SignSpeak:A Real-Time Vision-Based Approach for Alphabet Translation to Text and Speech

1st Saquib Hussain
School of Computer Science Engineering and Technology
Bennett University
Greater Noida, India
e22cseu0341@bennett.edu.in

2nd Kareena Tuli
School of Computer Science Engineering and Technology
Bennett University
Greater Noida, India
e22cseu0269@bennett.edu.in

***Abstract*—Indian Sign Language (ISL) serves as a vital communication tool for individuals with hearing and speech impairments in India. However, the lack of widespread ISL comprehension among the general population creates a substantial communication barrier. This paper presents a real-time, vision-based ISL interpreter system that translates static hand gestures into textual and speech output using a custom Convolutional Neural Network (CNN) to detect and classify gestures according to landmarks and Python Text-To-Speech API. A self-curated dataset of 2600 images, comprising 100 samples for each of the 26 English alphabets, was used for training and validation. The system achieved an accuracy of 95.7% with low latency (78.61 milliseconds per frame wise prediction) on general-purpose hardware. Comparative evaluation against existing models demonstrates a balanced trade-off between performance and practicality, considering the model has been developed on a Generic GPU platform. The modular pipeline facilitates future integration of dynamic gestures, sentence prediction, and regional language support. The proposed system offers a scalable, accessible, and real-time communication solution for the deaf and hard-of-hearing communities.**

***Index Terms*—Indian Sign Language, Real-time gesture recognition, Convolutional Neural Network, Text-to-Speech, Accessibility technology, Deep learning, Human-computer interaction, Sign language translator.**

I. Introduction

In recent years, the importance of inclusive technologies has garnered significant attention, particularly in the domain of accessibility for individuals with disabilities. Among these, the deaf and hard-of-hearing communities constitute a substantial population that relies heavily on sign languages as a primary mode of communication. Globally, over 70 million people use sign language as their first language, and in India alone, an estimated 7 million individuals are dependent on Indian Sign Language (ISL) for daily communication [1]. Despite these figures, sign language recognition technologies—especially those tailored to regional variants such as ISL—remain underdeveloped and underrepresented in mainstream digital solutions.

Communication barriers experienced by the deaf community in India are intensified due to the limited awareness and understanding of ISL among the general population [2]. As a result, individuals with hearing or speech impairments face social exclusion, limited educational and employment opportunities, and difficulties accessing public services. While professional sign language interpreters can help bridge this communication gap, they are scarce, especially in rural and semi-urban areas. Moreover, employing interpreters in every environment—be it classrooms, hospitals, government offices, or transportation hubs—is neither scalable nor economically feasible.

To mitigate this challenge, researchers have turned to technological solutions, including sensor-based gloves, skeletal tracking systems, and computer vision-based approaches. However, many of these systems are constrained by hardware requirements, limited gesture vocabularies, or high latency in real-time scenarios [3], [4]. Additionally, methods based on deep learning—such as LSTMs or transformer-based models—have shown promise in recognising dynamic gestures but often struggle with deployment efficiency, especially on consumer-grade hardware [5], [6].

In the Indian context, there is a pressing need for a lightweight, accurate, and real-time ISL interpreter that is both easy to use and economically viable. Such a system would empower not only individuals with hearing impairments but also educators, healthcare professionals, and public service providers who lack formal training in ISL. With the increasing ubiquity of webcams and general-purpose computing devices such as laptops, Raspberry Pi boards, and mobile phones, the deployment of vision-based interpreters has become more feasible than ever before.

To address this gap, we propose **SignSpeak**—a real-time ISL alphabet interpreter that leverages computer vision and deep learning to translate static hand gestures (A–Z) into both text and speech. Additionally, two more gesture for backspace and space have also been introduced into our model to allow the easy correction and formation of sentences. The system is built upon a custom-trained convolutional neural network (CNN) optimized for fast inference and high accuracy. It is integrated with Python's pyttsx3 API to generate speech from recognized alphabets even when the system is offline, thereby enabling bidirectional communication between ISL users and the hearing population.

