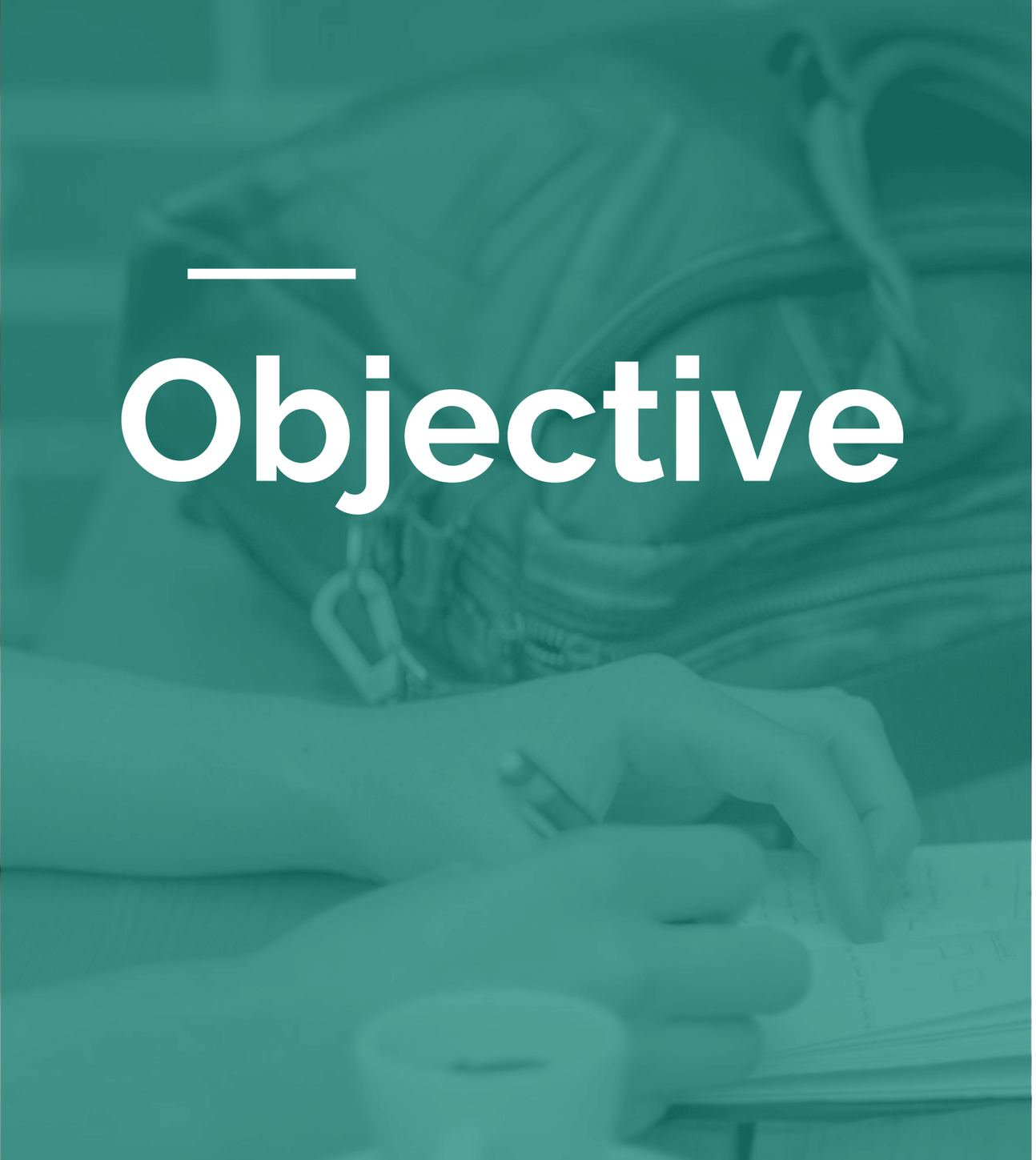
INDIAN SIGN LANGUAGE RECOGNITION SYSTEM

Under the esteemed guidance of
Ritesh Prasad

Prepared by Group: 9
Sattam Bhattacharyya(12230621028)
Aahona Sinha Roy (12230621054)
Aditya Ghosh (12230621042)
Chandrani Das Sharma (12230621038)



Objective

The main aim of the project is to develop a sign language recognition system using CNN.

