SIGN2SOUND

Real-Time Sign Language Recognition System

Phase 2 – Model Training & System Demonstration

SIGN2SOUND Innovation Challenge

Team Name

SIGN2SOUND_CodeCrew

Project Title

SIGN2SOUND: Real-Time Sign-to-Text Recognition Using Skeletal Hand Landmarks

---

Problem Statement

To design and implement a real-time AI system capable of recognizing sign language gestures using skeletal landmark data and converting them into readable textual output.

Dataset Used

Indian Sign Language Skeletal-point NumPy Array (MediaPipe)
Source: **IEEE DataPort**

Core Technologies

Programming & Frameworks

- Python 3
- PyTorch (LSTM-based sequence modeling)

Computer Vision & Gesture Processing

- MediaPipe (hand landmark extraction)
- OpenCV (real-time webcam interface)

Data Processing & Evaluation

- NumPy
- Scikit-learn (metrics & evaluation)
- Matplotlib (training and evaluation plots)
- pandas (per-class performance analysis)

Speech-to-Text (Auxiliary Module)

- Vosk (offline speech recognition)
- sounddevice (audio capture)

Utilities

- tqdm (training progress visualization)

Submission Phase

Phase 2: Model Training & System Demonstration

Submission Date

January 2026

> *This document presents the design, implementation, training pipeline, and evaluation of the SIGN2SOUND real-time sign language recognition system developed as part of the SIGN2SOUND Phase 2 challenge.*

System Overview (NEXT SECTION)

System Overview

SIGN2SOUND is a real-time sign language recognition system designed to convert hand gestures into readable text using skeletal landmark data. The system operates by capturing live

webcam input, extracting hand landmarks using MediaPipe, processing temporal sequences of landmarks, and performing classification using a deep learning model.

The system follows a modular pipeline consisting of:

1. Video capture and preprocessing
2. Hand landmark extraction
3. Temporal sequence formation
4. Deep learning–based gesture classification
5. Prediction smoothing and visualization

An optional speech-to-text module is integrated to demonstrate bidirectional accessibility, though the core focus of this phase remains sign-to-text recognition.

Dataset Description & Implementation

Dataset Description

The system utilizes the **Indian Sign Language Skeletal-point NumPy Array dataset**, provided through IEEE DataPort as part of the SIGN2SOUND challenge resources. The dataset contains pre-extracted skeletal hand landmarks generated using MediaPipe.

Each sample represents a sign gesture as a temporal sequence of hand keypoints, making it suitable for sequence-based deep learning models such as LSTMs.

Dataset Statistics

- Total samples: 3000
- Number of classes: 25 alphabet-level signs
- Sequence length: 30 frames
- Features per frame: 63 (21 hand landmarks × x, y, z)
- Train / validation split: 80% / 20%

Dataset Integration Strategy

Raw dataset files are not included in the repository due to IEEE DataPort licensing restrictions. The dataset is loaded dynamically using custom preprocessing scripts, ensuring reproducibility while maintaining compliance with licensing requirements.

Preprocessing & Feature Extraction

Preprocessing Pipeline

Each sample in the dataset undergoes the following preprocessing steps:

- Landmark normalization
- Fixed-length sequence alignment (30 frames per sample)
- Conversion to NumPy arrays suitable for PyTorch tensors

No image-based resizing is required, as the dataset consists of skeletal landmark data rather than raw images.

Feature Extraction

Feature extraction is based on **hand skeletal landmarks** obtained via MediaPipe:

- 21 landmarks per hand
- Each landmark represented by (x, y, z) coordinates
- Resulting in 63 features per frame

Only hand landmarks are used in the current implementation. Body pose and facial features are kept modular for future scalability but are not utilized in Phase 2.

Model Architecture

Model Selection

A **Long Short-Term Memory (LSTM)** network was chosen due to its effectiveness in modeling temporal dependencies in sequential data such as sign language gestures.

Architecture Details

- Input size: 63 features per frame
- Sequence length: 30 frames
- LSTM layers: 2
- Hidden units: 256
- Dropout: 0.3
- Output layer: Fully connected classification layer producing class logits

The model outputs a probability distribution over all sign classes.