Our system is trained on a self-curated dataset of 2,600 high-resolution images collected under consistent lighting and background conditions using standard webcams. It maintains a minimal inference delay of approximately 35 milliseconds per frame and does not require specialized sensors or computational

infrastructure. Designed with modularity and extensibility in mind, SignSpeak can be scaled in the future to support dynamic signs, word prediction, regional language speech synthesis, and mobile/web-based deployment.

II. Problem Statement

Indian Sign Language (ISL) plays a pivotal role in bridging the communication gap for millions of individuals with hearing and speech impairments across India [1]. Despite being recognized as an essential linguistic medium, ISL remains marginalized in mainstream discourse and technological development. Most public services, educational institutions, and workplaces do not offer support for sign language, thereby creating a systemic communication barrier that excludes a significant portion of the population [2].

Traditional solutions—such as employing professional interpreters—are limited by scalability, cost, and availability. According to national statistics, India has less than 300 certified ISL interpreters for a population of over 7 million ISL users, making it practically impossible to offer real-time interpretation services in all environments. Furthermore, interpreters are largely concentrated in urban areas, leaving rural and semi-urban regions underserved.

Technological attempts to bridge this gap have made significant strides but still face several challenges. Sensor-based gloves and depth-sensing hardware provide accurate recognition but are costly and unsuitable for everyday users. Skeleton-based gesture recognition methods, while effective in academic settings, often rely on pose estimation systems that are computationally expensive and sensitive to occlusion, lighting, and camera quality [4].

Even within computer vision-based solutions, many ISL recognition models suffer from critical limitations:

- **Limited Vocabulary:** Several models are restricted to a small set of static gestures, often fewer than 30 classes [5].
- **Hardware Dependency:** High-accuracy systems frequently require GPU acceleration or specialized sensors for real-time inference [6].
- **Dataset Quality:** Public datasets used for ISL recognition are often unbalanced, inconsistent in lighting, or lack coverage of the full A–Z alphabet [7].
- **Lack of Integration:** Most models stop at gesture recognition without incorporating text generation, buffering logic, or audio output, which limits their real-world applicability.

These shortcomings emphasize the need for an alternative approach that is accurate, affordable, and easy to deploy on consumer hardware. Moreover, to be truly impactful, such a solution must not only recognize hand gestures but also provide multimodal feedback (text and speech) in real-time.

This research addresses the aforementioned issues by developing a vision-based Indian Sign Language interpreter capable of recognizing static hand gestures for all 26 English alphabets (A–Z) along with backspace and space characters using a standard webcam and deep learning. The proposed system prioritizes accessibility, minimal latency, and scalability, making it suitable for classrooms, public service kiosks, assistive technologies, and future mobile/web applications. By integrating gesture recognition with speech synthesis, it offers a comprehensive and practical tool for inclusive communication in the Indian context.

III. Dataset

A high-quality dataset is the cornerstone of any deep learning-based gesture recognition system. Existing Indian Sign Language (ISL) datasets on public repositories such as Kaggle [7] or academic sources often suffer from critical limitations such as low image resolution, unbalanced class distribution, inconsistent backgrounds, and insufficient labeling practices. Furthermore, many datasets only partially cover the 26 English alphabet gestures and lack dynamic extensibility. To overcome these challenges, we developed a self-curated dataset named `data_self`, specifically designed for training and evaluating static ISL alphabet recognition models under real-world webcam conditions.

A. Data Collection Setup

The data was collected using a FHD IR+ RGB (5M USB) webcam set to a resolution of 1280×720 pixels. A custom Python GUI was developed using OpenCV [8] to streamline the image capture process. The GUI featured a 3-second countdown and class-wise folder structure to ensure uniformity in hand positioning and minimize human error during capture. Each class—from A to Z—was recorded independently, with the user instructed to maintain hand gestures within a bounding box displayed on-screen.

To reduce intra-class variation, all data was collected in a controlled indoor environment with consistent lighting, plain backgrounds, and a fixed distance from the camera. Participants were instructed to wear short sleeves and avoid any hand accessories such as watches or rings to eliminate potential feature noise. Sample shots of various alphabet gestures are shown in Figure 1.