Key goals include:

- **CNN-based system for static ISL gesture classification.**
- **Image preprocessing improves input quality.**
- **Cost-effective, real-time solution using a regular webcam.**
- **Expanding communication for the DHH (Deaf and Hard of Hearing) community in India.**

LIBRARIES USED

The system is built using Python and a collection of modern deep learning and computer vision libraries.

Tools and Libraries Used:

- Python + PIP (v25.1.1)
- TensorFlow & Keras
- OpenCV (v4.11.0)
- NumPy & Pandas
- Matplotlib & Seaborn
- Streamlit

Concept and Role of Gesture Recognition

- Sign language detection apps use computer vision & AI to recognize gestures.
- Converts hand movements into text/speech in real time.
- Bridges the communication gap between deaf individuals & the hearing community.
- Real-life impact: Enables seamless interaction, e.g., ordering food at a restaurant.

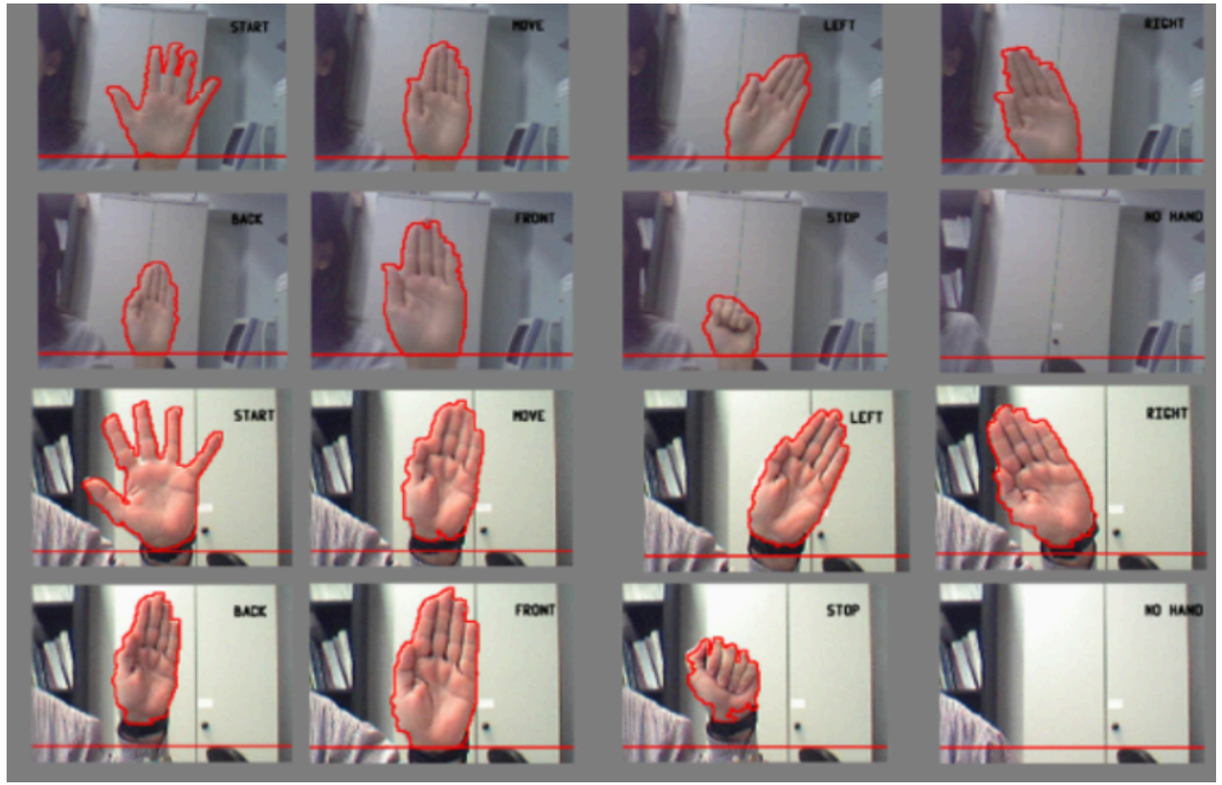


Fig 2: Hand Gesture Recognition

Dataset Generation Process

- Dataset Creation: Existing datasets contained only numerical RGB values, lacking actual images. No dataset matched format & model needs.

- Data Collection: Used OpenCV & webcam to capture hand gestures. Created 800 training & 200 testing images per ASL symbol.