Training Pipeline & Configuration

Training Environment

- Framework: PyTorch
- Device: CPU
- Batch size: 32
- Epochs: 40
- Optimizer: Adam
- Learning rate: 0.001
- Loss function: CrossEntropyLoss

Training Strategy

The dataset is split into training and validation sets using an 80/20 ratio with randomized shuffling. Model checkpoints are saved based on validation accuracy improvements.

Training metrics including loss and accuracy are logged for each epoch.

Evaluation & Performance Analysis

Evaluation Metrics

The trained model is evaluated on the validation set using:

- Accuracy
- Precision
- Recall
- F1-score

Results Summary

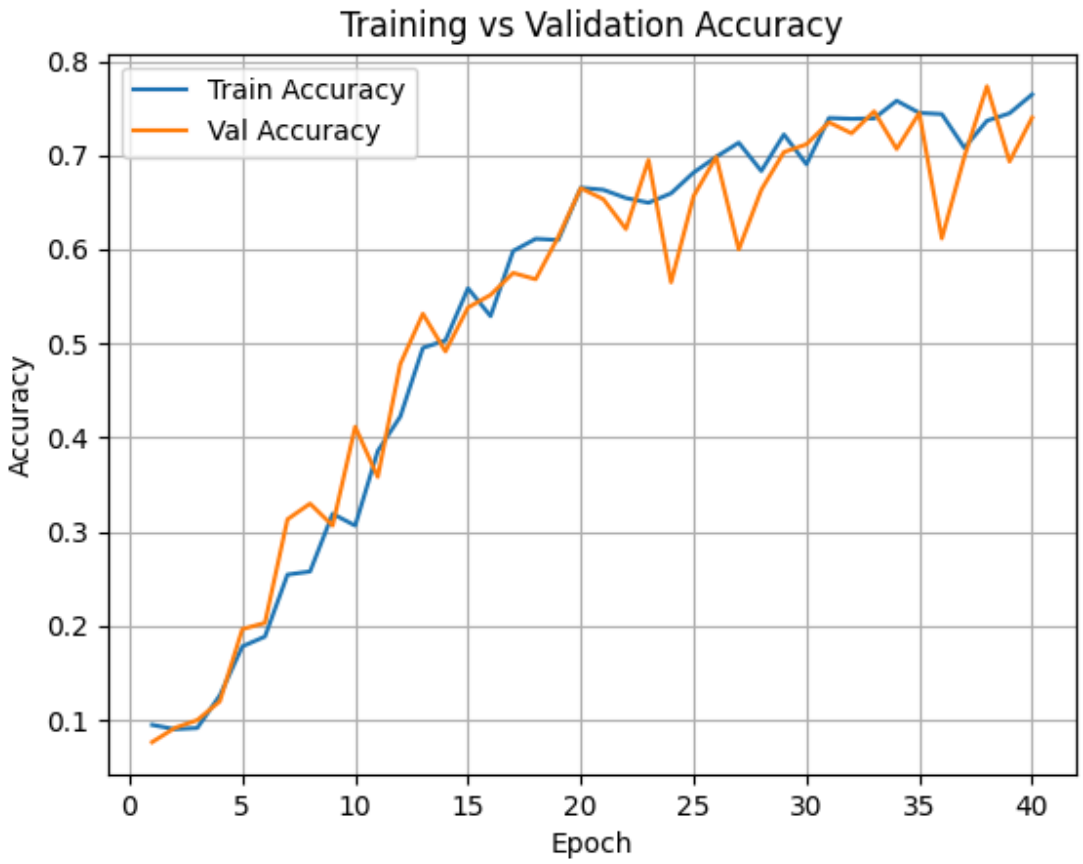
- Accuracy: ~74–77%
- Precision: ~80%
- Recall: ~74–76%
- F1-score: ~72–75%