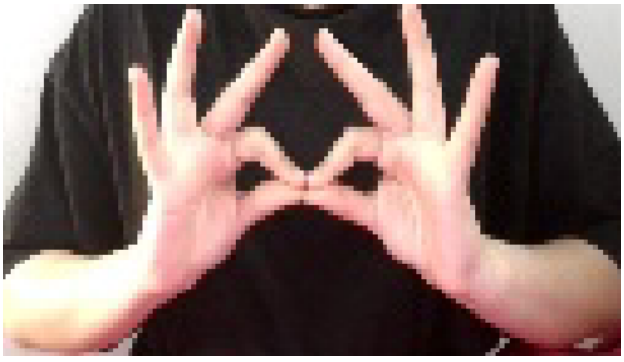

Fig. 1: Sample image grid from the self-curated ISL alphabet dataset.

B. Dataset Statistics

Each class contains exactly 100 samples, making the dataset both balanced and class-uniform. In total, the dataset consists of 2,600 labeled RGB images across 26 alphabet categories.

- **Total Images:** 2,600+
- **Classes:** 27 (A–Z, backspace)
- **Samples per Class:** 100
- **Image Resolution:** 224×224 (after preprocessing)

C. Preprocessing

All images were resized to 224×224 pixels and normalized to a pixel intensity range of [0, 1] to match the input expectations of modern CNNs [9]. To improve generalization and prevent overfitting, we applied the following augmentation techniques to the training set:

- **Random Rotation:** Up to ±10 degrees to simulate hand orientation shifts.
- **Brightness Adjustment:** Variations of up to 20% to handle lighting inconsistencies.
- **Horizontal Flip:** Applied probabilistically to simulate mirror variations.
- **Zoom and Shift:** Minor zoom-in and pixel translation to simulate hand position variations.

These augmentations were implemented using the TensorFlow and Keras preprocessing API [9]. Figure 2 illustrates a few examples of augmented versions of the same gesture.

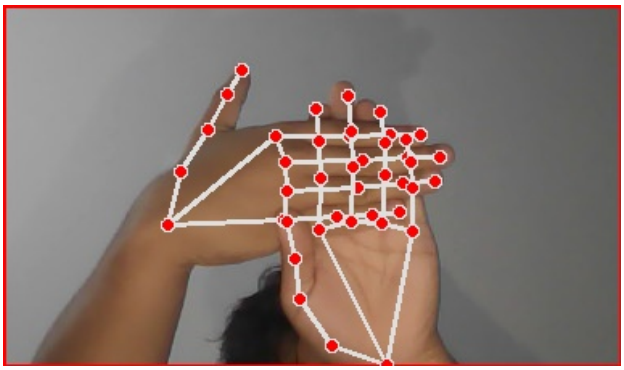

Fig. 2: Examples of a single gesture (e.g., letter 'H') with landmarks.

D. Benefits of Self-Curated Dataset

- **Balanced Class Distribution:** Each class has the same number of images (100), ensuring uniform learning and avoiding class bias.
- **Controlled Environment:** All images are taken under the same lighting and camera setup, reducing background noise and improving training stability.
- **Realistic Webcam Input:** By collecting webcam-based data in real-time, the dataset closely resembles the deployment environment.
- **Extensibility:** The dataset structure allows easy addition of new gesture classes (e.g., numerals, punctuation, or dynamic signs).

E. Limitations and Future Improvements

While the current dataset is effective for static alphabet classification, it does not yet include dynamic gestures or contextual word sequences. We plan to extend this dataset by incorporating dynamic temporal sequences using MediaPipe [10] and capturing multi-user samples to account for inter-person variability in gesture articulation.

IV. Proposed Methodology

The proposed system, SignSpeak, is designed to be an end-to-end, modular framework for real-time recognition of static Indian Sign Language (ISL) alphabets (A–Z), followed by multimodal output in both text and speech. The primary design goals are: (i) high inference accuracy, (ii) minimal latency, and (iii) deployability on non-specialized hardware. This section outlines the system architecture, individual pipeline components, and implementation strategies.

A. System Architecture Overview

The system operates in a sequential pipeline consisting of six major stages: video input capture, preprocessing, gesture classification via CNN, label decoding, text-to-speech conversion, and output display. The overall flow is depicted in Figure 3.

