirectories corresponding to each sequence (e.g., sequence 0, sequence 1, ..., sequence 99).

- Each sequence contains **30 frames**, with each frame's keypoint data stored in a separate `frame_X.npy` file, where `X` is the frame number.

For example, the directory structure for the sign "hello" with sequence 1 will look like this:

```
dataset/hello/1/frame_0.npy
dataset/hello/1/frame_1.npy
dataset/hello/1/frame_2.npy
...
dataset/hello/1/frame_29.npy
```

This structure ensures that each frame's keypoint data is associated with the correct sign label and sequence.

4.4.6 Real-time Video Feed and Display

During the data collection process, a live video feed is shown to the user, where the detected landmarks are overlaid on the video frame. This allows the user to visually verify the accuracy of the detected landmarks. Each frame is displayed with an overlay indicating the sign being collected, as well as the sequence and frame numbers. This live feedback helps to ensure that the data being collected is accurate and allows for corrections if necessary.
The video feed is captured using OpenCV, and the detected landmarks are drawn over the video frame in real-time using the drawing utilities provided by the MediaPipe library. This ensures that the keypoints are accurately detected and displayed.

4.4.7 Stopping Data Collection

The data collection for each sequence can be stopped manually by pressing the 'q' key. This allows the user to pause the data collection process, review the captured data, and proceed to the next sequence. Once all sequences for a particular sign are collected, the system moves to the next sign in the dataset.
The data is collected in the form of keypoints representing various body parts, including hands, face, and pose. The collected data is normalized and saved in a structured format, which is ready for use in training a machine learning model for ISL gesture recognition. The process ensures that the data is consistent, accurate, and suitable for model training, contributing to the overall goal of developing an efficient ISL recognition system.

4.5 Feature extraction:

From each frame, the extracted landmarks are flattened into a single-dimensional array. This includes:

132 pose features (33×4)

1404 face features (468×3)
63 left-hand features (21×3)
63 right-hand features (21×3)
Total: 1662 features per frame.
Each input to the LSTM model becomes a sequence of shape (30 frames, 1662 features).

4.6 Model Training and Evaluation Using LSTM Neural Network

In this step, we design, train, and evaluate a deep learning model for the classification of dynamic Indian Sign Language (ISL) sentences using sequences of human pose keypoints extracted from videos. The model leverages a Long Short-Term Memory (LSTM) based architecture, which is particularly effective for learning from sequential data, such as time-series representations of sign gestures.

4.6.1 Data Preparation

We organize the dataset into sequences of 30 frames (keypoint arrays) per sample. Each sign sentence is represented by a unique folder containing subfolders for each sequence (`frame_0.npy` to `frame_29.npy`). We perform a stratified train-test split using `train_test_split` to preserve the class distribution across training and validation datasets.

4.6.2 Data Augmentation

To enhance the model's robustness against minor variations, we optionally apply keypoint-level augmentation techniques:

1. **Additive Gaussian Noise:** Used to simulate sensor noise or lighting variation during training:
$$x' = x + \mathcal{N}(0, \sigma^2)$$
where $\mathcal{N}(0, \sigma^2)$ is Gaussian noise with mean 0 and variance σ^2.

2. **Scaling Transformation:** Resizes input keypoints or images to simulate distance changes:
$$x' = s \cdot x$$
where s is a scaling factor.

3. **Shifting (Translation) Transformation:** Simulates small movements in the input:
$$x' = x + \Delta$$
where Δ is a small translation vector added to the coordinates.

These augmentations simulate real-world inconsistencies in gesture size, position, and camera capture.