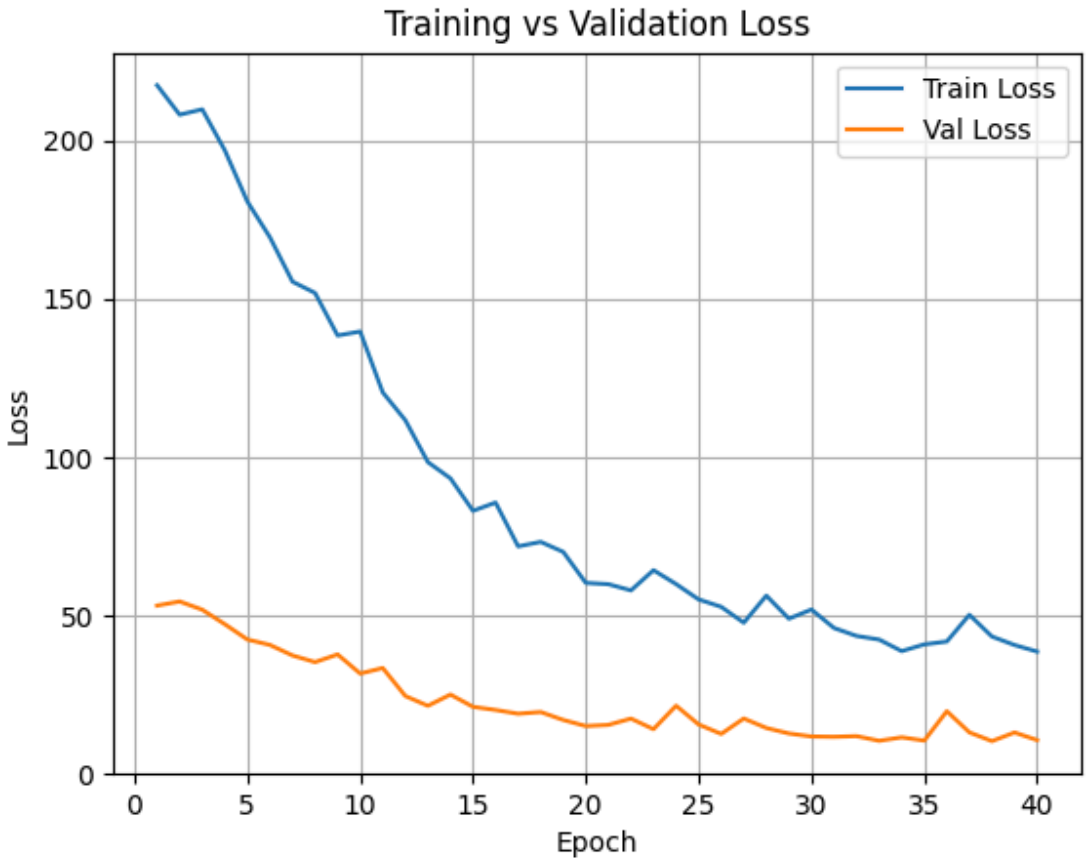
A confusion matrix and per-class performance analysis are generated to identify class-wise behavior and misclassifications.

Training and Validation Curves

Figures below show the training and validation accuracy and loss across epochs.