- Region of Interest (ROI): Defined using a blue rectangle to isolate the hand in each frame.

- Image Processing:

Converted ROI to grayscale.

Applied Gaussian Blur (5×5, σ = 2) to reduce noise & enhance gesture features for better thresholding.

Fig 3: Data Acquisition using OpenCV

Data Acquisition and Preprocessing:

.Vision-Based Approach for data acquisition:

- Captures hand gestures using a standard camera.
- Enables natural interaction without extra hardware.

Data Preprocessing:

- Gaussian Blur (OpenCV): Applied for noise reduction & smoother images before training.
- Color Segmentation Issues: Inconsistent due to lighting, failed with similar symbols (‘M’ vs. ‘N’).
- Uniform Background: Used stable single-color background to enhance accuracy, removing need for skin-color segmentation.
- Data Augmentation: image Rescale, rotate, mirror image, brightness adjusting, sheer range, shifting etc.

Fig 4 : Example of a preprocessed image

Activation, Training, and Optimization

Training Details and Performance -

- **Activation Function:** ReLU speeds up training and prevents the vanishing gradient issue.
- **Pooling Strategy:** Max Pooling (2×2) reduces computation and improves generalization.
- **Dropout Layer:** Prevents overfitting by randomly deactivating neurons during training.
- **Optimizer:** Adam Optimizer (combines AdaGrad & RMSProp) for efficient learning.

Training Process:

- **Images grayscale → blurred → thresholded → resized (128×128).**
- Model trained with labeled data; Softmax normalizes predictions.
- **Cross-entropy** loss minimized using Gradient Descent

CNN Architecture

Layer Type	Details	Input Shape (Example)	Output Shape (Approx.)	Parameters/Notes
Input Layer	Image Input	-	128x128x3	RGB Image
Convolutional Layer 1	32 filters, 3x3 kernel size	128x128x3	126x126x32	
Max Pooling Layer 1	2x2 pool size, default stride 2	126x126x32	63x63x32	Downsamples feature maps
Convolutional Layer 2	32 filters, 3x3 kernel size	63x63x32	60x60x32	
Max Pooling Layer 2	2x2 pool size, default stride 2	60x60x32	30x30x32	Downsamples feature maps
Flatten Layer	Converts 3D feature maps to 1D vector	30x30x32	28,800	30 * 30 * 32 = 28,800 values
Dense Layer 1	Fully connected layer	28,800	128	128 neurons, includes Dropout (0.5)
Dense Layer 2	Fully connected layer	128	96	96 neurons
Output Layer	Softmax activation	96	29	29 classes, for classification
Regularization		-	-	Batch Normalization, Dropout, L2