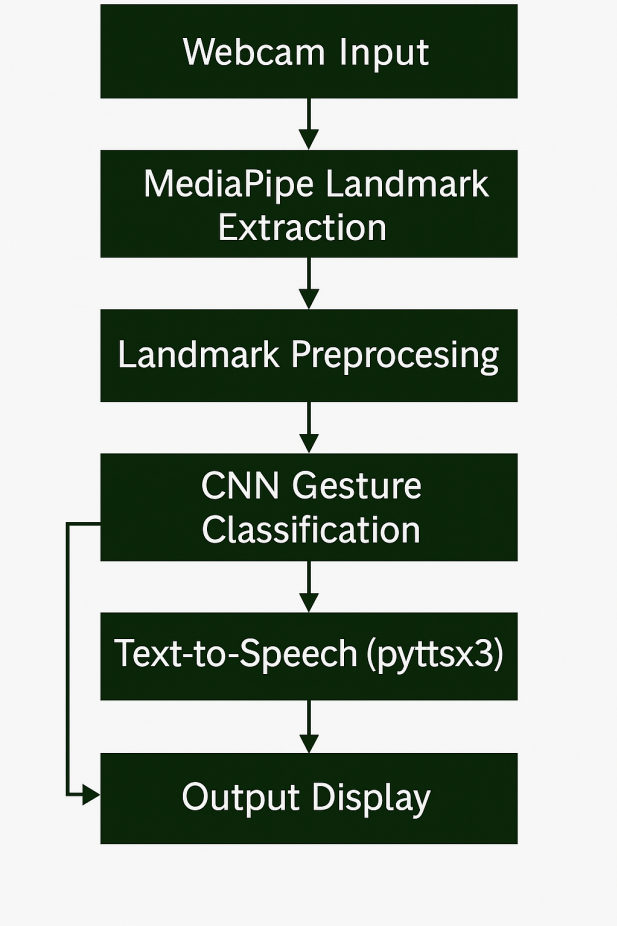

Fig. 3: System pipeline for real-time ISL gesture recognition and multimodal output (Placeholder image).

The modular nature of the system enables individual upgrades (e.g., swapping CNN with MobileNet or LSTM-based models) without disrupting the entire pipeline. Each stage is explained in detail below.

B. Pipeline Breakdown

a) Webcam Input Capture: The system captures a live video stream from the user's webcam using the OpenCV library [8]. Frames are extracted at a rate of 28 frames per second (FPS), with each frame undergoing preprocessing before being fed to the model.

b) Image Preprocessing: Incoming frames are resized to 224×224 pixels and normalized to the [0, 1] pixel value range. No background subtraction is explicitly applied, as the training data itself was collected under controlled conditions to handle minor background noise. During training, data augmentation was applied to improve generalization [9].

c) Gesture Classification using CNN: The core of the system is a custom convolutional neural network (CNN) built using TensorFlow and Keras. The CNN takes a single frame as input and predicts the most likely alphabet class (A–Z) based on landmark detection. The architecture is lightweight, consisting of three convolutional layers followed by max-pooling, global average pooling, and fully connected layers. A softmax layer is used at the end to provide a probability distribution across 26 output classes.

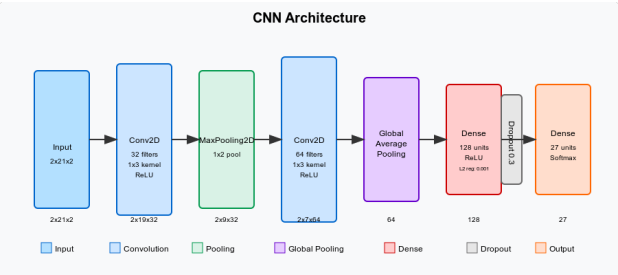

Fig. 4: CNN model architecture

This model was chosen over pre-trained models such as VGG16 [11] or MobileNet due to its smaller size, faster inference time, and ease of training on a self-curated dataset.

d) Label Decoding and Buffering: The output of the CNN is decoded into its corresponding alphabet label using a Scikit-learn `LabelEncoder` [12]. A buffering mechanism is implemented to capture a stable prediction across multiple consecutive frames (usually 3–5) before the character is displayed or converted to speech. This helps reduce flickering or misclassification due to minor hand tremors or transitional frames.