4.6.3 Model Architecture

We build a sequential deep learning model using TensorFlow Keras with the following layers:

- `LSTM (128 units)`: return sequences, L2 regularization
- `Dropout (0.4)`: for regularization

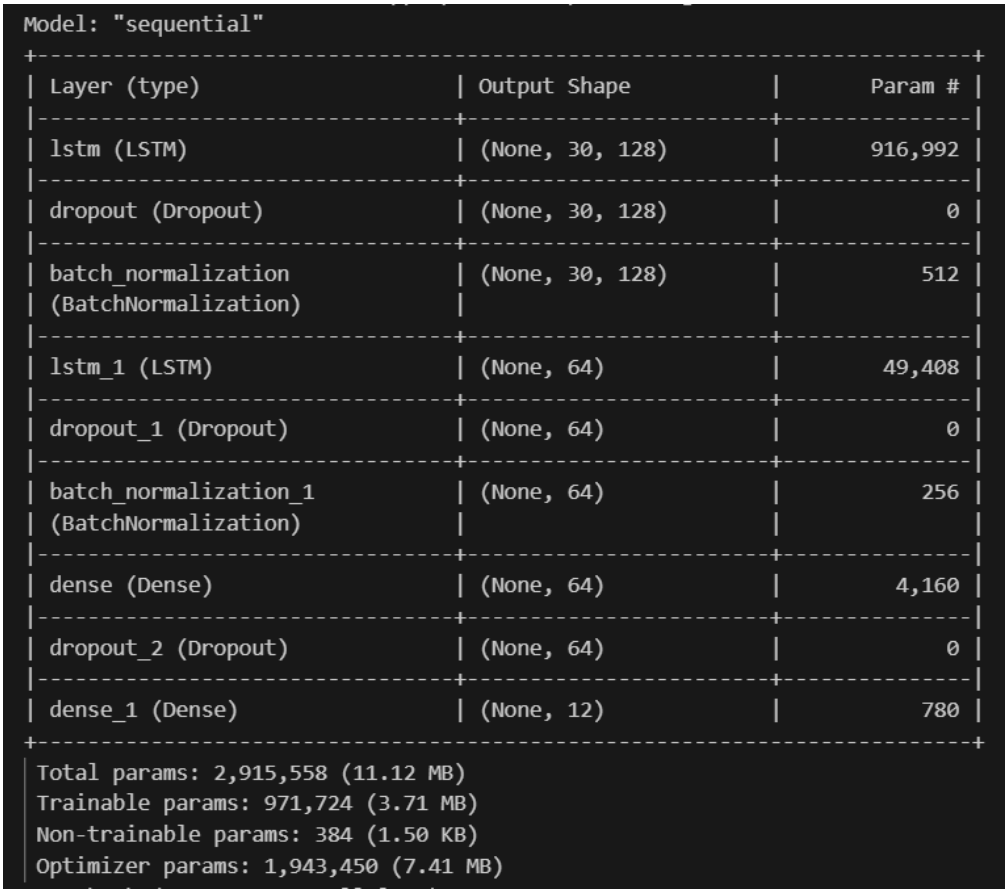
$$\text{output} = \begin{cases} 0 & \text{with probability } p \\ \text{scaled output} & \text{with probability } (1-p) \end{cases}$$

- `BatchNormalization`: stabilizes learning
- `LSTM (64 units)`: no sequence return, L2 regularization
- `Dropout (0.3)`
- `BatchNormalization`
- `Dense (64 units)`: ReLU activation, L2 regularization
- `Dropout (0.3)`
- `Output Dense`: Softmax activation with 12 output units (one for each class)

The model is compiled using the categorical crossentropy loss:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

We use the Adam optimizer for efficient gradient-based optimization.



Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 30, 128)	916,992
dropout (Dropout)	(None, 30, 128)	0
batch_normalization (BatchNormalization)	(None, 30, 128)	512
lstm_1 (LSTM)	(None, 64)	49,408
dropout_1 (Dropout)	(None, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dense (Dense)	(None, 64)	4,160
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 12)	780

Total params: 2,915,558 (11.12 MB)
Trainable params: 971,724 (3.71 MB)
Non-trainable params: 384 (1.50 KB)
Optimizer params: 1,943,450 (7.41 MB)

Figure 4.1: Model Summary

4.6.4 Training Setup

- **Epochs**: 100
- **Batch Size**: 16
- **Callbacks**:
  - `EarlyStopping`: Stops training if validation loss doesn't improve for 10 epochs.
  - `ReduceLROnPlateau`: Reduces learning rate on performance plateau.
  - `ModelCheckpoint`: Saves best model based on validation accuracy.