Table 1 : CNN configuration

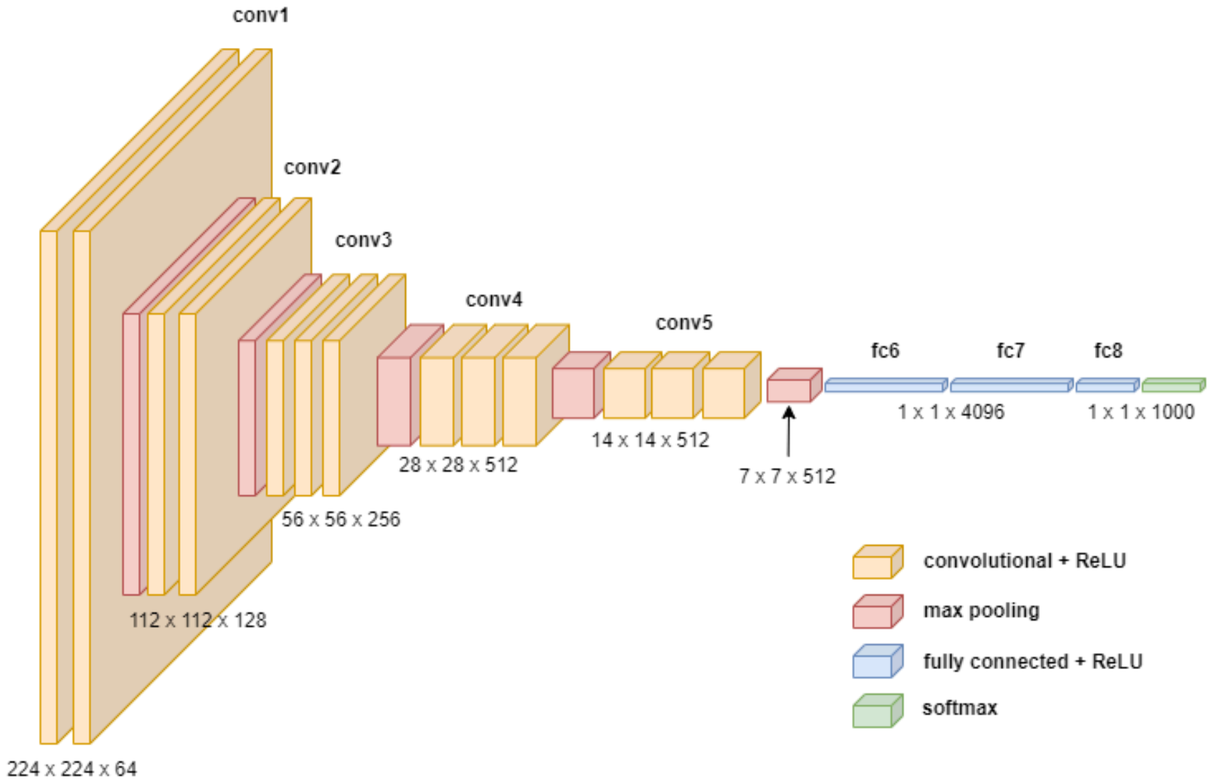


fig 5: CNN architecture

Dual Model and Post Prediction

We Constructed two Convolutional Neural Networks (CNNs) based on MobileNetV2.

Primary Model: Main classifier for sign recognition.

Verification Model: Secondary model for improving confidence and handling ambiguities.

Both models leverage transfer learning from ImageNet pre-trained weights.

Two-Phase Primary Model Training:

Phase 1 (Head Training): MobileNetV2 base is frozen; only newly added classification layers are trained.

Phase 2 (Fine-tuning): A portion of the MobileNetV2 base layers are unfrozen and trained with a very low learning rate for optimized performance.

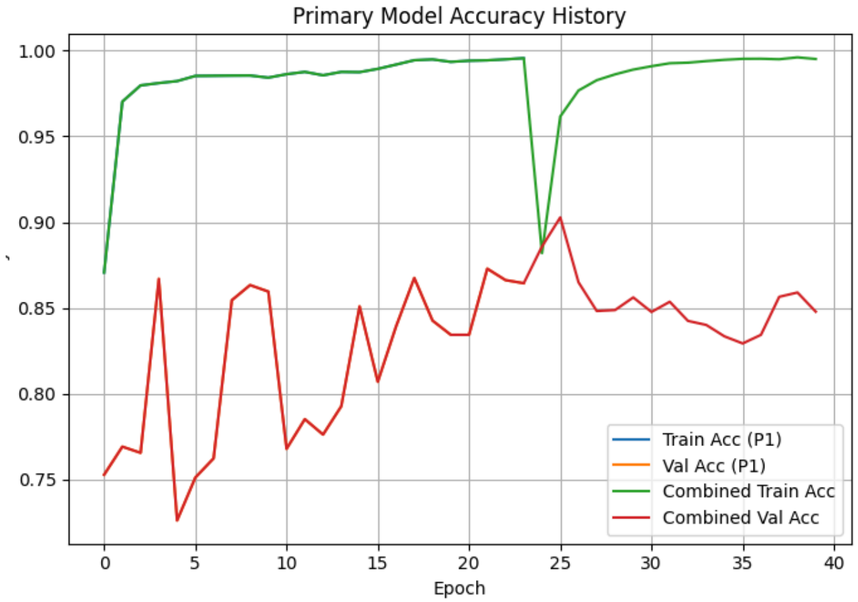
Sophisticated Post-Prediction Filtering:

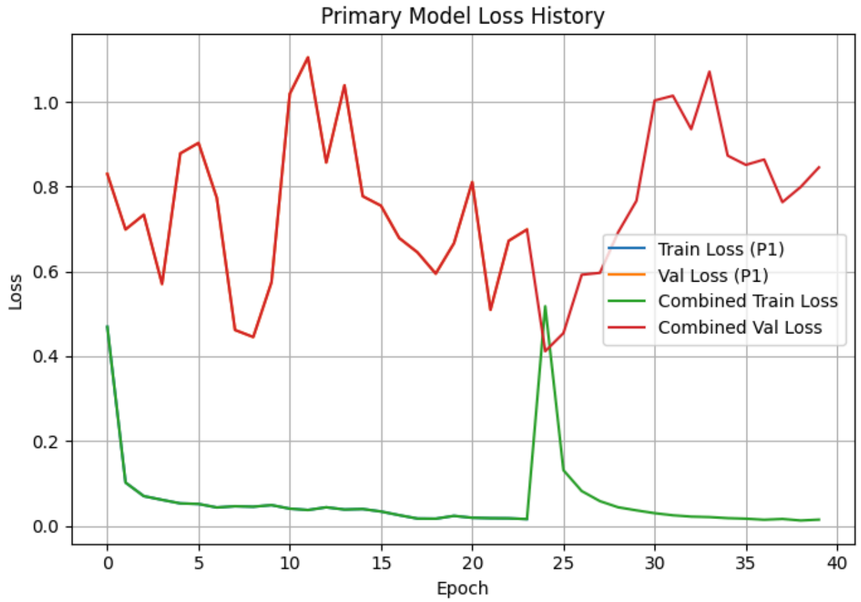
Applies a multi-stage re-checking mechanism on test set predictions.

Confidence-Based Verification: Low-confidence predictions from the primary model are re-evaluated by the verification model.

KNN-Based Refinement: For problematic/confusable sign pairs (e.g., 'M' vs. 'N') or secondary model disagreements, K-Nearest Neighbors (KNN) is employed on extracted features to further refine the prediction.

Graphical Representation of Model Accuracy And Model Loss





Result Analysis & Evaluation Metrics :

Recognition Performance:

- 95.8% accuracy with Layer 1 alone
- 97.61% accuracy after adding KNN and verification layers
- Outperforms existing models using only a standard webcam, without background subtraction or expensive sensors.

High Accuracy: Predictions match actual labels for most classes.

- Balanced Class Representation: No major bias detected.
- Areas for Improvement:
- Confusion between similar gestures (e.g., 'M' & 'N', 'S' & 'G').
- Feature refinement & preprocessing could enhance differentiation.

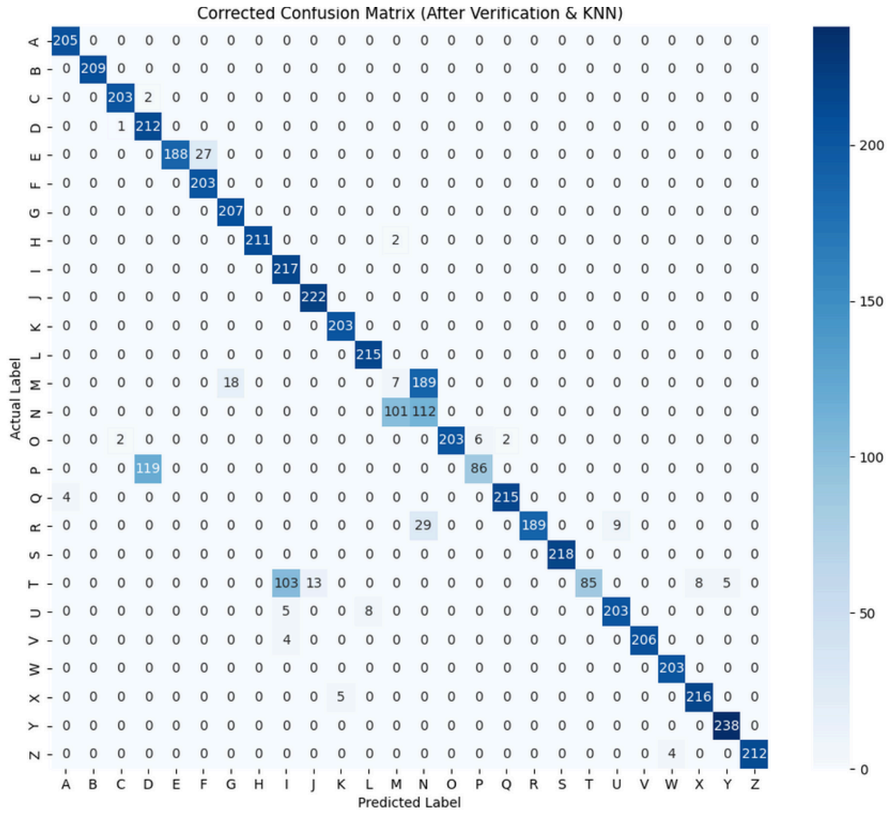


Fig 5: Confusion Matrix

RESULTS:

- **Webcam Integration:** Captures live video frames from the user's webcam.
- **User Control:** Webcam activation and deactivation via a simple checkbox.
- **Real-time Display:** Continuous, near real-time display of the live video stream.
- **Interactive Guidance:** Frames are flipped (mirror effect), and a Region of Interest (ROI) is drawn to guide hand placement.
- **Sign Prediction:** Displays the recognized sign label dynamically based on model output.

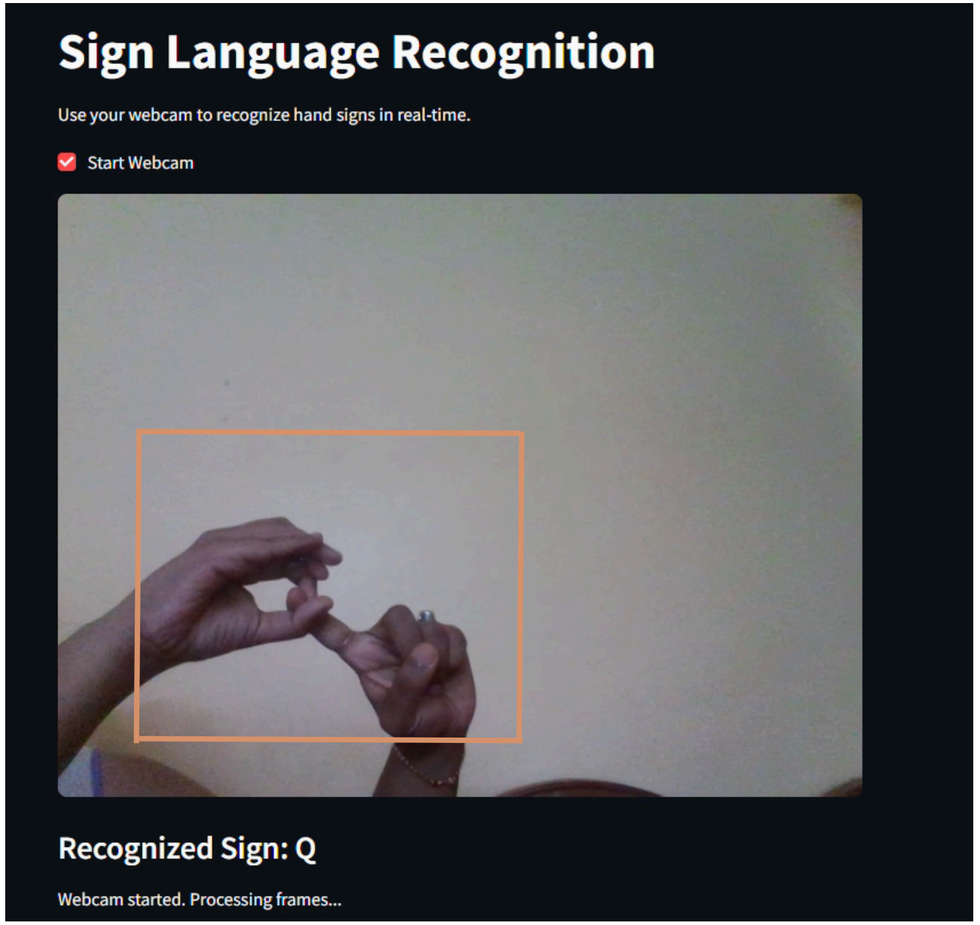


fig 5: Gui based output (Gesture Classification)

Conclusion And Future Scope :

Advancing Accessibility: CNN-based ISL recognition enhances inclusion for the deaf and hard-of-hearing community.

Key Achievements:

- High accuracy in recognizing hand shapes & gestures.
- Works across lighting conditions without specialized hardware.
- Affordable, real-time communication bridge between signers & non-signers.

Future Directions:

- Expand dataset with diverse hand shapes & backgrounds.
- Optimize accuracy with attention-based hybrid CNN architectures.
- Deploy on mobile & edge devices for real-time use.
- Enhance regional sign support with expert collaboration.
- Extend to multilingual systems for broader accessibility.

Thank You !