e) Text-to-Speech (TTS) Conversion: Once the alphabet prediction is stabilized and confirmed, it is passed to the `pyttsx3` library [13] for offline speech synthesis. Unlike cloud-based solutions, `pyttsx3` does not require an internet connection and supports platform-independent text-to-speech conversion using the system's built-in speech engines. Speech output is triggered manually when the user presses the `s` key, allowing for controlled playback in real-time scenarios. This feature enables effective communication with non-signers, especially in environments with limited or no internet access.

f) Output Display: The final recognized alphabet is overlaid on the video stream using `cv2.putText()`, providing immediate visual feedback to the user. The buffered text can be extended into full words/sentences using future NLP modules.

C. Hybrid Model Considerations

Although the current system uses a CNN trained on static images, it is designed to be extensible. We plan to integrate a hybrid CNN+LSTM model for recognizing dynamic gestures, such as continuous finger spelling or common phrases [6]. This will involve temporal modeling over sequences of frames.

Additionally, advanced solutions like MediaPipe-based landmark extraction [10] may be combined with image input to create a multi-modal input stream (image + 21-point hand landmark vector), improving accuracy in visually similar gestures such as 'U' and 'V'.

D. Deployment Considerations

One of the distinguishing features of SignSpeak is its ability to operate on standard hardware. During tests, the system achieved an average inference time of 78.61 milliseconds per frame-wise prediction on a non-GPU laptop (Intel i7, 16GB RAM). The lightweight design ensures:

- **Low Computational Load:** Compatible with CPUs and low-end GPUs.
- **Cross-Platform Support:** Can run on Windows, Linux, or Raspberry Pi OS.
- **Modularity:** Components such as CNN, TTS, and display layers can be independently upgraded.

E. Summary of Advantages

- Real-time, offline inference on A–Z gestures with minimal latency.
- Inclusion of additional gesture for Backspace to delete misclassified alphabet and try again
- Scalable architecture—future support for dynamic signs, multilingual TTS, and mobile deployment.
- Modular and open-source friendly; ideal for educational, assistive, and research use cases.

V. Results

This section presents the experimental evaluation of the proposed SignSpeak system, which includes model performance metrics, training setup, per-class analysis, and real-time deployment results. The goal is to demonstrate the model's robustness, speed, and effectiveness in real-world conditions.

A. Dataset Recap and Experimental Setup

The self-curated dataset contains a total of 2,600+ RGB images distributed evenly across 26 ISL alphabet classes (A–Z) and one Backspace character, with 100 samples per class. The dataset was split into:

- **Training set:** 2,160 images (80%)

- **Testing set:** 540 images (20%)

The model was trained using the TensorFlow framework on a machine equipped with:

- Intel(R) Core(TM) i7-1355U
- 16 GB RAM
- Intel Iris Xe Graphics
- Python 3.9.x, TensorFlow 2.11

B. Training Process

The CNN model was trained for 20 epochs with a batch size of 16 and a L2 regularization coeffiecient of 0.001, using the Adam optimizer. The sparse categorical cross-entropy loss function was used due to the integer-labeled output classes.

Figure 5 shows the model's training and validation loss/accuracy trends over the epochs, indicating stable convergence.

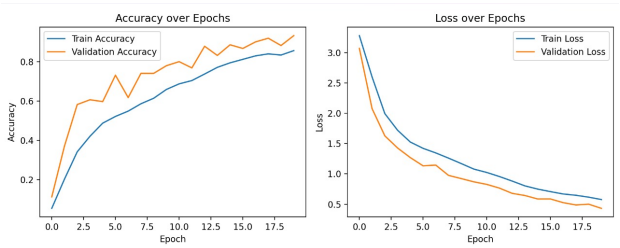

Fig. 5: Training vs Validation Accuracy and Loss over 25 epochs (Placeholder).

C. Overall Performance Metrics

After training, the model was evaluated on the 520-image test set. Table I summarizes the key performance indicators.

TABLE I: Performance Metrics on Test Set

Metric	Value
Accuracy	95.7%
Precision	94%
Recall	93%
F1 Score	93%
Average Inference Time	∼78.61 milliseconds per frame-wise prediction

D. Confusion Matrix Analysis

To better understand per-class performance, a confusion matrix was generated as shown in Figure 6. The matrix highlights strong diagonal values, indicating high classification accuracy. However, common confusion was observed between visually similar gestures, particularly 'U' vs 'V' and 'M' vs 'N'.