4.6.5 Evaluation and Visualization

After training, the model is evaluated using:

- **Accuracy and Loss Curves**: Plots of training and validation accuracy/loss over epochs.
- **Confusion Matrix**:

$$\text{CM}(i, j) = \text{Number of times true class } i \text{ was predicted as class } j$$

A confusion matrix is generated to visualize classification performance across all classes. This is plotted using `seaborn.heatmap()` for better clarity.

The trained model is saved as `isl_model.keras`, which will be used in the real-time inference system.

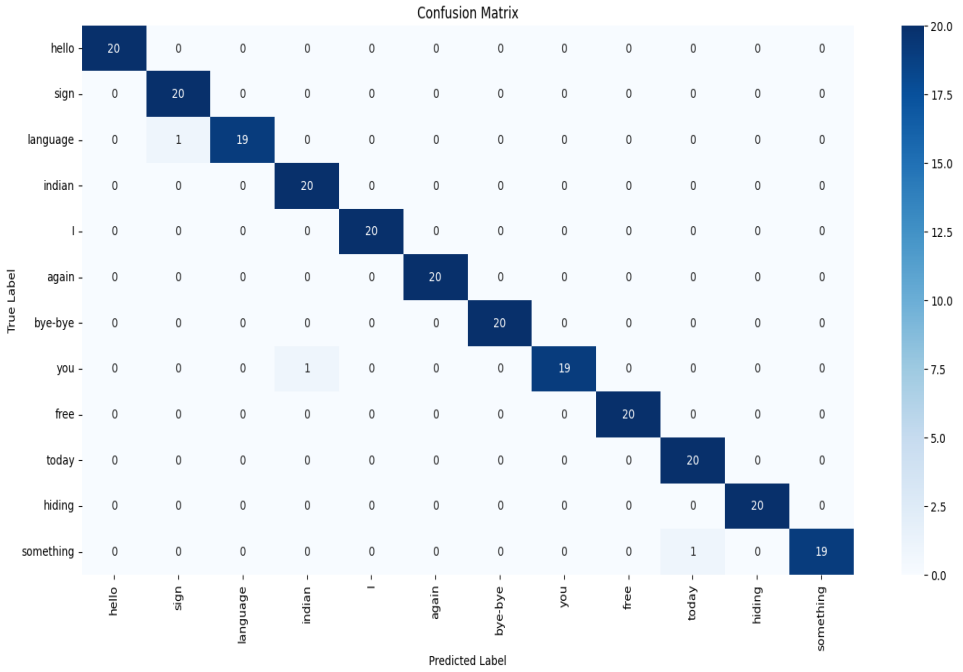


Figure 4.2: Confusion Matrix

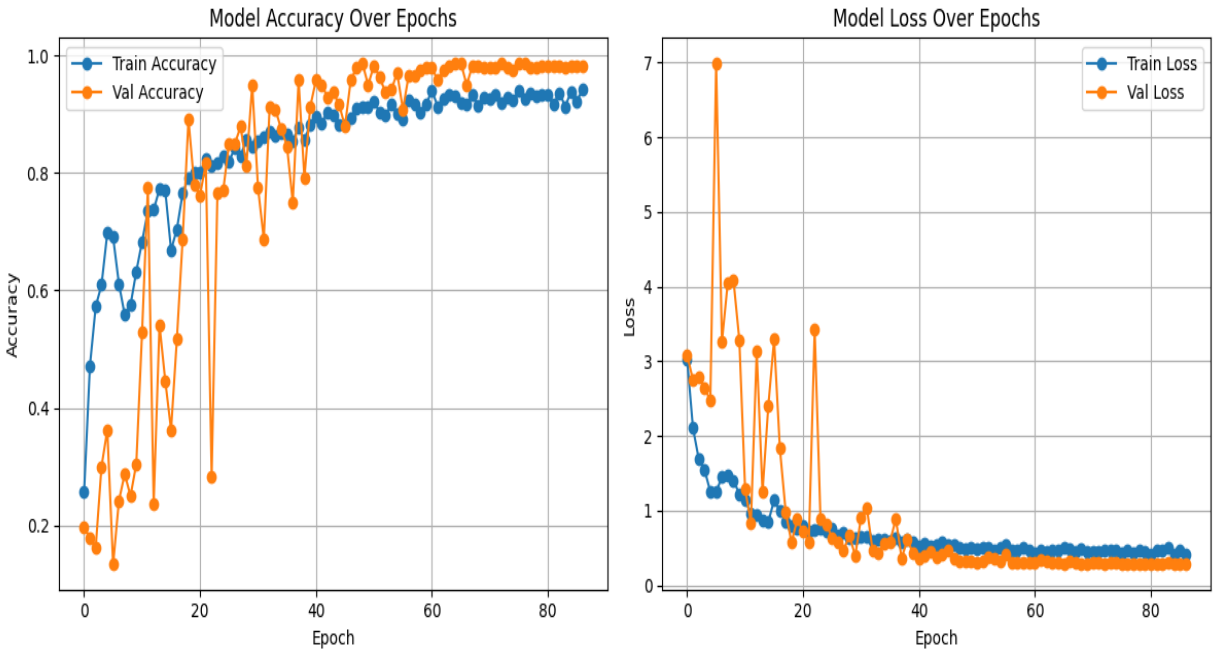


Figure 4.3: Model Accuracy and Loss Over Epochs

Where,**Cross-Entropy Loss=**

$$L = -\sum_{i} y_i \log(\hat{y}_i)$$

4.6.6 Real time prediction:

The real-time ISL recognition system converts dynamic Indian Sign Language gestures into text and natural-sounding speech. It combines computer vision, deep learning, and speech synthesis into a single interactive application. The implementation was done using Python, TensorFlow, OpenCV, MediaPipe, and pyttsx3.

System Architecture and Pipeline

The system pipeline includes the following stages:

1. **Video Capture:** The webcam feed is accessed using OpenCV (`cv2.VideoCapture(0)`), and each frame is read and processed in real-time.
2. **Pose and Landmark Extraction:** MediaPipe Holistic is used to detect 3D keypoints for the face (468), pose (33), left hand (21), and right hand (21). These are concatenated into a single keypoint vector:
   - Face: 468 landmarks × 3 coordinates (x, y, z)

- Pose: 33 landmarks × 4 values (x, y, z, visibility)