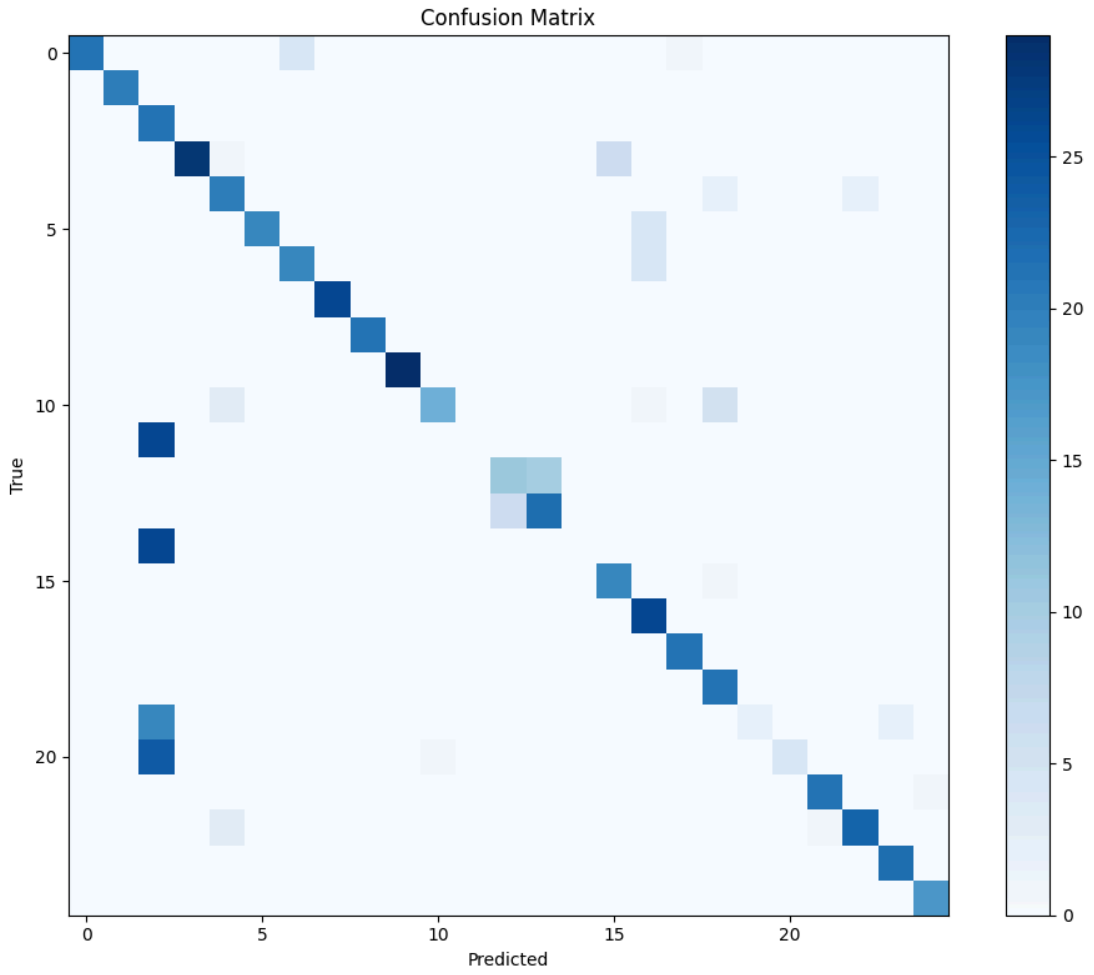
The model converges within a small number of epochs, indicating stable training without overfitting.





Confusion matrix

The confusion matrix illustrates strong diagonal dominance, indicating correct classification for most sign classes. Visually distinct signs such as H, I, J, Q, S, Y, and Z show high true positive rates. Misclassifications are primarily observed among signs with similar hand shapes and orientations (e.g., B, L, O, and V), which share overlapping skeletal landmark patterns. This behavior is expected in alphabet-level sign recognition using hand landmarks alone.



per-class performance summary

Per-class evaluation shows higher accuracy for visually distinct signs, while reduced performance is observed for signs with similar hand shapes and orientations. This confusion is expected due to overlapping skeletal landmark patterns in alphabet-level gestures.

Inference speed

Real-time inference is achieved on a CPU-only system using skeletal landmarks and fixed-length sequences. The system processes frames continuously with minimal latency suitable for live demonstration.

Real-Time System Demonstration

Live Inference Pipeline

The real-time demo operates as follows:

1. Webcam captures live video
2. MediaPipe extracts hand landmarks per frame
3. Landmarks are accumulated into sequences
4. The trained LSTM model predicts the sign
5. Temporal smoothing stabilizes predictions
6. Results are displayed via an OpenCV GUI

Prediction Smoothing

A sliding window–based majority voting mechanism is used to reduce prediction flicker and ensure stable outputs.

Limitations & Future Work

Current Limitations

- Limited to alphabet-level signs
- Uses hand landmarks only
- Performance dependent on lighting and hand visibility

Future Improvements

- Word- and sentence-level recognition
- Integration of body pose and facial expressions
- Transformer-based temporal modeling
- Enhanced multilingual support

Conclusion

SIGN2SOUND successfully demonstrates a real-time sign language recognition system using skeletal landmark data and deep learning. The system meets the objectives of Phase 2 by providing a working model, live demonstration, and comprehensive evaluation, while maintaining modularity for future expansion.

Reference

[1] IEEE DataPort, "Indian Sign Language Skeletal-point Dataset (MediaPipe)."

[2] Hochreiter, S., & Schmidhuber, J. (1997).
Long Short-Term Memory. Neural Computation.

[3] MediaPipe: A Framework for Building Perception Pipelines.
Google Research.

[4] PyTorch Documentation – https://pytorch.org