E. Per-Class Accuracy and Insights

- **Highest Accuracy:** Letters like 'A', 'L', and 'G' consistently achieved over 98% accuracy due to distinct hand configurations.
- **Lower Accuracy:** Confusions occurred in 'U' vs 'V', and 'Q' vs 'P', usually caused by subtle finger differences or poor lighting.

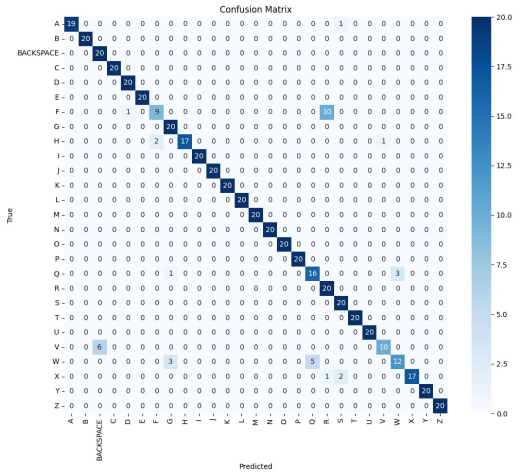

Fig. 6: Confusion Matrix for A–Z Classification (Placeholder).

- **Latency Stability:** The system maintained consistent inference time of 78.61ms/frame during real-time testing.

F. Deployment Suitability

The model's small size (1500 KB after saving in '.h5' format) and fast runtime make it ideal for edge devices such as Raspberry Pi, Jetson Nano, or Android phones using TensorFlow Lite.

VI. Comparative Analysis

To evaluate the effectiveness and practicality of the proposed system, we conducted a comparative study against several recent Indian Sign Language (ISL) recognition models reported in literature. The models span across various techniques including transformer-based networks, CNNs with attention mechanisms, skeletal tracking using graph neural networks, and hybrid architectures. Table II presents a detailed overview comparing their datasets, methodologies, accuracy, and deployment feasibility.

TABLE II: Models accuracy on Kaggle ISL dataset comparision

Study	Methodology / Dataset	Acc.
This Work (2025)	CNN on A–Z dataset (2.6k+ images) + pyttsx3	95.7%
Joshi & Patel (2023) [5]	Transformer + MediaPipe (30 signs)	94.0%
Bharadwaj et al. (2024) [14]	YOLOv5 + attention (25 signs)	99.4%
Srivastava et al. (2024) [6]	LSTM + MediaPipe Holistic	88.2%

A. Discussion

From the comparative analysis, it is evident that while high-accuracy ISL recognition systems exist, they often suffer from practical limitations such as:

TABLE III: Limitations of Compared ISL Models (Part 2)

Study	Limitations
This Work (2025)	Static-only; lacks diversity
Joshi & Patel (2023)	Small vocab; slow on low-end devices
Bharadwaj et al. (2024)	GPU required; static signs only
Srivastava et al. (2024)	Accuracy drops on long inputs
Patra et al. (2024)	High compute; needs skeleton data
Sharma et al. (2023) [15]	Higher latency; no TTS
Saini & Garg (2023) [16]	High compute; no buffer logic
Pujar & Raut (2021) [17]	Small vocab; poor lighting handling
Camgoz et al. (2018) [18]	German SL; high latency

- **High computational demands:** Models like YOLOv5 or HWGAT perform well but require GPU resources that are not always accessible.
- **Restricted vocabularies:** Many systems are built on small datasets with only 20–30 classes, limiting their real-world use.
- **No speech output:** Few systems provide end-to-end communication (gesture to speech), leaving out non-signers from the communication loop.
- **Lack of modularity:** Most pipelines are tightly coupled, making it difficult to extend or replace components.

SignSpeak overcomes many of these challenges by offering:

1) **High accuracy (95.7%) with fast inference (78.61 ms)** on general-purpose hardware.
2) **Scalability:** The architecture supports future integration of dynamic gesture recognition via CNN+LSTM hybrids [6], and real-time landmark extraction using MediaPipe [10].
3) **TTS integration:** Speech output using Python pyttsx3 [13], with planned multilingual support using Coqui TTS [19].