- Hands: 21 landmarks × 3 coordinates (per hand)

3. **Normalization:** Keypoints are normalized:
   - Translated relative to the nose landmark.
   - Scaled based on the distance between the left and right shoulder to make the model invariant to scale and distance.
4. **Temporal Sequence Formation:** A `deque` of length 30 frames forms a sequence window. Each time the sequence reaches full length, it is passed to the trained model for prediction.
5. **Smoothing:** A Savitzky-Golay filter is applied on the keypoint sequence to reduce jitter. The output predictions over the last 10 frames are stored and their most frequent class is selected for stability.

   Prediction Smoothing:
   $$\hat{y}_{\text{smooth}} = \frac{1}{N} \sum_{t=1}^{N} \hat{y}_t$$
6. **Sign Prediction:** The trained CNN-BiLSTM model outputs a probability vector for each sign. If the maximum probability exceeds a confidence threshold (e.g., 0.7), the predicted sign is displayed and temporarily held for confirmation.
7. **User Confirmation:** A 3-second confirmation window is activated. The user is prompted to press 'y' to accept or 'n' to reject the prediction. Accepted signs are added to a list of confirmed signs.
8. **Speech Output:** When a sign is confirmed, the system uses `pyttsx3` to vocalize the word. The speech rate is adjusted for normal and question-like sentences.
9. **Sentence Construction:** Users can insert predefined grammar words such as "am" (key 'a') and "are" (key 'r') manually. They can also add punctuation using '.' or '?'. Pressing 's' speaks the constructed sentence aloud.
10. **Reset and Exit:** The sentence can be cleared using the 'c' key, and the program can be terminated using 'q'.