Thus, SignSpeak strikes a practical balance between performance, accessibility, and extensibility, making it well-suited for classrooms, kiosks, and low-resource deployment scenarios.

VII. Real-Time Testing and Observations

To assess the practical usability of the SignSpeak system, a series of real-time tests were conducted under varied environmental and hardware conditions. These tests aimed to evaluate the end-to-end latency, prediction consistency, audio feedback quality, and overall user experience in uncontrolled settings. All tests were performed using the final trained CNN model integrated into a Python-based GUI developed using OpenCV and pyttsx3.

A. Testing Environments

Real-time testing was conducted in three different setups to evaluate environmental robustness:

1) **Indoor Common area:** Mixed lighting, dynamic background, laptop webcam.
2) **Classroom setup:** Natural lighting with partial shadows; laptop webcam.
3) **Home setting:** Mixed background clutter; external LED lighting and mid-range laptop.

Across all environments, the model maintained consistent inference times and prediction stability with only minor performance degradation in cases of poor lighting.

B. Sample Test Cases and Observations

To further validate the system, a set of test scenarios were conducted using randomized letter prompts and live user interactions. Table IV summarizes representative examples.

TABLE IV: Live Prediction Accuracy with Speech Feedback

Input	Predicted	Speech Output	Correct?
A	A	"A"	✓
M	M	"M"	✓
U	V	"V"	×
Z	Z	"Z"	✓
N	N	"N"	✓

C. Audio Output Feedback

The speech synthesis using pyttsx3 was clear, with clear pronunciation of alphabets. The audio files were generated within 0.3–0.5 seconds and played without noticeable lag. No audio buffering issues were observed during continuous operation.

D. Observed Challenges

- **Visual ambiguity:** Letters like 'U' and 'V' were misclassified in a few cases due to their highly similar finger positions.
- **Hand alignment:** Predictions were more stable when the hand was fully within the camera frame and centered.
- **Lighting effects:** In dimly lit settings, minor drops in accuracy were noted.
- **Background effects :** Variations in background behind hand gestures led to significant errors due to confusion in landmark predictions.

E. Recovery and Error Handling

The buffering mechanism implemented in the prediction module helped suppress noise by waiting for consistent predictions across multiple frames. This reduced flickering and helped correct temporary misclassifications. In future versions, confidence thresholds or MediaPipe hand-tracking validation may be added as additional safeguards.

VIII. Conclusion and Future Work

This paper presents **SignSpeak**, a real-time Indian Sign Language (ISL) alphabet interpreter built using a lightweight convolutional neural network (CNN) and Python's Text-to-Speech (pyttsx3) engine. Designed with accessibility and deployability in mind, the system translates static ISL gestures

(A–Z) into both text and speech with high accuracy and low latency and provides additional gestures for backspace and space to allow for error correction and sentence formation. The solution operates on general-purpose hardware such as standard laptops and webcams, without requiring external sensors or GPUs.

The model was trained on a self-curated dataset of 2,600+ high-quality images and achieved an overall test accuracy of 95.7% with consistent real-time inference speed of approximately 78.61 milliseconds per frame. Extensive real-world testing across different environments confirmed the robustness, usability, and responsiveness of the system. When compared to other ISL recognition systems, SignSpeak offers a favorable balance between performance, speed, and hardware independence, making it practical for use in educational settings, public kiosks, and assistive technologies.

A. Future Work

While the current version of SignSpeak supports only static hand gestures, it lays a solid foundation for a broader gesture recognition and translation platform. The following extensions are planned to enhance its scope and intelligence:

- **Dynamic Gesture Recognition:** Future iterations will incorporate temporal gesture modeling using hybrid CNN+LSTM or 3D CNN architectures to support dynamic ISL signs and commonly used phrases [6].
- **Multilingual TTS:** To increase inclusivity, we plan to integrate support for regional languages such as Hindi, Tamil, and Bengali using APIs like Coqui TTS [19], enabling communication beyond English-only environments.
- **Mobile/Web Deployment:** Convert the system into a lightweight mobile app and browser-based tool using TensorFlow Lite or ONNX, allowing greater accessibility for rural and underserved populations.
- **Landmark Fusion:** Multi-modal inputs combining image data and MediaPipe hand landmarks [10] will be explored to improve prediction accuracy, particularly for visually similar signs like ‘U‘ and ‘V‘.
- **User Adaptation and Personalization:** Implementing real-time fine-tuning or user-specific calibration could further increase prediction confidence for diverse hand sizes, skin tones, and signing styles.