Real-time Interface and Display

The OpenCV window displays the following information in real-time:

- **Live Webcam Feed** with drawn landmarks for pose, face, and both hands using MediaPipe's drawing utilities.

- **Predicted Sign and Confidence Score** shown in green (above threshold) or red (below threshold).
- **Instruction Prompts:** Dynamic text on screen guides the user to confirm or reject signs.
- **Constructed Sentence Preview:** The full confirmed sentence is displayed at the bottom.
- **Interactive Keys Guide:**
  - `'y'`: Confirm predicted sign
  - `'n'`: Reject predicted sign
  - `'a'`: Insert "am"
  - `'r'`: Insert "are"
  - `'.'`: Add full stop
  - `'?'`: Add question mark
  - `'s'`: Speak the sentence
  - `'c'`: Clear sentence
  - `'q'`: Quit program

Key Features and Functionalities

- **Dynamic Confirmation:** Prevents false predictions from being added by requiring user input for verification.
- **Natural Speech Output:** Adjusts rate for questions to sound natural. Uses `pyttsx3` for offline synthesis.
- **Prediction Stability:** Uses majority voting over a prediction buffer to smooth out model outputs.
- **Contextual Sentence Building:** Allows construction of grammatically correct sentences using gestures and key-assisted grammar.

Use Case Example

Consider a user performing ISL gestures corresponding to "are you free today?". The system:

1. Recognizes and confirms each word in the sentence.
2. Appends "are" manually using the 'r' key.
3. Adds "?" punctuation using the '?' key.

4. Speaks the complete sentence in a question tone when 's' is pressed.

This architecture enables natural, responsive, and robust communication using Indian Sign Language in real-time.

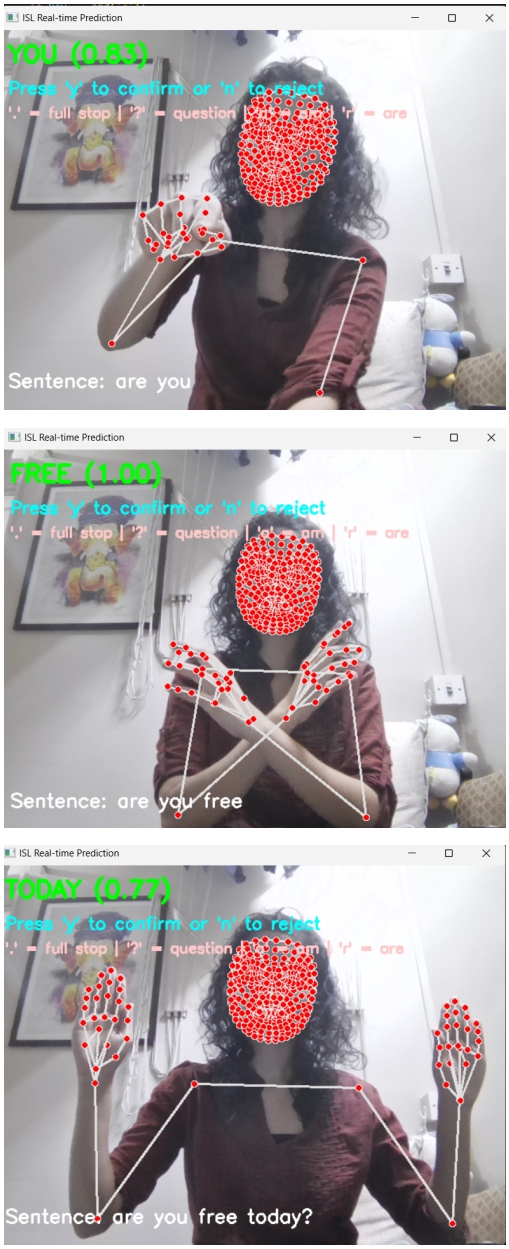


Figure 4.4: Signing steps for the sentence: are you free today?

Chapter 5

Conclusion and Future Scope

Conclusion

This project presents a real-time Indian Sign Language (ISL) recognition system that accurately translates dynamic sign gestures into textual and speech outputs. By leveraging **MediaPipe Holistic** for landmark detection and a deep learning model combining **Convolutional Neural Networks (CNN)** and **Bidirectional Long Short-Term Memory (BiLSTM)** networks, the system effectively captures both spatial and temporal features of sign language gestures. The model demonstrates high accuracy across a set of predefined ISL sentences and words, even in varying real-world conditions.
Additionally, the integration of a natural-sounding **Text-to-Speech (TTS)** engine enhances communication for users with hearing or speech impairments. The system allows real-time feedback, manual corrections, and punctuation support, thereby ensuring natural, expressive sentence construction. Overall, this approach makes sign language more accessible and bridges the communication gap between the Deaf community and non-signers.

Future Scope

- **Expansion to a Larger Vocabulary:** Future work can include support for larger, more diverse vocabularies and even free-form sentence construction using a grammar-aware decoder.
- **Multilingual Speech Output:** Incorporating support for multiple spoken languages (e.g., Hindi, Tamil, Bengali) to enhance inclusivity for regional users.
- **Continuous Sign Recognition:** Eliminate the need for fixed-length input and user confirmation, enabling fully continuous sign recognition with automatic sentence boundary detection.

- **Deployment on Mobile and Edge Devices:** Optimizing the model for Android/iOS or edge hardware (e.g., Raspberry Pi with Coral TPU) for portability and offline usability.
- **Facial Expression and Emotion Recognition:** Enhance contextual accuracy by incorporating facial expression and emotion analysis as part of sign interpretation.
- **Two-way Communication:** Enable bidirectional communication by converting spoken input back into ISL using animated avatars or AR-based hand gestures.
- **User Personalization and Learning Mode:** Introduce a module for users to train their own custom signs or use a tutorial mode for non-signers to learn ISL.

Appendix A

Important Terms

To compare quantitatively with various techniques, the following set of criteria is established to ...

1. **Indian Sign Language (ISL):** A visual language used by the deaf and hard of hearing community in India, consisting of hand gestures, facial expressions, and body movements.

2. **MediaPipe Holistic:** A framework by Google for detecting and tracking human body, face, and hand landmarks in real-time, used to extract keypoints for gesture recognition.

3. **Keypoints:** Coordinates representing specific landmarks on the body, face, or hands, used as features for gesture recognition models.

4. **LSTM (Long Short-Term Memory):** A type of recurrent neural network (RNN) designed to handle long-term dependencies in sequential data, used for recognizing gesture sequences.

5. **Sequence Length:** The number of frames considered as input to the model in one sequence, affecting how much temporal context the model can capture.

6. **Real-time Prediction:** The process of making predictions on live data, such as a webcam feed, and providing immediate feedback.

7. **Normalization:** Scaling keypoints to a consistent range to handle variations in size, orientation, and position of the subject.

8. **Softmax Confidence Score:** A probability distribution generated by the softmax function, indicating the model's confidence in its predictions.

9. **pyttsx3 (TTS Library):** A Python library for converting text into speech (Text-to-Speech), used to provide vocal feedback for recognized gestures.

10. **Savitzky-Golay Filter:** A smoothing filter applied to the keypoints sequence to reduce noise and improve prediction stability.

11. **Overfitting:** When a model memorizes the training data too well, causing poor performance on new, unseen data.

12. **Dropout:** A regularization technique where random neurons are deactivated during training to prevent overfitting.

13. **Batch Normalization:** A technique to normalize inputs to each layer, improving training speed and stability.

14. **Model Accuracy:** The percentage of correct predictions made by the model, indicating its performance on test data.

15. **Confusion Matrix:** A tool to evaluate classification performance, showing true positives, false positives, true negatives, and false negatives.

16. **Text-to-Speech (TTS):** A technology that converts written text into spoken words, used for vocal feedback in this project.