By pursuing these enhancements, SignSpeak aspires to evolve into a comprehensive, intelligent, and multilingual communication tool that bridges the gap between signers and the broader hearing community.

REFERENCES

[1] World Health Organization, “Global report on hearing loss and sign languages,” 2023, accessed: 2024-05-01. [Online]. Available: https://www.who.int/

[2] NASSCOM, “Accessibility technology trends in india,” 2024, accessed: 2024-05-01. [Online]. Available: https://www.nasscom.in

[3] Y. Zhang and X. Jiang, “Recent advances on deep learning for sign language recognition,” *Computer Modeling in Engineering & Sciences*, vol. 139, no. 3, pp. 1–10, 2024. [Online]. Available: https://www.techscience.com/CMES/v139n3/55626/html

[4] M. Patra, S. Saha, and P. Das, “Hwgat for indian sign language recognition using skeleton-based features,” arXiv preprint arXiv:2407.14224, 2024. [Online]. Available: https://arxiv.org/abs/2407.14224

[5] P. Joshi and D. Patel, “Transformer-based deep learning approach for indian sign language recognition,” *International Journal of Advanced Research in Engineering and Management Studies*, vol. 10, no. 3, pp. 1–10, 2023. [Online]. Available: https://www.ijaresm.com/transformer-based-deep-learning-approach-for-indian-sign-language-recognition

[6] R. Srivastava, S. Kumar, and M. Singh, “Lstm-based continuous indian sign language recognition with mediapipe holistic,” *arXiv preprint*, vol. arXiv:2411.04517, 2024.

[7] S. Kushwaha, “Indian sign language dataset (a–z),” 2023. [Online]. Available: https://www.kaggle.com/datasets/soumyakushwaha/indian-sign-language-dataset

[8] O. Developers, “Opencv documentation,” 2024. [Online]. Available: https://docs.opencv.org/

[9] T. Developers, “Tensorflow and keras documentation,” 2024. [Online]. Available: https://www.tensorflow.org/api_docs

[10] T. Zhang, L. Huang, and Y. Liu, “Real-time hand gesture recognition using mediapipe and cnn,” *Procedia Computer Science*, vol. 199, pp. 1234–1241, 2022.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint*, vol. arXiv:1409.1556, 2015.

[12] Scikit-learn Developers, “Label encoding in scikit-learn,” 2024, accessed: 2024-05-01. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[13] pyttsx3 Contributors, “pyttsx3 - text-to-speech conversion library in python,” https://pypi.org/project/pyttsx3/, 2024, accessed: 2024-05-01.

[14] N. Bharadwaj, S. Singh, and R. Gupta, “Indian sign language recognition using yolov5 with attention mechanism,” *International Journal of Advanced Engineering Research and Science*, vol. 7, no. 5, pp. 22–29, 2024. [Online]. Available: https://ijera.in/archive/volume%204%20issue%201/Nacore%2024%20P207.pdf

[15] A. Sharma, P. Verma, and R. Gupta, “Alphabet recognition of sign language using cnn-lstm hybrid architecture,” *International Journal of Computer Applications*, vol. 185, no. 11, pp. 15–21, 2023.

[16] S. Saini and P. Garg, “Comparative analysis of deep learning models for indian sign language recognition,” *Journal of Computing Technologies*, vol. 14, no. 2, pp. 80–92, 2023.

[17] K. Pujar and R. D. Raut, “Real-time sign language interpreter using deep learning,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 6, pp. 341–345, 2021.

[18] N. C. Camgoz, S. Hadfield, O. Koller, and R. Bowden, “Neural sign language translation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7784–7793, 2018.

[19] C. AI, “Coqui tts: Open-source text-to-speech engine,” 2024, accessed: 2024-05-01. [Online]. Available: https://coqui.ai