17. **Data Augmentation:** Techniques like rotation, scaling, and flipping applied to training data to improve model robustness by simulating different conditions.

18. **Activation Function (Softmax):** A function used in the output layer to convert raw scores into probabilities, aiding in classification.

19. **Cross-Entropy Loss:** A loss function used for classification tasks that measures the difference between predicted and true probability distributions.

Appendix B

Mathematics

1. **Softmax Activation Function:**

$$P(y = i \mid x) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

2. **Cross-Entropy Loss:**

$$L = -\sum_i y_i \log(\hat{y}_i)$$

3. **Dropout Regularization:**

$$\text{output} = \begin{cases} 0 & \text{with probability } p \\ \text{scaled output} & \text{with probability } (1-p) \end{cases}$$

4. **LSTM Cell Operation:**

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned}$$

5. **Prediction Smoothing:**

$$\hat{y}_{\text{smooth}} = \frac{1}{N} \sum_{t=1}^{N} \hat{y}_t$$

6. **Additive Gaussian Noise:** Used to simulate sensor noise or lighting variation during training:

$$x' = x + \mathcal{N}(0, \sigma^2)$$

where $\mathcal{N}(0, \sigma^2)$ is Gaussian noise with mean 0 and variance σ^2.

7. **Scaling Transformation:** Resizes input keypoints or images to simulate distance changes:
$$x' = s \cdot x$$
where s is a scaling factor.

8. **Shifting (Translation) Transformation:** Simulates small movements in the input:
$$x' = x + \Delta$$
where Δ is a small translation vector added to the coordinates.

References

[1] A. S. M. Miah, M. A. M. Hasan, Y. Tomioka, and J. Shin, "Hand Gesture Recognition for Multi-Culture Sign Language Using Graph and General Deep Learning Network," *IEEE Open Journal of the Computer Society*, vol. 5, pp. 1–12, 2024.

[2] T. Shanableh, "Two-Stage Deep Learning Solution for Continuous Arabic Sign Language Recognition Using Word Count Prediction and Motion Images," *IEEE Access*, vol. 11, pp. 126823–126837, 2023.

[3] B. Natarajan, E. Rajalakshmi, R. Elakkiya, K. Kotecha, A. Abraham, L. A. Gabralla, and V. Subramaniyaswamy, "Development of an End-to-End Deep Learning Framework for Sign Language Recognition, Translation, and Video Generation," *IEEE Access*, vol. 10, pp. 104358–104371, 2022.

[4] K. S. Sindhu, M. Mehnaaz, B. Nikitha, P. L. Varma, and C. Uddagiri, "Sign Language Recognition and Translation Systems for Enhanced Communication for the Hearing Impaired," in *Proc. 1st Int. Conf. Cognit., Green, Ubiquitous Comput. (IC-CGU)*, 2024, pp. 1–6.

[5] A. Ojha, A. Pandey, S. Maurya, A. Thakur, and D. P. Dayananda, "Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 8, no. 15, pp. 191–198, 2020.

[6] M. Papatsimouli, K. F. Kollias, L. Lazaridis, and G. S. Maraslidis, "Real Time Sign Language Translation Systems: A Review Study," in *Proc. IEEE Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, 2022.

[7] YouTube, "Indian Sign Language to Text and Speech — Real-time Demonstration," Available: `https://youtu.be/VtbYvVDItvg?si=CcN3eRnJGOjuCfQ7`

[8] Mendeley Data, "Indian Sign Language Dataset," Available: `https://data.mendeley.com/datasets/kcmpdxky7p/1`

Acknowledgement

I have a great pleasure to express my gratitude to all those who have contributed and motivated me during my project work.I am also deeply thankful to Usha Mittal Institute of Technology(UMIT) for providing the necessary resources and a conducive environment for conducting this research. Special appreciation goes to my professors, mentors, and colleagues for their constructive feedback and motivation.

Date: 21/02/2025

Vidhi Agre, Kanchan Bhange, Shruti Ghume